CSRI SUMMER PROCEEDINGS 2009

The **C**omputer **S**cience **R**esearch **I**nstitute
at Sandia National Laboratories

Editors:
Zhaofang Wen and S. Scott Collis
Sandia National Laboratories

January 11, 2010

SAND2010-3083P

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from

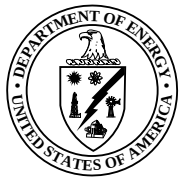
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/ordering.htm



Preface

The Computer Science Research Institute (CSRI) brings university faculty and students to Sandia National Laboratories for focused collaborative research on computer science, computational science, and mathematics problems that are critical to the mission of the laboratories, the Department of Energy, and the United States. CSRI provides a mechanism by which university researchers learn about and impact national– and global–scale problems while simultaneously bringing new ideas from the academic research community to bear on these important problems.

A key component of CSRI programs over the last decade has been an active and productive summer program where students from around the country conduct internships at CSRI. Each student is paired with a Sandia staff member who serves as technical advisor and mentor. The goals of the summer program are to expose the students to research in mathematical and computer sciences at Sandia and to conduct a meaningful and impactful summer research project with their Sandia mentor. Every effort is made to align summer projects with the student's research objectives and all work is coordinated with the ongoing research activities of the Sandia mentor in alignment with Sandia technical thrusts and the needs of the NNSA Advanced Scientific Computing (ASC) program that has funded CSRI from its onset.

Starting in 2006, CSRI has encouraged all summer participants and their mentors to contribute a technical article to the CSRI Summer Proceedings, of which this document is the fouth installment. In many cases, the CSRI proceedings are the first opportunity that students have to write a research article. Not only do these proceedings serve to document the research conducted at CSRI but, as part of the research training goals of CSRI, it is the intent that these articles serve as precursors to or first drafts of articles that could be submitted to peer–reviewed journals. As such, each article has been reviewed by a Sandia staff member knowledgeable in that technical area with feedback provided to the authors. Several articles have or are in the process of being submitted to peer–reviewed conferences or journals and we anticipate that additional submissions will be forthcoming.

For the 2009 CSRI Proceedings, research articles have been organized into the following broad technical focus areas — *computational mathematics and algorithms, discrete mathematics and informatics, architectures and systems software, and applications* and these areas are well aligned with Sandia's strategic thrusts in computer and information sciences.

We would like to thank all participants who have contributed to the outstanding technical accomplishments of CSRI in 2009 as documented by the high quality articles in this proceedings. The success of CSRI hinged on the hard work of over 25 enthusiastic student collaborators and their dedicated Sandia technical staff mentors. It is truly impressive that the research described herein occurred primarily over a three month period of intensive collaboration.

CSRI benefited from the administrative help of Deanna Ceballos, Bernadette Watts, Mel Loran, Dee Cadena, and Vonda Coleman. The success of CSRI is, in large part, due to their dedication and care, which are much appreciated. We would also like to thank those who reviewed articles for this proceedings — their feedback is an important part of the research training process and has significantly improved the quality of the papers herein. Finally, we want to acknowledge the ASC program for their continued support of the CSRI and its activities which have benefited both Sandia and the greater research community.

Zhaofang Wen
S. Scott Collis

January 11, 2010

Table of Contents

Computational Mathematics and Algorithms

Articles in this section focus on fundamental numerical algorithms ranging from mesh adaptation, optimal quadrature, and model reduction to numerical linear algebr, multigrid algorithms and uncertainty quantification that each have broad potential for application in a variety of computational disciplines.

Z. Wen
S.S. Collis

January 11, 2010

QUADRATIC TRIANGLE MESH UNTANGLING AND OPTIMIZATION VIA THE TARGET-MATRIX PARADIGM

PATRICK KNUPP*, NICHOLAS VOSHELL†, AND JASON KRAFTCHECK‡

Abstract. Meshes containing high-order nodes sometimes contain inverted elements due to the projection of coarse elements onto the domain boundary. Usually, such meshes are problematic and fixing the associated problems requires either re-meshing or post-processing techniques. Mesh optimization methods based on the Target-matrix paradigm are studied as a post-processing step to determine if the approach gives a viable solution to both untangling and improving the quality of a quadratic triangle mesh. Several preliminary algorithms are described, including one involving two successive optimizations. For that algorithm, a first optimization using a non-barrier size metric with a properly constructed target-matrix can potentially untangle the mesh. If the first optimization untangles the mesh, a second optimization of that result using a barrier-based shape metric can improve shape while keeping the mesh untangled. Numerical experiments applying the algorithms on planar quadratic triangle meshes demonstrate that optimization-based node-movement methods can successfully untangle and improve element shapes in high-order meshes. Further study of these algorithms is needed before they can be used with confidence on realistic meshes.

1. Introduction. To achieve high accuracy simulations, second-order finite elements are sometimes used. Unfortunately there are no reliable tools for meshing curved geometries known to the authors; however mesh generation packages usually follow an a posteriori approach to create boundary-conforming, quadratic element meshes on complex geometries by first creating elements with straight sides and then curving boundary sides by projecting mid-side and mid-face nodes onto the bounding geometry [11]. The quality of the resulting boundary elements can be poor, particularly if elements are large compared to the curvature of the associated geometry. This impacts simulation accuracy, efficiency and, if the Jacobian is negative, even the ability to proceed with the calculation.

The convergence theory of finite elements requires that mesh elements not be inverted. Most elements are defined based on a mapping from a logical element and the Jacobian matrix, J, of this mapping. The strict definition of an inverted element is that there exists a point (ξ, η) within the logical element such that $det(J_{\xi,\eta}) \leq 0$. If such a point does not exist, then the element is said to be non-inverted. Given an arbitrary linear planar triangle or quadrilateral, it is simple to determine from this criterion whether the element is inverted because the determinant is linear in the logical coordinates. However, for hexahedra, quadratic, and other element types, it is difficult to find robust and efficient numerical algorithms for determining whether a given element is valid.

The present work does not focus directly on the detection of inverted elements. Since it is possible to improve mesh quality without knowing beforehand if the mesh contains inverted elements, the question to be considered here is: given a mesh of quadratic elements, can we improve shape and size quality, such that the result rarely contains inverted elements? It is recognized that edge flipping and node insertion are powerful techniques in their own right. However, the approach taken here is based on a node movement strategy called the Target-matrix paradigm [6]. It is hoped that, once it is understood how to effectively optimize high-order element meshes using node movement alone, all these techniques can be combined into a single algorithm. The capability to optimize the quality of high-order finite element meshes by node-movement was recently added to the Mesquite mesh quality improvement library [1]. The present study uses the new capability to explore how one might improve meshes containing high-order nodes. For simplicity, attention is confined to planar quadratic triangle

*Sandia National Laboratories, pknupp@sandia.gov
†The Pennsylvania State University, njv116@cse.psu.edu
‡The University of Wisconsin, kraftche@cae.wisc.edu

meshes, but the approach is generalizable to other types of high-order elements. Studies involving other element types are planned for future work.

There have been limited studies that attempt to untangle or improve element shapes within meshes that include high-order elements (which are elements that have polynomials of degree greater than one defining their shape). Investigations defining valid regions for the placement of mid-nodes to ensure untangled elements (and other properties) were calculated in [13, 14], this could help in detecting untangled elements, or guiding vertex placement. In [9, 10], a mesh modification technique is employed to improve the quality of a high order element mesh (as well as enabling refinement and/or coarsening of the mesh). The technique was developed using vertex insertion and removal, flipping of edges and faces, along with other topological mesh modifications. One method for assessing the quality of quadratic triangle elements (that could be extended to give quality metrics for mesh optimization) was given in [12]. Proposed optimization-based node-movement strategies for improving quadratic meshes appear to be limited to [2, 3]. In [3], a linear quality metric was modified to be sensitive to high-order node positions by adding an angle-based penalty term. The quality metric was extended to quadrilaterals in [2] and a node-movement strategy that automatically switched between constrained Laplace smoothing and an optimization procedure based on the linear quality metric and the penalty term was applied to improve quadratic meshes. Limitations of the method include being applicable only to homogeneous, isotropic two-dimensional meshes.

2. Quadratic Elements. Using the standard basis vectors for $\mathbb{R}^2$ one can define a logical triangle which serves as the master element for the transformations used by quadratic triangles. The other elements that are of interest in this work are the active element (which is the element of interest in the mesh) and the target element (the desired element), which the active element is compared to. Any given quadratic triangle can then be defined by six points, corner vertices ($\mathbf{x}_0$, $\mathbf{x}_1$, and $\mathbf{x}_2$) and mid-edge nodes ($\mathbf{x}_3$, $\mathbf{x}_4$, and $\mathbf{x}_5$). A mapping from point $(\xi\,,\,\eta)$ of the logical triangle to point $\mathbf{X}$ of the quadratic triangle can then be defined using the following equations:

$$\mathbf{X}(\{\mathbf{x}_i\},\xi,\eta) = \mathbf{x}_0 + \mathbf{c}_1\xi + \mathbf{c}_2\eta + \mathbf{c}_3\xi\eta + \mathbf{c}_4\xi^2 + \mathbf{c}_5\eta^2 \tag{2.1}$$

$$\mathbf{c}_1 = -3\mathbf{x}_0 - \mathbf{x}_1 + 4\mathbf{x}_3 \tag{2.2}$$

$$\mathbf{c}_2 = -3\mathbf{x}_0 - \mathbf{x}_2 + 4\mathbf{x}_5 \tag{2.3}$$

$$\mathbf{c}_3 = 4(\mathbf{x}_0 - \mathbf{x}_3 + \mathbf{x}_4 - \mathbf{x}_5) \tag{2.4}$$

$$\mathbf{c}_4 = 2(\mathbf{x}_0 + \mathbf{x}_1 - 2\mathbf{x}_3) \tag{2.5}$$

$$\mathbf{c}_5 = 2(\mathbf{x}_0 + \mathbf{x}_2 - 2\mathbf{x}_5) \tag{2.6}$$

The vertices of the quadratic triangle are enumerated 0, 3, 1, 4, 2, 5 in counter-clockwise order (both vertices and mid-side nodes are in counter-clockwise order with mid-side node 3 following vertex 0). The Jacobian matrix for this transformation can be defined at each point within the logical triangle using the following equation:

$$J = [\,\mathbf{c}_1 + \mathbf{c}_3\eta + 2\mathbf{c}_4\xi\,,\;\mathbf{c}_2 + \mathbf{c}_3\xi + 2\mathbf{c}_5\eta\,] \tag{2.7}$$

Note that a linear mapping can be considered a special case of this in which the mid-nodes are mapped to the mean of the two corner vertices they are connected to. This causes $\mathbf{c}_3 = \mathbf{c}_4 = \mathbf{c}_5 = 0$, which gives a linear mapping function with the following constant Jacobian:

$$J = [\,\mathbf{c}_1\,,\;\mathbf{c}_2\,] \tag{2.8}$$

Given an active quadratic element (i.e., one that belongs to the mesh to be optimized) and a sample point (ξ_k, η_k) within the logical element, we have $A_k = J_{active}(\xi_k, \eta_k)$. Similarly, given

a quadratic target element, $W_k = J_{target}(\xi_k, \eta_k)$. Thus for every sample point, we have the pair A_k, W_k. When speaking in general about active and target-matrices, the sample point index is suppressed, giving A and W.

From this one can define the Jacobian matrix for the mapping from the target triangle to the active triangle (the inverse of logical $\mapsto$ target composed with logical $\mapsto$ active) as follows:

$$T = A\,W^{-1} \tag{2.9}$$

For an isotropic domain, there is no reason for the target element to break symmetry, so the equilateral triangle is a good choice for the target. For theoretical reasons explained in [4], we choose the area of the target equilateral element to be one-half. Then the Jacobian of the logical to target mapping (up to multiplication by some rotation matrix) is as follows:

$$W = \frac{1}{\sqrt{2}\sqrt[4]{3}} \begin{bmatrix} 2 & 1 \\ 0 & \sqrt{3} \end{bmatrix} \tag{2.10}$$

3. Mesh Optimization. Mesh optimization, as the term is used here, is a technique for modifying a mesh by moving vertices without changing the connectivity. In mesh optimization one defines an objective function to quantitatively compare the quality of alternative meshes. The optimal mesh has the best objective function score subject to constraints (on the position of boundary vertices, for example). The best valid objective function value is found using numerical optimization techniques. The objective function is usually some combination of measurements of the quality of individual components (such as elements or vertices) in the mesh. Most effort, as of this writing, has been on optimizing meshes composed of linear elements. For these linear meshes, good element quality metrics have been found, particularly to measure the shape of triangular elements.

In Mesquite there are four main options for the movement of each mesh vertex, which form optimization constraints. The four options are (a) fixed (no movement permitted), (b) free (any movement permitted), (c) constrained to the geometry (where the vertex is constrained to a lower dimensional space, such as the boundary), and (d) slaved (where a mid-side node is constrained to the midpoint of the neighboring vertices). The first three options apply to all vertices, while (d) applies only to mid-side nodes.

Some of the work on improving mesh quality has dealt with meshes where elements are inverted. For quadratic element meshes, the issue frequently arises during mesh creation, where curving the boundary of the initial linear mesh to conform to the geometry causes some elements to be poorly shaped or even inverted. To eliminate inverted elements from the mesh via optimization, a number of mesh untangling algorithms have been proposed (see [5] and the references therein). Unfortunately, none of these algorithms guarantee that the resulting mesh will, in fact, be untangled. Furthermore, by focusing exclusively on the inversion issue, the untangling techniques ignore other important aspects of mesh quality. To eliminate inverted elements and improve shape in quadratic element meshes we take a different approach that does not use a direct untangling algorithm. The approach in this work uses local quality metrics from the Target-matrix paradigm.

In the Target-matrix paradigm, each element of the mesh has a C^1 mapping from a target element to the active element. The Jacobian of this mapping (evaluated at sample point k) is the matrix, T_k. Then the quality at k is measured by local quality metric $\mu_k = \mu(T_k)$, which gives a non-negative real number. The metrics measure the degree to which A is 'close' to W, in terms of properties such as shape, size, and orientation.

There are two basic types of local metrics in the Target-matrix paradigm, those having 'barriers' and those which do not. Barrier metrics are used when the initial mesh is not

inverted; the barrier guarantees that the optimal mesh is not inverted at the sample points. Non-barrier metrics are used when the initial mesh contains one or more inverted sample points. The optimal mesh resulting from a non-barrier metric may or may not contain inverted elements. Since the initial quadratic meshes that we wish to improve are assumed to contain elements which are inverted, we use non-barrier metrics to optimize the initial mesh. If optimization with a non-barrier metric succeeds in creating a non-inverted mesh, the result can be further optimized using a barrier metric.

The first non-barrier metric of interest is the Shape (S) metric given by:

$$\mu_S(T) = \|T\|_F^2 - 2\,det(T) \tag{3.1}$$

In [6, 7] it was shown that $\mu_S \geq 0$ for all T and $\mu_S = 0$ if and only if T is a scaled rotation matrix. In that case, A has the form $A = sRW$ with arbitrary scalar s and arbitrary rotation R. Then the shape of the active matrix is the shape of the target-matrix, which in the present case corresponds to the equilateral triangle. This is a non-barrier metric, so it can potentially untangle an initially inverted mesh, as well as improve shape. The barrier form of the shape metric is

$$\mu_{SB}(T) = \frac{\|T\|_F^2}{2\,det(T)} \tag{3.2}$$

The second non-barrier metric of interest is the Size (Sz) metric, given by:

$$\mu_{Sz}(T) = (det(T) - 1)^2 \tag{3.3}$$

The metric obeys $\mu_{Sz} \geq 0$ for all T and $\mu_{Sz} = 0$ if and only if $det(T) = 1$, i.e., $det(A) = det(W)$. Thus, at the minimum, the local areas of the active and target matrices agree. Because this metric has no barrier, it can potentially untangle an inverted mesh as well as improve relative size. It does not, however, encourage the shape of the active element to be close to the shape of the target element. Thus, its primary use in the present application is to encourage mesh untangling because $det(W) = \frac{1}{2}$.

Finally, we consider the Shape+Size (SS) metric, and it's barrier form:

$$\mu_{SS}(T) = \|T\|_F^2 - 2\,\sqrt{\|T\|_F^2 + 2\,det(T)} + 2 \tag{3.4}$$

$$\mu_{SSB}(T) = \frac{1}{2\,det(T)}\mu_{SS}(T) \tag{3.5}$$

It was shown in the previously-cited references that the metric is non-negative and is minimized for $A = R\,W$ so that, at the minimum, both the shape and size of A agree with the shape and size of the Target-matrix. The metric can also be used to encourage the untangling of meshes.

Many of the issues involved in measuring sample point quality within an element are described in [8]. That work specifically considers quality metrics of the form investigated here, including results for quadratic triangular elements. It also introduces a relevant property known as label invariance. Note that x_0 can be assigned to any corner vertex, thus there are three labellings for the corner vertices and mid-side nodes of a quadratic triangle element that conform to the naming conventions. A local quality metric is label invariant if all three labellings of the active element give the same quality. Different conditions are derived for the metric formulation, sample point selection, and target element selection that guarantee label invariance of the local quality. Specifically, if the target element is a straight edged equilateral triangle and the sample points are selected to respect the symmetry of the element labellings, then the resulting quality metric will be locally label invariant.

An important issue concerns how to select the sample point locations within an element. As mentioned in the Introduction, detecting inverted elements requires, in principle, an infinite number of sample points. However, the present goal is not to detect inverted elements, but simply to improve quality and attempt to untangle inverted elements. Thus, we investigate whether it is sufficient to use a small number of sample points for this task. In particular, we choose to place sample points at all the corners and mid-side nodes of each triangular element in the mesh.[1] Numerical experiments will reveal whether this collection of sample points is sufficient for untangling realistic meshes.

Qualities, μ_k, at different sample points are combined into an objective function to be used in our numerical optimization procedure. To do so, this study uses a power-mean, giving the following objective function:

$$OF = \frac{1}{N} \sum_{Sample\ Point\ k} \mu_k \tag{3.6}$$

4. Numerical Examples. Because this is a preliminary investigation, the issue of algorithm efficiency is not considered. Instead, algorithm effectiveness (robustness) is investigated. The numerical optimizations were carried out using Mesquite's Feasible Newton solver, global patch smoothing, and a termination criterion of 400 iterations. The solutions were well-converged on the small meshes used in this study.

The a posteriori approach to quadratic mesh generation tends to localize poor quality (or inverted) elements on a curved boundary (rather than the interior of the mesh), which guides experiment selection. The initial experiment involves a small patch of six quadratic elements sharing a vertex, with a fixed boundary (see Figure 4.1), where the two triangles at the bottom of the patch are curved so that one element is initially inverted.

The figure depicts an investigation into which metrics and which combinations of free, fixed, and slaved vertices give the best results. The pictures are arranged in normal (left to right and top to bottom) order with the initial configuration first, then the results, with all mid-side nodes slaved, optimizing for Size, Shape, and Shape+Size (respectively). All of these meshes are inverted, however optimizing for Size comes closest to untangling, and optimizing for Shape+Size comes close to untangling while maintaining better shape. After these come the results optimizing for the Shape+Size metric, but freeing some mid-side nodes, first freeing only the interior mid-side nodes in the initially inverted element, then freeing all the internal mid-side nodes. Both of these final two techniques untangled the mesh (with the increased freedom creating more curved edges but better shape).

After modifying the geometry of this mesh a few times it was found that the Size metric untangled more cases than the Shape+Size metric (which untangled more cases than the Shape metric). An instructive investigation performed was one in which the leftmost vertex of the small patch was moved to the right and the mesh was optimized for different metrics with all internal mid-side nodes free. See figure 4.2 for a depiction of the results. A big challenge to untangling this patch is that it significantly reduces the quality of the two leftmost elements (especially the lower one) and perturbing the leftmost vertex exacerbates this competition (making untangling the element more challenging). Optimizing with the Shape+Size metric was unable to untangle perturbations less than 0.4, whereas optimizing with the Size metric was able to untangle a perturbation of 0.6 (but not 0.7). In both cases the point at which the problem with untangling first occurs is the lower left vertex (that starts out inverted). Note that there exists an untangled mesh even when the leftmost vertex has been moved by 0.7, so there are cases where optimizing for the Size metric is insufficient to find a valid untangled mesh configuration.

[1] Note that, in this approach, a mesh vertex can have multiple associated sample points, as can a mid-side node.

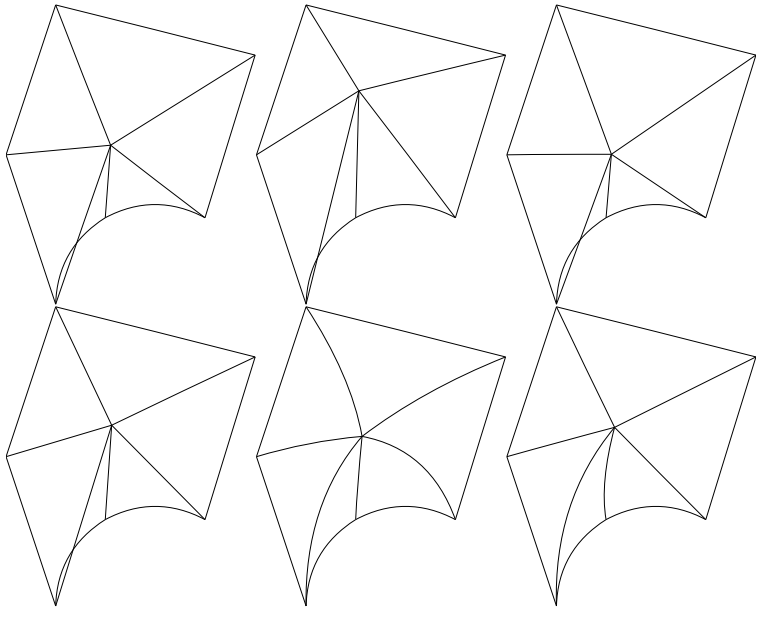


FIG. 4.1. *Small Mesh Experiment: The initial mesh is given in the upper left figure. Then the Shape, Size, and Shape+Size metrics are used (with all mid-nodes slaved) in the upper middle, upper right, and bottom left (respectively). The bottom row also gives the results with the Shape+Size metric with none of the mid-side nodes slaved (in the middle), and using a technique where the mid-side nodes are free in the initially inverted elements (but are otherwise slaved), at bottom right.*

Another, larger, mesh was created to give a more realistic test example and to emulate longer range interactions among multiple nodes, as is commonly seen within larger meshes. Based on smoothing tests it was confirmed that the Size metric was most useful in untangling the mesh, with the downside that it tends to have poorly shaped elements. Further investigation revealed that, in many cases, an untangled mesh with nicely shaped elements could be found by first optimizing with the Size metric (to untangle), then optimizing with the Size Barrier or Shape+Size Barrier metric (to improve element shapes while preventing inversions). One drawback of this approach is that the second optimization causes many vertices to return near to their position before the first optimization except, of course, for the initially inverted regions of the mesh. The approach can potentially be improved by halting the first optimization pass as soon as the mesh becomes untangled. Other approaches within the Target-matrix paradigm can also be envisioned.

Some variations of the larger mesh were created, which increased heterogeneity in element sizes, to investigate the importance of reference element size. One example is depicted in figure 4.3. The intuition is that an inverted sample point will have a small negative $det(T)$ value. Comparing this to the $det(T)$ value for an element that is optimally shaped, but scaled much larger than the target element, the $det(T)$ value for the inverted sample point will be closer to the ideal value (of 1). These effects may cause the optimization to move vertices more to improve elements that aren't inverted, preventing the untangling of the mesh. However the meshes created to explore this effect didn't substantially affect the ability of the optimization process to untangle the mesh. Further experiments with more heterogeneity could alter this conclusion.

Looking at figure 4.3, the experiment depicted starts with an initial mesh and looks into

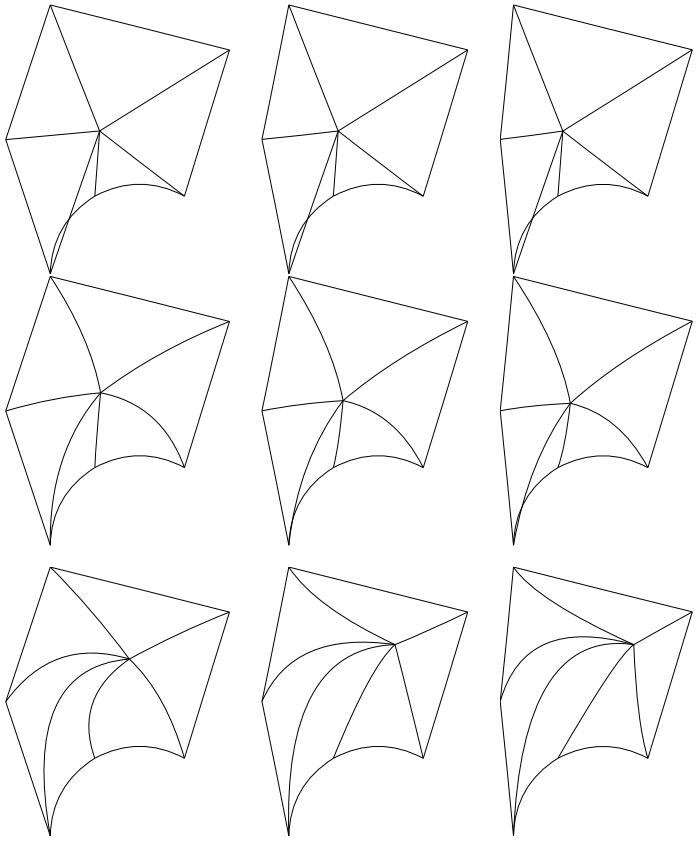


FIG. 4.2. *Right Perturbation: The leftmost point of the small mesh is perturbed to the right and results are depicted for optimizing the mesh using two different metrics. The initial mesh is shown in the top row, the Shape+Size optimized mesh is shown in the middle row, and the Size optimized mesh is depicted in the bottom row. The left column shows the unperturbed patch; in this case both metrics led to an untangled mesh. The middle column shows results for a perturbation of* 0.4*, in this case the Size optimized mesh is untangled but the Shape+Size optimized mesh is inverted. The right column shows the resulting meshes for a perturbation of* 0.7*, where both the Size and Shape+Size optimized meshes are inverted.*

different metrics for the second optimization pass. The initial mesh is shown in the upper left corner, the areas range from around $\frac{1}{8}$ to around 4 times the area of the target element. It also has three inverted elements. The results after the first optimization pass, using the Size metric, is depicted in the upper right of the figure. The mesh has been untangled, but the element shapes have become quite poor (especially at right, where many small elements have become stretched out). As has been described previously, the second optimization pass then optimizes for a barrier metric to improve shape without inverting the mesh. Two metrics were investigated for the second pass, the Size Barrier metric (at lower left) and the Shape+Size Barrier metric (at lower right). The results indicate no clear winner for the second pass metric, but do show that the two pass approach untangles the mesh and improves element shapes. One alternative approach might be to intelligently slave more of the interior nodes during the first step in which one optimizes with Size, to reduce the number of degrees of freedom and to avoid the creation of unnecessarily curved edges.

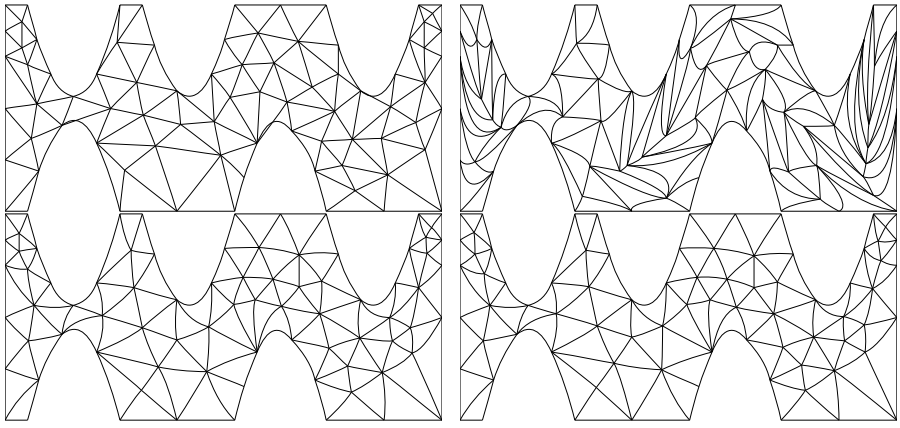


FIG. 4.3. *Heterogeneous Mesh Experiment: The initial mesh is given in the upper left figure. Then the Size metric is used (with no mid-side nodes slaved), and shown in the upper right. Optimization with other metrics didn't untangle the mesh, demonstrating the tendency of the Size techniques to perform better at untangling the samples in this study. The Size optimized mesh is then used as input for optimization using the Shape Barrier (bottom left) and Shape+Size Barrier (bottom right). Both of these 'two pass' results produce untangled meshes with improved shape compared to either the initial mesh or the Size optimized result.*

5. Conclusions and Future Work. The Target-matrix paradigm provides an approach to the smoothing of quadratic triangle meshes that involves the use of a target matrix based on an equilateral element; Size, Shape, and Shape+Size local quality metrics (both barrier and non-barrier); and an objective function that is defined in terms of the average of the sample point quality metrics. For each element, six sample points were selected, one at each corner vertex and one at each mid-side node. Non-barrier metrics are used on the initial mesh since it is expected to contain inverted elements. One promising strategy in eliminating inverted elements and improving shape is to fix all the boundary nodes, free all the interior corner and mid-side nodes, and optimize using the Size metric first. That tends to make local sizes equal and thus to remove outliers such as those corresponding to inverted elements. This is followed by an optimization of the untangled mesh with a barrier metric to improve Shape or Shape+Size. The optimizations did not make use of the concept of element quality; instead, optimizing local quality was sufficient to untangle and improve the test meshes. Of course, this method does not work on all triangle meshes, but the results suggest that this approach may lead to a useful algorithm.

The formulation of the objective function and its component parts in this study can be readily extended to other element types such as high-order quadrilaterals, tetrahedra, and hexahedra. The effectiveness of the algorithm on such elements remains to be determined. It may also be worthwhile to investigate different combinations of fixed, free, and slaved vertices in order to maintain as many straight-sided elements in the interior as possible, along with allowing boundary nodes to move along their associated boundary instead of being fixed. Future work also includes investigations of different sets of sample points; using an effective untangling metric; optimization based on element quality instead of sample point quality; and applications to real meshes.

REFERENCES

[1] M. BREWER, L. DIACHIN, P. KNUPP, T. LEURENT, AND D. MELANDER, *The Mesquite mesh quality improvement toolkit*, Proceedings of the 12th International Meshing Roundtable, (2003), pp. 239–250.

[2] Z. Chen, J. Tristano, and W. Kwok, *Combined Laplacian and optimization-based smoothing for quadratic mixed surface meshes*, Proceedings of the 12th International Meshing Roundtable, (2003), pp. 361–370.
[3] ———, *Construction of an objective function for optimization-based smoothing*, Engr. w/Cmptrs., 20 (2004), pp. 184–192.
[4] P. Knupp, *Construction of target-matrices for the target-matrix paradgim*, manuscript in progress.
[5] ———, *Hexahedral mesh untangling & algebraic mesh quality metrics*, Proceedings of the 9th International Meshing Roundtable, (2000), pp. 173–183.
[6] ———, *Local 2D metrics for mesh optimization in the target-matrix paradigm*, Sandia National Laboratories, (2006), pp. SAND2006–7382J.
[7] ———, *Analysis of 2D, rotation-invariant, non-barrier metrics in the target-matrix paradigm*, Sandia National Laboratories, (2008), pp. SAND2008–8219P.
[8] ———, *Label-invariant mesh quality metrics*, to appear in Proceedings of the 18th International Meshing Roundtable, (2009).
[9] X. Luo, M. Shephard, L. Lee, L. Ge, and C. Ng, *Moving curved mesh adaptation for higher order finite element simulations*, Engineering with Computers, submitted., (2006), pp. 42–142.
[10] X. Luo, M. Shephard, L. Lee, C. Ng, and L. Ge, *Tracking adaptive moving mesh refinements in 3d curved domains for large-scale higher order finite element simulations*, Proceedings of the 17th International Meshing Roundtable, (2008), pp. 585–601.
[11] S. Robert, R. O'bara, and M. Shephard, *Curvilinear mesh generation in 3D*, Proceedings of the 8th International Meshing Roundtable, (1999), pp. 407–417.
[12] A. Salem, S. Canann, and S. Saigal, *Robust distortion metric for quadratic trianglular 2D finite elements*, Trends in Unstructured Mesh Generation, ASME, AMD-Vol. 220 (1997), pp. 73–80.
[13] ———, *Mid-node admissible spaces for quadratic triangular arbitrarily curved 2D finite elements*, Int. J. Numerical Methods in Engineering, 50(2) (2001), pp. 253–272.
[14] A. Salem, S. Saigal, and S. Canann, *Mid-node admissible space for 3D quadratic tetrahedral finite elements*, Engineering with Computers, 17 (2001), pp. 39–54.

A COMPARISON OF NONLOCAL DIFFUSION EQUATIONS TO THEIR CLASSICAL ANALOGS

NATHANIAL J. BURCH[†] AND RICHARD B. LEHOUCQ[‡]

Abstract. In this note, we investigate relationships between nonlocal diffusion equations and their classical, local, analogs. For instance, we show how the latter can be derived from the former via truncation of higher-order statistical moments in a Kramers-Moyal expansion. Further, we demonstrate, via random walks, Lévy processes, and anomalous diffusion, that nonlocal diffusion equations generalize their classical analogs. We survey selected theory of nonlocal diffusion equations and other emerging theories, such as nonlocal Green's identities. From which, we formulate a variational approach for nonlocal diffusion equations and discuss a Galerkin finite element formulation. A means for comparing finite element approximate solutions of the nonlocal diffusion equation to those of the associated classical equation is discussed and related to existing convergence results.

1. Introduction. Recently, nonlocal diffusion equations, e.g.,

$$\frac{\partial u}{\partial t} = \int_{\mathbb{R}^n} (u(x+s,t) - u(x,t))\phi(s)\,\mathrm{d}s, \tag{1.1}$$

have been used as a model for various diffusion processes. A discrete-time variation of (1.1) is used in [6] to study a single species population model following spatial redistribution and local regulation laws. In [2, 3, 4], nonlocal effects arising from (1.1), e.g., diffusion correlations, are investigated for the modeling of diffusion in fluids with large mean-free-paths. Further, in [30], integrodifferential equations, such as (1.1), arise as models for the membrane potential of nerve cells in neural networks. Moreover, applications of (1.1) to nonlocal image processing are discussed in [15].

We contrast (1.1) with the classical, local, diffusion equation,

$$\frac{\partial u}{\partial t} = D\Delta u. \tag{1.2}$$

In the first part of this paper, we address several relationships between (1.1) and (1.2), establishing connections with random walks, Wiener and Lévy processes [8, 14, 18, 19, 23], master equations [14, 17, 29], Fokker-Planck equations [8, 14], stochastic differential equations [22, 24], anomalous diffusion processes [5, 18, 23], and fractional diffusion equations [5, 19, 25]. With these perspectives, we gain an understanding of how (1.2) can be presented as a special case of (1.1). Moreover, we investigate potentially important dynamics that are discarded when using (1.2) in place of (1.1).

In the second part of this paper, we present selected theory of nonlocal diffusion equations from [1, 7, 9, 10] and introduce Dirichlet and Neumann nonlocal boundary value problems, which are nonlocal analogs of the classical Dirichlet and Neumann problems. Finally, we discuss the emergence of a nonlocal calculus [16], demonstrate its relationship to the problem at hand, and perform a variational analysis. The latter is necessary for implementing Galerkin finite element methods to approximate solutions to nonlocal diffusion boundary value problems, which we briefly motivate.

2. Nonlocal Diffusion Equations and Classical Diffusion Equations.

2.1. Derivation of the Nonlocal Diffusion Equation via a Random Walk. Let $u(x,t)$ describe a density of particles at position x and time t, where a particle may represent, for instance, a molecule in a fluid, an individual in a population, or a probability. We model the

[†]Colorado State University, Department of Mathematics, Department of Statistics, burchn@lamar.colostate.edu
[‡]Sandia National Laboratories, rblehou@sandia.gov

movement of each particle with a discrete-time, discrete-space random walk on the grid $k\mathrm{d}x$, $k \in \mathbb{Z}$. That is, in a short time interval τ, each particle moves a distance $k\mathrm{d}x$, independent of all other particles, with probability ψ_k. For future use, let $\phi_\tau(s)$, $s \in \mathbb{R}$, be a symmetric probability density function and

$$\psi_k = \int_{(k-1/2)\mathrm{d}x}^{(k+1/2)\mathrm{d}x} \phi_\tau(s)\,\mathrm{d}s \approx \phi_\tau(k\mathrm{d}x)\mathrm{d}x.$$

Thus, the density of particles at position x and time $t+\tau$ satisfies [27]

$$u(x,t+\tau) = u(x,t) + \sum_{k\neq 0} u(x+k\mathrm{d}x,t)\psi_k - \sum_{k\neq 0} u(x,t)\psi_k. \tag{2.1}$$

The terms on the right hand side of (2.1), from left to right, represent the particles present at position x and time t, particles arriving to position x from other positions during the time interval τ, and particles leaving position x during the time interval τ. Combining the sums in (2.1) yields

$$u(x,t+\tau) - u(x,t) = \sum_{k\in\mathbb{Z}} (u(x+k\mathrm{d}x,t) - u(x,t))\psi_k. \tag{2.2}$$

Taking $\mathrm{d}x \to 0$, we obtain

$$u(x,t+\tau) - u(x,t) = \int_{\mathbb{R}} (u(x+s,t) - u(x,t))\phi_\tau(s)\,\mathrm{d}s, \tag{2.3}$$

which is the analog of (2.2) for a discrete-time, continuous-space random walk on $\mathbb{R}$ governed by the probability density function ϕ_τ. Because ϕ_τ is a probability density function, we have that (2.3) is equivalent to

$$u(x,t+\tau) = \int_{\mathbb{R}} u(x+s,t)\phi_\tau(s)\,\mathrm{d}s, \tag{2.4}$$

which is an equation familiar to several disciplines. For instance, (2.4) is often referred to as Einstein's master equation, or just a master equation, which Albert Einstein used to model the movement of particles suspended in a fluid in his celebrated seminal paper on the theory of Brownian motion [11].

Under appropriate conditions[1], dividing (2.3) by τ yields the one-dimensional nonlocal diffusion equation (1.1) in the limit as $\tau \to 0$. The integrodifferential equation (1.1) has several aliases, such as master equation, Smoluchowski equation, and nonlocal diffusion equation, which we use interchangeably.

2.2. Arrival at the Classical Diffusion Equation via a Kramers-Moyal Expansion. It is well understood in the literature [8, 27] that the classical diffusion equation can be derived through a special case of (2.2) when $\psi_1 = \psi_{-1} = 1/2$ and all other ψ_k's are zero. In this case, (2.2) reduces to

$$u(x,t+\tau) - u(x,t) = \frac{1}{2}(u(x+\mathrm{d}x,t) - 2u(x,t) + u(x-\mathrm{d}x,t)).$$

Dividing by τ and assuming the scaling of time and space satisfies

$$\mathrm{d}x^2 = 2D\tau,$$

[1] See [14, pp. 47-52], or [28], for a detailed account of these conditions and the ensuing derivation.

where D is the diffusion coefficient, we obtain (1.2) in one dimension in the limit of $\tau \to 0$. Thus, both the nonlocal diffusion and the classical diffusion equations can be derived through a random walk, with the latter arriving from a specific, localized, random walk. By localized, we are referring to each particle's random walk motion being restricted to immediate neighboring nodes only. In this light, (1.1) is seen as a generalization of (1.2).

Another relationship between (1.1) and (1.2) shows how the latter is derived from the former. Assuming appropriate regularity, we perform a Kramers-Moyal expansion [14, 29] on the right hand side of (1.1) to obtain

$$\begin{aligned}\frac{\partial u}{\partial t} &= \int_{\mathbb{R}} \left[s\frac{\partial u}{\partial x} + \frac{s^2}{2!}\frac{\partial^2 u}{\partial x^2} + \frac{s^3}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{s^4}{4!}\frac{\partial^4 u}{\partial x^4} + \cdots \right] \phi(s)\,\mathrm{d}s \\ &= \frac{\partial^2 u}{\partial x^2}\left(\frac{\mathbb{E}(s^2)}{2!}\right) + \frac{\partial^4 u}{\partial x^4}\left(\frac{\mathbb{E}(s^4)}{4!}\right) + \frac{\partial^6 u}{\partial x^6}\left(\frac{\mathbb{E}(s^6)}{6!}\right) + \cdots, \end{aligned} \tag{2.5}$$

where the odd moments disappear due to symmetry of ϕ. We suppose

$$\mathbb{E}\left(s^2\right) = 2D \tag{2.6}$$

and that the larger than second-order moments of s are negligible. Thus, truncating (2.5) again yields (1.2), which illustrates how, under appropriate regularity conditions, the classical diffusion equation is a second-order truncation of a series representation of the nonlocal diffusion equation. Einstein, in [11], justified the truncation of (2.5) by assuming "$\phi(s)$ only differs from zero for very small values of s" so that "every succeeding term is very small compared with the preceding." Truncating (2.5) after more terms than just the first gives rise to a family of higher-order differential equations [26].

To reemphasize the important point, although the origins of both (1.1) and (1.2) can be traced back to a random walk, the former should be understood as either a special case or a second-order truncation of the latter. In the next section, we further develop this understanding through a discussion of stochastic processes. The validity of truncating (2.5) to obtain (1.2) has been a recent focus of attention. For instance, the authors in [2, 3, 4] show that such a truncation is not valid for diffusion in fluids with large mean-free-path by comparing analytic solutions (1.1) and (1.2), thus providing a convincing argument that nonlocal effects present in (1.1) should not be ignored.

2.3. Master Equations, Fokker-Planck Equations, and Stochastic Differential Equations. The master equation (1.1) describes the time evolution of the probability density function, $u(x,t)$, of the state of a system obeying a pure-jump stochastic process [14]. Such a process, a specific type of Lévy process[2], can be shown to have discontinuous sample paths and is encountered when modeling anomalous diffusion, turbulent fluid flow, and ballistic particle diffusion, to name a few [5]. In contrast, the approximating classical diffusion equation (1.2) is a Fokker-Planck equation [14, 29], named after Adriaan Fokker and Max Planck. As such, (1.2) describes the time evolution of the probability density function, $u(x,t)$, of the state of a system obeying an Itō stochastic differential equation (SDE). That is, suppose that the state of a system, X_t, is given by the Itō SDE

$$\mathrm{d}X_t = \mu(X_t,t)\mathrm{d}t + \sigma(X_t,t)\mathrm{d}W_t, \tag{2.7}$$

where W_t is a standard Wiener process[3] [8, 22, 24]. Loosely, (2.7) specifies that the amount X_t changes in a small time increment is distributed as $\mathrm{N}(\mu(X_t,t), \sigma^2(X_t,t)t)$, where $\mu(X_t,t)$

[2] A Lévy process is a general class of stationary stochastic processes restricted to start at the origin, have independent increments, and be right continuous with left limits.

[3] A Wiener process is a specific case of a Lévy process in which any increment, $W_t - W_s$, is a zero-mean Gaussian random variable with variance $t - s$.

and $\sigma(X_t, t)$ are commonly referred to as the drift and diffusion coefficients of the process, respectively. Then, the Fokker-Planck equation associated with (2.7) is

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(\mu(x,t)u(x,t)) + \frac{\partial^2}{\partial x^2}\left(\frac{1}{2}\sigma^2(x,t)u(x,t)\right). \tag{2.8}$$

Using the symmetry of ϕ, which induces a zero drift in the Itō SDE, and assumption (2.6), (2.7) becomes

$$\mathrm{d}X_t = \sqrt{2D}\mathrm{d}W_t, \tag{2.9}$$

while (2.8) becomes (1.2). The solution of (2.7) can be shown to have continuous sample paths almost surely. Consequently, although the underlying jump process of (1.1) has discontinuous paths, the underlying Itō process of (1.2) has continuous paths, identifying a limitation of using (1.2) in place of (1.1)

2.4. Anomalous Diffusion and Fractional Diffusion Equations. To understand the relationship between diffusion and the master equation we invoke some basic theory of Fourier transforms. Taking the Fourier transform of (1.2),

$$\frac{\partial \widehat{u}}{\partial t} = -Dk^2\widehat{u}(k,t), \tag{2.10}$$

where $\widehat{u}(k,t)$ represents the Fourier transform in space of the function $u(x,t)$. Solving this ordinary differential equation with initial condition $u(x,0) = \delta(x)$ gives

$$\widehat{u}(k,t) = \frac{1}{\sqrt{2\pi}}\exp(-Dk^2t),$$

which can be inverted to yield the fundamental solution of (1.2), denoted with g,

$$g(x,t) = \frac{1}{\sqrt{4\pi Dt}}\exp\left(-\frac{x^2}{4Dt}\right). \tag{2.11}$$

From (2.11), given that a particle begins at the origin, its position after time t is given by a zero-mean Gaussian random variable with variance $2Dt$, characteristic of Brownian motion, as witnessed also in (2.9). For (1.2) with $u(x,0) = u_0(x)$,

$$u(x,t) = (u_0 * g)(x,t) = \frac{1}{\sqrt{4\pi Dt}}\int_{\mathbb{R}}\exp\left(-\frac{(x-y)^2}{4Dt}\right)u_0(y)\,\mathrm{d}y. \tag{2.12}$$

Rewriting (2.12),

$$u(x,t) = \int_{\mathbb{R}} g(y,t)u_0(x-y)\,\mathrm{d}y,$$

which suggests that the solution $u(x,t)$ is the probabilistic expectation of the initial density of particles at position $x-y$ that have diffused to position x in time t through Brownian motion.

Now, consider the nonlocal diffusion equation (1.1). Following work in [5], we define the function

$$K(s) = \phi(s) - \delta(s). \tag{2.13}$$

Using (2.13), simple calculations give a convenient form of (1.1),

$$\frac{\partial u}{\partial t} = \int_{\mathbb{R}} u(x+s,t)K(s)\,\mathrm{d}s = (K * u)(x,t), \tag{2.14}$$

where the last equality follows from a change of variables and the symmetry of K. Taking the Fourier transform of the convolution in (2.14) gives

$$\frac{\partial \widehat{u}}{\partial t} = \sqrt{2\pi}\widehat{K}\widehat{u}.$$

Notice that if $\widehat{K} \equiv -(2\pi)^{-1/2}Dk^2$, this situation collapses to the classical diffusion process. When this is not the case, for instance when

$$\widehat{K} = -(2\pi)^{-1/2}D|k|^{\alpha_L}, \text{ for } \alpha_L \in (0,2),$$

more interesting diffusion processes can be described. Such diffusion processes are referred to as anomalous diffusion processes [5, 19, 20, 21, 23]. In this case, we solve the ordinary differential equation

$$\frac{\partial \widehat{u}}{\partial t} = -D|k|^{\alpha_L}\widehat{u}, \tag{2.15}$$

with initial condition $u(x,0) = \delta(x)$, yielding

$$\widehat{u}(k,t) = \frac{1}{\sqrt{2\pi}}\exp\left(-D|k|^{\alpha_L}t\right).$$

Then, the fundamental solution, denoted with g_{α_L}, satisfies

$$\widehat{g}_{\alpha_L}(k,t) = \frac{1}{\sqrt{2\pi}}\exp\left(-D|k|^{\alpha_L}t\right). \tag{2.16}$$

Unfortunately, except for special cases of α_L, finding a closed form of $g_{\alpha_L}(x,t)$ is nontrivial [5, 19, 23]. When normalized appropriately, $g_{\alpha_L}(x,t)$ is a centered and symmetric stable distribution, or Lévy distribution, characterized in [18, 19] by (2.16).

The case $\alpha_L = 2$ corresponds to the classical diffusion process, i.e., particles follow Brownian motion, in which their mean-square displacement is $2Dt$ and sample paths are almost surely continuous. However, when $\alpha_L < 2$, particles follow a Lévy process. Such Lévy distributions have power-law tails, so that the mean-square displacement of a particle is infinite. Consequently, sample paths of a Lévy process differ significantly from those of Brownian motion, e.g., are discontinuous, due to the presence of more frequently occurring long jumps. Finally, one can show that $g_{\alpha_L}(x,t)$ satisfies a fractional diffusion equation [23], identifying an important relationship between nonlocal diffusion equations, Lévy processes, and fractional diffusion equations [5, 19, 25].

3. Selected Theory of Nonlocal Diffusion Equations.

3.1. The Nonlocal Cauchy Problem. Performing a simple change of variables in (1.1) and adding an initial condition, we arrive at the nonlocal Cauchy problem [7]

$$\begin{cases} u_t(x,t) = \displaystyle\int_{\mathbb{R}^n} (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y, & x \in \mathbb{R}^n \\ u(x,0) = u_0(x), & x \in \mathbb{R}^n. \end{cases} \tag{3.1}$$

The authors in [7] show that if $\phi \in \mathcal{S}(\mathbb{R}^n)$, the Schwartz space of rapidly decreasing functions, then the solution to (3.1) with $u_0(x) = \delta(x)$ admits

$$u(x,t) = e^{-t}\delta(x) + v(x,t), \tag{3.2}$$

where v is smooth. Unlike the fundamental solution of the classical equation (2.11), the delta function remains in the fundamental solution of the Cauchy problem (3.2), decaying exponentially for all time. This indicates that the solution to (3.1) is as regular, but no more, than the initial distribution $u_0(x)$. The classical diffusion equation, on the other hand, immediately smoothes, or regularizes, the initial condition. A special case of this phenomenon is observed in [4] for diffusion profiles in fluids with large mean-free-paths.

In what follows, we add appropriate nonlocal boundary conditions to (3.1) to arrive at nonlocal Dirichlet and Neumann boundary value problems, analogous to the classical Dirichlet and Neumann problems. In [7], existence and uniqueness of solutions to such nonlocal boundary value problems is demonstrated via a Banach fixed point theorem [13] argument. For the duration of this paper, we assume that $\Omega \subseteq \mathbb{R}^n$ is a bounded and smooth domain, $\phi : \mathbb{R}^n \to \mathbb{R}$ is a radial probability density function that is strictly positive in $B(0,d)$ and vanishes elsewhere, and that

$$\widetilde{\Omega} \subseteq \{x \in \Omega \,|\, \operatorname{dist}(x, \partial\Omega) > d\}. \tag{3.3}$$

3.2. The Homogeneous Nonlocal Dirichlet Boundary Value Problem. Consider the homogeneous nonlocal Dirichlet boundary value problem [7]

$$\begin{cases} u_t(x,t) = \displaystyle\int_\Omega (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y, & x \in \widetilde{\Omega}, \\ u(x,t) = 0, & x \notin \widetilde{\Omega}, \\ u(x,0) = u_0(x), & x \in \Omega. \end{cases} \tag{3.4}$$

Since the integral in (3.4) is over Ω, particles may exit the region $\widetilde{\Omega}$. However, particles entering $\widetilde{\Omega}$ is prohibited by the condition that u vanishes outside of $\widetilde{\Omega}$, which is seen by rewriting the integral in (3.4) to obtain

$$u_t(x,t) = \int_{\widetilde{\Omega}} (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y - u(x,t)\int_{\Omega\setminus\widetilde{\Omega}} \phi(x-y)\,\mathrm{d}y. \tag{3.5}$$

An application of Fubini's theorem, followed by a change of variables, shows

$$\int_{\widehat{\Omega}}\int_{\widehat{\Omega}} (u(y,s) - u(x,s))\phi(x-y)\,\mathrm{d}y\,\mathrm{d}x = 0, \quad \forall \widehat{\Omega} \subseteq \Omega. \tag{3.6}$$

Integrating (3.5) with respect to time, then with respect to x, applying Fubini's theorem, and utilizing (3.6), we arrive at

$$\int_{\widetilde{\Omega}} u(x,t)\,\mathrm{d}x = \int_{\widetilde{\Omega}} u_0(x)\,\mathrm{d}x - \int_0^t \int_{\widetilde{\Omega}}\int_{\Omega\setminus\widetilde{\Omega}} u(x,s)\phi(x-y)\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}s. \tag{3.7}$$

Applying the maximum principle for nonlocal diffusion equations [12], the rightmost integrand in (3.7) is nonnegative and we conclude that the integrated density over $\widetilde{\Omega}$ decreases with time. The unique stationary solution of (3.4) is $u \equiv 0$, as (3.7) and experience with the classical homogeneous Dirichlet problem suggest.

3.3. The Homogeneous Nonlocal Neumann Boundary Value Problem. Next, consider the homogeneous nonlocal Neumann boundary value problem [7]

$$\begin{cases} u_t(x,t) = \displaystyle\int_\Omega (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y, & x \in \Omega, \\ u(x,0) = u_0(x), & x \in \Omega. \end{cases} \tag{3.8}$$

Since the integral in (3.8) is over Ω, diffusion only occurs in Ω, i.e., no particles enter or exit the region Ω. This condition is analogous to the Neumann boundary condition for the classical diffusion equation. Consequently, we expect the integrated density over Ω to be constant. To verify this, we integrate (3.8) with respect to time, then with respect to x, and apply Fubini's theorem to obtain

$$\int_\Omega u(x,t)\,\mathrm{d}x = \int_\Omega u_0(x)\,\mathrm{d}x + \int_0^t \int_\Omega \int_\Omega (u(y,s) - u(x,s))\phi(x-y)\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}s.$$

Using (3.6)

$$\int_\Omega u(x,t)\,\mathrm{d}x = \int_\Omega u_0(x)\,\mathrm{d}x$$

and conclude that the integrated density over Ω is constant for all time t. Stationary solutions to (3.8) are constant in Ω and given by

$$u(x) = \frac{1}{|\Omega|}\int_\Omega u_0(x)\,\mathrm{d}x.$$

3.4. The Inhomogeneous Nonlocal Neumann Boundary Value Problem. We expand our analysis from the previous section and consider the inhomogeneous nonlocal Neumann boundary value problem [7]

$$\begin{cases} u_t(x,t) = \displaystyle\int_\Omega (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y + \int_{\mathbb{R}^n\setminus\Omega} G(y,t)\phi(x-y)\,\mathrm{d}y, & x \in \Omega, \\ u(x,0) = u_0(x), & x \in \Omega. \end{cases}$$

As before, the first integral illustrates that diffusion occurs exclusively in Ω. The second integral represents particles entering or leaving Ω (depending on the sign of G) from $y \notin \Omega$ to $x \in \Omega$. Using the support of ϕ and (3.3), we introduce

$$h(x,t) = \int_{\mathbb{R}^n\setminus\Omega} G(y,t)\phi(x-y)\,\mathrm{d}y, \qquad x \in \Omega, \tag{3.9}$$

and note that $h(x,t)$ is only nonzero for $x \in \Omega\backslash\widetilde{\Omega}$. Thus, the inhomogeneous nonlocal Neumann boundary value problem takes the form

$$\begin{cases} u_t(x,t) = \displaystyle\int_\Omega (u(y,t) - u(x,t))\phi(x-y)\,\mathrm{d}y + h(x,t), & x \in \Omega, \\ u(x,0) = u_0(x), & x \in \Omega. \end{cases} \tag{3.10}$$

As expected, the integrated density of particles over Ω changes with time, unlike (3.8), as seen with

$$\int_\Omega u(x,t)\,\mathrm{d}x = \int_\Omega u_0(x)\,\mathrm{d}x + \int_0^t \int_\Omega h(x,s)\,\mathrm{d}x\,\mathrm{d}s. \tag{3.11}$$

Differentiating (3.11) with respect to t,

$$\int_\Omega u_t(x,t)\,\mathrm{d}x = \int_\Omega h(x,t)\,\mathrm{d}x. \tag{3.12}$$

Suppose that $G(y,t) = G(y)$, i.e., $h(x,t) = h(x)$. Then, from (3.12), a necessary condition for stationary solutions of (3.10) to exist is the compatibility condition

$$\int_\Omega h(x)\,\mathrm{d}x = \int_\Omega \int_{\mathbb{R}^n\setminus\Omega} G(y)\phi(x-y)\,\mathrm{d}y\,\mathrm{d}x = 0. \tag{3.13}$$

As shown in [9], (3.13) is also a sufficient condition.

4. A Nonlocal Calculus and Variational Analysis.

4.1. A Nonlocal Calculus. Let $V(\Omega)$ denote the Hilbert space of test functions defined in Ω and $U(\Omega)$ be that of trial functions. Suppose that f is an antisymmetric mapping from $\Omega\times\Omega \to \mathbb{R}$ and that α is a symmetric mapping from $\Omega\times\Omega\to\mathbb{R}$. We define $\mathcal{D}(f) : \widetilde{\Omega} \to \mathbb{R}$ via

$$\mathcal{D}(f)(x) = 2\int_\Omega f(x,y)\alpha(x,y)\,\mathrm{d}y, \quad x \in \widetilde{\Omega}. \tag{4.1}$$

Similarly, $\mathcal{N}(f) : (\Omega\setminus\widetilde{\Omega}) \to \mathbb{R}$ by

$$\mathcal{N}(f)(x) = -2\int_\Omega f(x,y)\alpha(x,y)\,\mathrm{d}y, \quad x \in \Omega\setminus\widetilde{\Omega}. \tag{4.2}$$

Define the operator $\mathcal{G}$ acting on $v \in V(\Omega)$ by

$$\mathcal{G}(v) = (v(y) - v(x))\alpha(x,y), \quad x,y \in \Omega. \tag{4.3}$$

Following [16], we formulate the variational problem of finding $u \in U(\Omega)$ such that

$$\int_\Omega\int_\Omega \mathcal{G}(v)\mathcal{K}\,\mathrm{d}y\,\mathrm{d}x = \int_{\Omega\setminus\widetilde{\Omega}} v(x)h(x)\,\mathrm{d}x, \quad \forall v \in V(\Omega), \tag{4.4}$$

where $\mathcal{K}$ is a antisymmetric operator on $U(\Omega)\times U(\Omega)\times\Omega\times\Omega$ and $h : (\Omega\setminus\widetilde{\Omega}) \to \mathbb{R}$.

We focus our attention on the case when $\mathcal{K}(u(x),u(y),x,y) = \beta(x,y)\mathcal{G}(u)$, where $\beta : \Omega\times\Omega\to\mathbb{R}$ is a symmetric function. Further, we assume both α and β are radial functions, the latter having support in $B(0,d)$. Omitting details, found in [16], the nonlocal Green's first and second identities, respectively, are

$$\int_{\widetilde{\Omega}} v(x)\mathcal{D}(\beta\mathcal{G}(u))\,\mathrm{d}x + \int_\Omega\int_\Omega \beta\mathcal{G}(v)\mathcal{G}(u)\,\mathrm{d}y\,\mathrm{d}x = \int_{\Omega\setminus\widetilde{\Omega}} v(x)\mathcal{N}(\beta\mathcal{G}(u))\,\mathrm{d}x \tag{4.5}$$

and

$$\begin{aligned}\int_{\widetilde{\Omega}} v(x)\mathcal{D}(\beta\mathcal{G}(u))\,\mathrm{d}x - \int_{\widetilde{\Omega}} u(x)\mathcal{D}(\beta\mathcal{G}(v))\,\mathrm{d}x \\ = \int_{\Omega\setminus\widetilde{\Omega}} \Big(v(x)\mathcal{N}(\beta\mathcal{G}(u)) - u(x)\mathcal{N}(\beta\mathcal{G}(v)) \Big)\,\mathrm{d}x.\end{aligned} \tag{4.6}$$

Assume that the function h satisfies the compatibility condition

$$\int_{\Omega\setminus\widetilde{\Omega}} h(x)\,\mathrm{d}x = 0, \tag{4.7}$$

which corresponds to (3.13) if we trivially extend h to be zero in $\widetilde{\Omega}$. The variational formula then reads: find $u \in U(\Omega)$ such that

$$\int_\Omega \int_\Omega \beta \mathcal{G}(v)\mathcal{G}(u)\,\mathrm{d}y\,\mathrm{d}x = \int_{\Omega\backslash\widetilde{\Omega}} v(x)h(x)\,\mathrm{d}x, \quad \forall v \in V(\Omega).$$

Application of the nonlocal Green's first identity yields

$$-\int_{\widetilde{\Omega}} v(x)\mathcal{D}(\beta\mathcal{G}(u))\,\mathrm{d}x + \int_{\Omega\backslash\widetilde{\Omega}} v(x)\mathcal{N}(\beta\mathcal{G}(u))\,\mathrm{d}x = \int_{\Omega\backslash\widetilde{\Omega}} v(x)h(x)\,\mathrm{d}x. \tag{4.8}$$

Choosing appropriate test functions, (4.8) is the weak form of

$$\begin{cases} -\mathcal{D}(\beta\mathcal{G}(u)) = 0, & x \in \widetilde{\Omega}, \\ \mathcal{N}(\beta\mathcal{G}(u)) = h(x), & x \in \Omega\backslash\widetilde{\Omega}. \end{cases} \tag{4.9}$$

Denoting $\phi = 2\alpha^2\beta$, we quickly see that (4.9) is equivalent to the steady-state form of (3.10). Similarly, imposing a nonlocal Dirichlet boundary condition, we have

$$\begin{cases} -\mathcal{D}(\beta\mathcal{G}(u)) = 0, & x \in \widetilde{\Omega}, \\ u = 0, & x \in \Omega\backslash\widetilde{\Omega}, \end{cases} \tag{4.10}$$

which is equivalent to the steady-state form of (3.4). Thus, we have established a link between the existence, uniqueness, and convergence results in [1, 7, 9, 10] and the emerging nonlocal calculus theory in [16].

4.2. Convergence to Classical Diffusion Equation. For simplicity, consider the one-dimensional case, i.e., $\Omega \subseteq \mathbb{R}$. Assume that α and β are radial functions and suppose, for a given ε, that β satisfies

$$\beta(x-y) = 0, \quad \text{for } |x-y| \geq \varepsilon.$$

We use the notation β_ε in place of β to denote this dependence. For $x \in \Omega$, define $\Omega_\varepsilon(x) = B(x,\varepsilon) \cap \Omega$. We tak $U(\Omega) = V(\Omega)$ and assume $u, v \in U(\Omega)$ are smooth so that, for instance,

$$\mathcal{G}(v) = (v(y) - v(x))\alpha(x-y) = \left((y-x)\frac{\mathrm{d}v}{\mathrm{d}x} + O(\varepsilon^2)\right)\alpha(x-y),$$

for all $y \in \Omega_\varepsilon(x)$. Thus, defining $\phi_\varepsilon = 2\alpha^2\beta_\varepsilon$,

$$\int_\Omega \int_\Omega \beta_\varepsilon \mathcal{G}(v)\mathcal{G}(u)\,\mathrm{d}y\,\mathrm{d}x \approx \int_\Omega \frac{\mathrm{d}v}{\mathrm{d}x}\frac{\mathrm{d}u}{\mathrm{d}x} D_\varepsilon(x)\,\mathrm{d}x,$$

where

$$D_\varepsilon(x) = \frac{1}{2}\int_{\Omega_\varepsilon(x)} (y-x)^2\phi_\varepsilon(x-y)\,\mathrm{d}y.$$

Notice, for $x \in \widetilde{\Omega}$, $D_\varepsilon(x)$ is a constant, namely

$$D_\varepsilon(x) = \frac{1}{2}\int_{B(x,\varepsilon)} (y-x)^2\phi_\varepsilon(x-y)\,\mathrm{d}y = \frac{1}{2}\int_{-\varepsilon}^{\varepsilon} s^2\phi_\varepsilon(s)\,\mathrm{d}s.$$

To ensure

$$D(x) = \lim_{\varepsilon \to 0} D_\varepsilon(x)$$

is not the zero function, assume

$$\phi_\varepsilon(s) = \frac{d^3}{\varepsilon^3}\phi\left(\frac{ds}{\varepsilon}\right).$$

Thus, for $x \in \widetilde{\Omega}$,

$$D_\varepsilon(x) = \frac{1}{2}\int_{-\varepsilon}^{\varepsilon} s^2\phi_\varepsilon(s)\,\mathrm{d}s = \frac{1}{2}\int_{-d}^{d} s^2\phi(s)\,\mathrm{d}s.$$

Notice, for $x \in \widetilde{\Omega}$, $D_\varepsilon(x)$ is the usual diffusion coefficient in (2.6). As $\varepsilon \to 0$, $\widetilde{\Omega} \to \Omega$ and we see that $D(x)$ is constant on all of Ω.

Further analysis [16], in higher dimensions, demonstrates

$$\begin{cases} -\mathcal{D}(\beta_\varepsilon \mathcal{G}(u)) = 0, & x \in \widetilde{\Omega}, \\ u = 0, & x \in \Omega\backslash\widetilde{\Omega}, \end{cases} \quad \to \quad \begin{cases} D\Delta u = 0, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases} \tag{4.11}$$

and, similarly,

$$\begin{cases} -\mathcal{D}(\beta_\varepsilon \mathcal{G}(u)) = 0, & x \in \widetilde{\Omega}, \\ \mathcal{N}(\beta_\varepsilon \mathcal{G}(u)) = h, & x \in \Omega\backslash\widetilde{\Omega}, \end{cases} \quad \to \quad \begin{cases} D\Delta u = 0, & x \in \Omega, \\ (D\nabla u)\cdot\eta = h, & x \in \partial\Omega, \end{cases} \tag{4.12}$$

as $\varepsilon \to 0$. These convergence results agree with similar results in [10], in which an ε-family nonlocal diffusion problems, such as (3.4) or (3.10), is studied. Heuristically, the solutions of this family of problems converges to the solution of the classical diffusion problem (1.2) as $\varepsilon \to 0$. With convergence results such as these in hand, we can study solutions of nonlocal diffusion equations and compare them to solutions of their *associated* classical diffusion equations.

4.3. Variational Analysis of Diffusion Equations. Let $U(\Omega) = V(\Omega)$ and define $V_0(\Omega) = \{v \in V(\Omega) \,|\, v = 0 \text{ on } \Omega\backslash\widetilde{\Omega}\}$. Then, the variational problem corresponding to (4.10) reads: find $u \in U(\Omega)$ such that $u = 0$ on $\Omega\backslash\widetilde{\Omega}$ and

$$\int_\Omega \int_\Omega \beta\mathcal{G}(v)\mathcal{G}(u)\,\mathrm{d}y\,\mathrm{d}x = 0, \quad \forall v \in V_0(\Omega). \tag{4.13}$$

Similarly, the variational problem corresponding to (4.9), with (4.7), reads: find $u \in U(\Omega)$ such that

$$\int_\Omega \int_\Omega \beta\mathcal{G}(v)\mathcal{G}(u)\,\mathrm{d}y\,\mathrm{d}x = \int_\Omega v(x)h(x)\,\mathrm{d}x, \quad \forall v \in V(\Omega). \tag{4.14}$$

Discretizing Ω and choosing appropriate test/trial spaces, e.g., discontinuous piecewise linear functions on Ω, aided by the nonlocal Green's identities, a Galerkin finite element approach can be used to approximate solutions to (4.13) and (4.14).

Simultaneously, classical finite element theory, with continuous piecewise linear functions for example, can be used on the classical diffusion equation. The convergence results of (4.11) and (4.12) justify comparing the resulting finite element approximations of the nonlocal and the classical diffusion equations. Preliminary results, not presented here, suggest that,

for small ε, the two approximate solutions are reasonably close, as theory suggests. However, as ε increases, the nonlocal effects present in the nonlocal diffusion equation produce significant differences in the solutions compared to solutions of the classical diffusion equation. This, for instance, confirms observations in [2, 3, 4], that the diffusion profiles of fluids with nonzero mean-free-path (the nonlocal diffusion equation) differ significantly from those of fluids with zero mean-free-path (the classical diffusion equation).

5. Conclusions and Future Research Direction. We have presented the classical diffusion equation as both a special case of the nonlocal diffusion equation and as a second-order truncation of a Kramers-Moyal expansion of the nonlocal diffusion equation. Through which, we identified limitations of using (1.2) in place of (1.1), which involved discussions on random walks, stochastic differential equations, Itō processes and Lévy jump processes, anomalous diffusion, Fokker-Planck equations and master equations, and fractional diffusion equations. Selected theory of nonlocal diffusion equations and a variational approach were presented, both necessary for formulating finite element approximations of solutions to nonlocal diffusion equations.

Continuing this work into the following year, the authors will investigate the effects of a finite element method discretization on nonlocal diffusion equations. Resulting finite element approximations will be compared to those of classical diffusion equations and deviations between the two will be explored by investigating the discarded statistical moments in the Kramers-Moyal expansion.

REFERENCES

[1] F. Andreu, J. Mazon, J. Rossi, and J. Toledo, *A nonlocal p-Laplacian evolution equation with Neumann boundary conditions*, Journal de mathématiques pures et appliquées, (2008).

[2] G. Aranovich and M. Donohue, *Eliminating the mean-free-path inconsistency in classical phenomenological model of diffusion for fluids*, Physica A: Statistical Mechanics and its Applications, 373 (2007), pp. 119–141.

[3] ———, *Limitations and generalizations of the classical phenomenological model for diffusion in fluids*, Molecular Physics, 105 (2007), pp. 1085–1093.

[4] ———, *Diffusion in fluids with large mean free paths: Non-classical behavior between Knudsen and Fickian limits*, Physica A: Statistical Mechanics and its Applications, (2009).

[5] O. Bakunin, *Turbulence and diffusion: scaling versus equations*, Springer Verlag, 2008.

[6] C. Carrillo and P. Fife, *Spatial effects in discrete generation population models*, Journal of Mathematical Biology, 50 (2005), pp. 161–188.

[7] E. Chasseigne, M. Chaves, and J. Rossi, *Asymptotic behavior for nonlocal diffusion equations*, Journal de mathématiques pures et appliquées, 86 (2006), pp. 271–291.

[8] A. Chorin and O. Hald, *Stochastic tools in mathematics and science*, Springer Verlag, 2006.

[9] C. Cortazar, M. Elgueta, J. Rossi, and N. Wolanski, *Boundary fluxes for nonlocal diffusion*, Journal of Differential Equations, 234 (2007), pp. 360–390.

[10] ———, *How to approximate the heat equation with Neumann boundary conditions by nonlocal diffusion problems*, Archive for Rational Mechanics and Analysis, 187 (2008), pp. 137–156.

[11] A. Einstein, *Investigations on the Theory of the Brownian Movement*, Dover Publications, 1956.

[12] P. Fife, *Some nonclassical trends in parabolic and parabolic-like evolutions*, Trends in nonlinear analysis, (2003), pp. 153–191.

[13] G. Folland, *Real analysis: modern techniques and their applications*, Wiley, 1999.

[14] C. Gardiner, *Handbook of stochastic methods (for physics, chemistry and the natural sciences)*, Springer series in synergetics, (2004).

[15] G. Gilboa and S. Osher, *Nonlocal operators with applications to image processing*, UCLA CAM Report, (2007), pp. 07–23.

[16] M. Gunzburger and R. Lehoucq, *A Nonlocal Vector Calculus with Applications to Nonlocal Boundary Value Problems*, Submitted for publication in Multiscale Modeling and Simulation, (2009).

[17] P. Hänggi, *Langevin description of Markovian integro-differential master equations*, Zeitschrift für Physik B Condensed Matter, 36 (1980), pp. 271–282.

[18] B. Hughes, M. Shlesinger, and E. Montroll, *Random walks with self-similar clusters*, Proceedings of the National Academy of Sciences, 78 (1981), pp. 3287–3291.

[19] S. Jespersen, R. Metzler, and H. Fogedby, *Levy flights in external force fields: Langevin and fractional Fokker-Planck equations and their solutions*, Physical Review - Series E, 59 (1999), pp. 2736–2745.
[20] J. Klafter, A. Blumen, and M. Shlesinger, *Stochastic pathway to anomalous diffusion*, Physical Review A, 35 (1987), pp. 3081–3085.
[21] J. Klafter, M. Shlesinger, and G. Zumofen, *Beyond brownian motion*, Physics Today, 49 (1996), pp. 33–39.
[22] G. Lawler, *Introduction to stochastic processes*, Chapman & Hall/CRC, 2006.
[23] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Physics Reports, 339 (2001), p. 1.
[24] B. Øksendal, *Stochastic differential equations: an introduction with applications*, Springer, 2003.
[25] D. Schertzer, M. Larcheveque, J. Duan, V. Yanovsky, and S. Lovejoy, *Fractional Fokker–Planck equation for nonlinear stochastic differential equations driven by non-Gaussian Lévy stable noises*, Journal of Mathematical Physics, 42 (2001), p. 200.
[26] P. Seleson, M. Parks, M. Gunzburger, and R. Lehoucq, *Peridynamics as an Upscaling of Molecular Dynamics*, Accepted for publication in Multiscale Modeling and Simulation, (2009).
[27] J. Sethna, *Statistical mechanics: entropy, order parameters and complexity*, Oxford University Press, USA, 2006.
[28] M. Ullah and O. Wolkenhauer, *Family tree of Markov models in systems biology*, Systems Biology, IET, 1 (2007), pp. 247–254.
[29] N. Van Kampen, *Stochastic processes in physics and chemistry*, North-Holland, 2007.
[30] L. Zhang, *Existence, uniqueness and exponential stability of traveling wave solutions of some integral differential equations arising from neuronal networks*, Journal of Differential Equations, 197 (2004), pp. 162–196.

A TRILINOS AND HYPRE INTERFACE

KELLY J. FERMOYLE[§] AND MICHAEL A. HEROUX[¶]

Abstract. Trilinos and hypre are mathematical software libraries that solve large-scale scientific and engineering problems. This work created an interface between the libraries to give Trilinos the full functionality of the hypre preconditioners. Several different interfaces were created to cater to all the needs of Trilinos users. The interfaces were designed to be as efficient as possible to minimize computational needs.

1. Introduction. Trilinos is a software library that uses an object-oriented framework to solve large-scale, complex multi-physics engineering, and scientific problems [6]. hypre is a separately developed set of preconditioners for solving large-scale structured and unstructured grid problems [1]. While Trilinos has always had many of the capabilities provided by hypre, there were some preconditioners not available to Trilinos [5]. This paper describes the project of interfacing Trilinos with hypre to give it the full functionality of the hypre linear-algebraic system interface (IJ) library. In addition the interface gives users access to hypre's solvers without the need to learn a new library.

2. Hypre-Epetra Interface. Trilinos is composed of many packages (originally three, but its capabilities have greatly expanded), each with different functionality [4]. Epetra is the foundation for all of Trilinos and acts as a translator between the packages. Almost all of the packages expect object types that are defined in Epetra. One of the most important classes in Epetra is the abstract class *Epetra_RowMatrix*, which defines the methods that distributed (or serial) sparse matrices have. This base class is used as the matrix parameter for many of Trilinos's computational methods.

The first step of the interface is to allow preexisting hypre matrices to use the functionality of Trilinos solvers. This means that Trilinos will need to interpret the hypre matrix as if it were an *Epetra_RowMatrix*. The intended audience is users of hypre that would like to take advantage of the resources provided by Trilinos. There are several different ways to accomplish this. The simplest can be done with a small programming effort by simply copying row-by-row from the hypre matrix into a new matrix in Trilinos. However, this is inefficient both computationally and for the limited memory system. We instead wrap the hypre matrix in Epetra and call hypre's internal methods.

Fortunately, Trilinos gives the developer a simple way to implement the methods of *Epetra_RowMatrix*, which is called *Epetra_BasicRowMatrix*. Trilinos is primarily written in C++ giving it an object-oriented paradigm, which the developers have taken full advantage of. *Epetra_HypreIJMatrix*, the class used as our interface, inherits all of *Epetra_BasicRowMatrix*'s implementation and is still an *Epetra_RowMatrix*. The *BasicRowMatrix* class is an adapter that only requires implementation for a minimum of four methods, constructor and destructor. It can use these methods to define the other 35 methods of *Epetra_RowMatrix*. In addition to the minimum implementation, the hypre interface implements the Multiply() and Solve() methods to make these computations more efficient by calling hypre's methods. The *BasicRowMatrix* implementation defines the Multiply() but not Solve(). Multiply() uses a copy of each row to do its computation and is not very efficient, so it was overwritten to avoid this. Solve() just returns an error code, so the implementation overrides this and can use any of hypre's solvers and preconditioners.

The following methods were implemented in the hypre-Epetra interface:

[§]The Computer Science and Engineering Department of the Pennsylvania State University, kjf198@cse.psu.edu
[¶]Sandia National Laboratories, maherou@sandia.gov

```
//Using HYPRE_IJMatrix A, and appropriate Vectors RHS, LHS
EpetraExt_HypreIJMatrix Matrix(A);
//HYPRE_Solver solver;
Matrix.SetParameter(Solver, BoomerAMG);
//HYPRE_BoomerAMGCreate(&solver);
//HYPRE_BoomerAMGSetup(solver, A, RHS, LHS);
Matrix.SetParameter(Solver, &HYPRE_BoomerAMGSetTol, 1E-9);
//HYPRE_BoomerAMGSetTol(solver, 1E-9);
Matrix.SetParameter(Solver, &HYPRE_BoomerAMGSetMaxIter, 1000);
//HYPRE_BoomerAMGSetMaxIter(solver, 1000);
Matrix.Solve(RHS, LHS);
//HYPRE_BoomerAMGSolve(solver, A, RHS, LHS);
//HYPRE_BoomerAMGDestroy(solver);
```

FIG. 2.1. *A comparison between the hypre interface and Trilinos interface. Both of these codes would accomplish the same solve. Note that in the Trilinos code, Solver is an enumerated type and not a variable.*

1. Constructor: Create the object as well as *Epetra_Map* objects and set defaults for the Solve() operation.
2. Destructor: Delete any created objects, but leave the underlying hypre matrix untouched so its use can continue.
3. NumMyRowEntries(): Determine the number of entries in a given row.
4. ExtractMyRowCopy(): Extract a copy of a specified row of the matrix.
5. LeftScale(): Scale the matrix by a vector on the left.
6. RightScale(): Scale the matrix by a vector on the right.
7. Multiply(): Multiply the matrix with a multivector (called by Apply()).
8. Solve(): Solve a linear system, taking a multivector (called by ApplyInverse()).
9. SetParameter(): Set parameter for the Solve method. Sets both the solver and preconditioner options.

The SetParameter() method gives the user access to any option of the hypre solver library. The user can choose to either use the solver or apply the preconditioner. Many of the SetParameter() calls will take a function pointer to the underlying method in hypre, to allow the user to set whatever options necessary. This was needed because hypre is a third-party library (TPL) developed independently of Trilinos and the methods may change. To avoid having to change our methods in the interface, we instead rely on the user to pass the functions. Figure 2.1 shows an example of the way the interface could be used to solve a system. The code segment gives a comparison of the interface in Epetra versus hypre.

From figure 2.1, we see the steps taken to create a solver and use it to solve a linear system. Note that the first step is to create the *EpetraExt_HypreIJMatrix* object. This takes care of all the background details of creating and setting up the solver. It also chooses the default parameters if we do not want to set our own. The next command sets this solver to use the BoomerAMG solver (an enumerated solver type). The next two lines set specific parameters for the solver by passing the function pointer and the one parameter needed. To see the full range of hypre parameters use [2]. Finally the Solve() command is called, passing the *Epetra_MultiVectors*. We do not need to worry about destroying the solver object because the class does that automatically when it is no longer used.

In addition to creating a wrapper for a hypre matrix, functionality has also been added to read the disk output from a parallel hypre matrix and create an *Epetra_CrsMatrix* with identical distribution. This conversion allows a Trilinos user to use more than just the methods provided by *Epetra_RowMatrix*, but also the specific methods of *Epetra_CrsMatrix*. There are a number of solvers and other packages in Trilinos that require this kind of matrix.

This method automatically determines the processor distribution given the file name. This, of course, requires the user to use the same number of processors as when the matrix was printed in hypre. Given the correct *Epetra_Comm* object, it will find the files and create the matrix, filling it row-by-row from the disk output. This method is also part of the EpetraExt package in the class *EpetraExt_CrsMatrixIn*. This class contains many different methods for translating from a matrix type to another from disk output. No object needs to be created, so the call is very simple:

```
1. Epetra_CrsMatrix* A;
2. EpetraExt::HypreFileToCrsMatrix("FileName", comm, A);
```

This code assumes that the *Epetra_Comm* object named comm has already been instantiated. The first line just creates the pointer to an *Epetra_CrsMatrix* A. The next line is the call to the new method; the first parameter takes the name of the file on the disk, the second is the *Epetra_Comm* object so it knows the processor layout, and finally the pointer to the new matrix. After this call, the pointer will be filled with the data from the disk. After these two simple lines of code, the *Epetra_CrsMatrix* A can be used just the same as if it were created like any other matrix. This type of matrix is used in most of the Trilinos package.

3. Epetra-Hypre Interface. Another scenario is that the user has an Epetra matrix and they would like to use the functionality of hypre without having to learn the intricacies of another library. This is the more likely situation among researchers at Sandia because so many are familiar with the Trilinos package and use it in their research. Ifpack is a package in Trilinos for preconditioned iterative solvers. We define a new class in Ifpack called *Ifpack_Hypre* that will be our interface. This interface will follow the same standards as any other class in the Ifpack package. An *Ifpack_Hypre* object can be created using the factory class just like any other Ifpack object. After construction, the standard order of function calls is: SetParameters(), Initialize(), Compute(), and ApplyInverse().

By carefully using the internal hypre methods, this class will use the same order. The SetParameters() method takes the standard *Teuchos::ParameterList*, but the list is abnormal because it will be calling a third-party library. As in the last section, we do not want this class to require modification every time hypre changes its methods. So, the SetParameters() is again forced to take function pointers from the user. This is done by creating an object that stores the function pointer and its parameters until it is called later. To avoid the use of a *Teuchos::ParameterList*, users can call SetParameter() and set the options one at a time.

Unfortunately, hypre uses a very different storage system for matrices than *Epetra_CrsMatrix* (the most commonly used matrix in Epetra), so the data must be copied into a new hypre matrix row-by-row. hypre stores two different compressed sparse row (CSR, or CRS for compressed row storage in Trilinos) matrices on each processor, one for the diagonal part – the part used by the local processor most often, and an off-diagonal part – used by other processors more often. A visual reference of this can be seen in Figure 3.1. The example shows a three processor distribution of a matrix. We see that each processor stores the two separate CSR matrices. More on this can be found in [3]. *Epetra_CrsMatrix*, on the other hand, only uses a single matrix locally to describe all of its elements. Thus, there is no way to avoid copying the matrix data if we need to use the hypre matrix struct. The creation and copying of this data is done in the Initialize() call.

The solver is finally setup using the created hypre matrix in the Compute() method. This setup will use all the parameters set earlier or a set of default parameters if none were given. This is usually the most computationally intensive method, but all of the work is done in hypre's methods. After the preconditioner has been computed, it can be used to solve a linear system. A system is solved using ApplyInverse() which takes a right-hand-side (RHS)

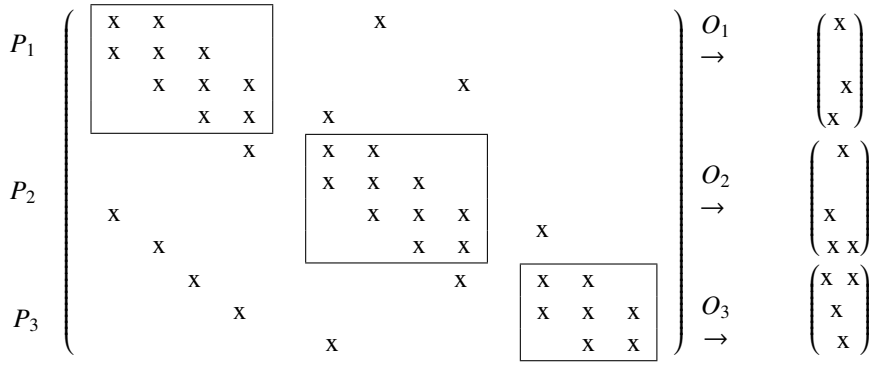


FIG. 3.1. *An example of a CSR matrix in hypre, distributed across three processors. The local matrices are shown in boxes. The rest of a processor's data is compressed into O_1, O_2, and O_3.*

```
//Using HYPRE_RowMatrix A, and appropriate Vectors RHS, LHS
Ifpack factory;
Ifpack_Preconditioner prec = factory.Create("Hypre", A);
RCP<FunctionParameter> parameters[2];
parameters[0] = rcp(new FunctionParameter(Solver, &HYPRE_BoomerAMGSetMaxIter, 1000));
parameters[1] = rcp(new FunctionParameter(Solver, &HYPRE_BoomerAMGSetTol, 1E-9));
Teuchos::ParameterList list("Hypre settings");
list.set("NumFunctions", 2);
list.set<RCP<FunctionParameter>*>("Functions", parameters);
list.set("Solver", BoomerAMG);
prec->SetParameters(list);
prec->Initialize();
prec->Compute();
prec->ApplyInverse(RHS, LHS);
delete prec;
```

FIG. 3.2. *An example of a session using the Ifpack_Hypre class. This will solve a linear system.*

multivector and a left-hand-side (LHS) multivector. The solution will be placed in the LHS. The Trilinos interface avoids copying the vector data by giving hypre a view of the data pointers.

Figure 3.2 shows a common sequence that may be used in *Ifpack_Hypre* to create the object and proceed to set specific parameters before solving the system. The first line creates an Ifpack object which is a factory for creating *Ifpack_Preconditioner* objects. The factory creates our object using a string describing what kind of preconditioner it is and the *Epetra_RowMatrix* it uses. Next we want to set the parameters that the solver will reference. Ifpack objects always take a *Teuchos::ParameterList* in the call to SetParameters() method. This class is different because one of the *ParameterEntries* is an object of type *FunctionParameter*. This object is created using a function pointer and the parameters to that function. The *ParameterList* also needs to know how many *FunctionParameter* objects were created.

After the parameters are set, we call Initialize(), which behind the scenes creates the hypre matrix and copies the rows over. The call to Compute() sets up the solver using the parameters (the parameters are required to be passed before Compute()). This call to Compute() is the most computationally intensive. Next, ApplyInverse() solves the system with the two *Epetra_MultiVector* objects.

Description	String	Default
Factorization level	SetLevel	(int) 1
Block Jacobi ILU	SetBJ	(int) 0
Print stats	SetStats	(int) 0
Print memory	SetMem	(int) 0
Drop tolerance	SetSparse	(double) 0.0
Scale values	SetRowScale	(int) 0
Use ILUT	SetILUT	(double) 0.0

FIG. 4.1. *A list of the parameters that can be set in Ifpack_Euclid. Should be used in the SetParameters() method with each being a string and parameter pair.*

4. Epetra-Euclid Interface. The interface described in section 3 is effective, but it has a critical inefficiency, and when memory is limited and using a large matrix, it might be prohibitive. The problem is that the entire matrix needs to be copied from the Trilinos platform into something recognizable to hypre. This requires twice the memory capabilities as if we did not have to make this copy. While we cannot solve this problem in general, there is a preconditioner that we can use without making the copy. Euclid is an important preconditioner in hypre that performs a parallel incomplete lower-upper triangular (ILU) decomposition. Euclid is part of the hypre library, but was designed before hypre and it uses a different matrix extraction scheme. This gives Trilinos the ability to interact with Euclid by just passing a view of each row, and not a hypre matrix.

Using the Euclid preconditioner with a slightly modified matrix interface required changing it so that it called the correct method of *Epetra_CrsMatrix*. The *CrsMatrix* needs to be used instead of *RowMatrix* because we need to get a view of the row, not a copy. This is only a method of the child class *CrsMatrix*. To provide this change to Trilinos, the Euclid files were added to the Trilinos library as part of a new class called *Ifpack_Euclid*. This class has a much simpler interface than *Ifpack_Hypre* because we are no longer using a TPL. Euclid is now part of the Trilinos library, so it will not change by itself.

The Euclid files are in a subdirectory of Ifpack called Euclid and are still very similar to the hypre library except the file *getRow*. *getRow* is used by hypre as the interface to get one row at a time from the Matrix. Instead of using the hypre call, it uses the method from *Epetra_CrsMatrix*. Since hypre is written in C, we need an intermediary method in a file *callEpetra*. This file just passes the call in a way recognizable to both programs. This is one of the few changes to Euclid to allow it to be used in *Ifpack_Euclid*. It is used just like other Ifpack classes except that it needs to take an *Epetra_CrsMatrix* instead of an *Epetra_RowMatrix*. The SetParameters() method is simple like most Ifpack classes and does not need to take function pointers. Similar to *Ifpack_Hypre*, SetParameter() is implemented to set options one at a time. The ApplyInverse() method takes *Epetra_MultiVectors* for the right-hand side and left-hand side of the linear system. It loops over the vectors making calls to the Euclid solver.

All of the creation and setup of the Euclid solver is done automatically. Usage of the class requires no prior knowledge of hypre and can be used with just Trilinos documentation. The class can compute a condition estimate of the preconditioner from the *Ifpack_Condest* class that is a very helpful part of the Ifpack package. The class also implements the Print() method, which helps the user to see details of how it has been used.

5. Conclusions. Trilinos and hypre are both very matured mathematical software libraries. This project worked to interface both of these libraries and make them easier to work with. We created a hypre-Trilinos interface that allows users with a hypre matrix to use the

full workings of most packages of Trilinos. This interface required no copying of data from the initial matrix to the new one, it instead wraps the matrix and translates any method on *Epetra_RowMatrix* to its equivalent in hypre. Capability was also added to create an *Epetra_CrsMatrix* using the disk output from a hypre call to print to a file.

The next interface that was created was for existing Trilinos users that just wanted to use the hypre solvers. This class allows these users to either use default settings for the solver, or set their own parameters. The parameters they pass may be a little complicated for novice users, but the documentation makes it simpler. The class requires the matrix to be copied because of the inherent difference in the way the matrices are stored. The time to copy data isn't nearly as important as the increased memory needs.

The problem of data copying can be solved if the user only wants to apply the Euclid preconditioner. This independently developed preconditioner can access rows of an *Epetra_CrsMatrix* without the need for a hypre matrix. This makes it much more manageable for large distributed systems that are running at the full capacity of the memory system.

6. Acknowledgements. We would like to thank Chris Baker, Todd Coffey, Jonathan Hu, and Chris Siefert of Sandia National Laboratories for their support in this project. We also thank Allison Baker, David Hysom and the rest of the hypre Support team for their help with the internal methods in hypre.

REFERENCES

[1] R. D. Falgout, A. H. Baker, V. E. Henson, U. M. Yang, and B. Lee, *hypre User's Manual*, Tech. Rep. 2.4.0b, Lawrence Livermore National Laboratory, 2008.

[2] R. D. Falgout, A. H. Baker, and B. L. Van E. Henson, Ulrike M. Yang, *hypre Reference Manual*, Tech. Rep. 2.4.0b, Lawrence Livermore National Laboratory, 2008.

[3] R. D. Falgout, J. E. Jones, and U. M. Yang, *Pursuing Scalability for hypre's Conceptual Interfaces*, ACM Transactions on Mathematical Software, (2004), pp. 3–10. Lawrence Livermore National Laboratory.

[4] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, *An Overview of Trilinos*, Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.

[5] M. A. Heroux, J. M. Willenbring, and R. Heaphy, *Trilinos Developers Guide*, Tech. Rep. SAND2003-1898, Sandia National Laboratories, 2003.

[6] M. Sala, M. A. Heroux, and D. M. Day, *Trilinos Tutorial*, Tech. Rep. SAND2004-2189, Sandia National Laboratories, 2004.

AUTOMATIC HEXAHEDRAL MESH GENERATION WITH A REFINED CARTESIAN GRID DATA STRUCTURE

JENNA M. KALLAHER* AND STEVEN J. OWEN†

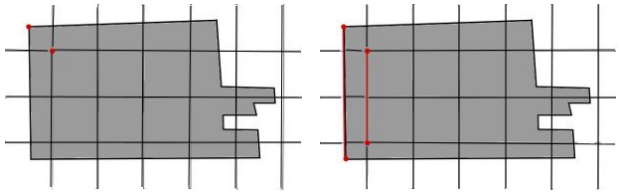
Abstract. The embedding algorithm for hexahedral mesh generation [2] has the potential to become a robust mesh generation algorithm; however, it still lacks the ability to handle many small features in geometry models. Smart automatic mesh refinement of the Cartesian grid could provide further resolution to capture geometric features in an arbitrary geometry model. This work deals with the construction of a smart refinement approach and a light-weight data structure to hold the resulting enriched grid information.

1. Introduction. Several algorithms for hexahedral mesh generation have been created to mesh specific geometries, but automatic hexahedral mesh generation remains an area of continuing research. Reasonable quality tetrahedral meshes may be generated with very little geometry decomposition for most geometry models; however, many analyses require hexahedral elements or would generate more accurate results with a hexahedral mesh. A robust algorithm for automatic hexahedral mesh generation would decrease the total time needed to perform an analysis and also increase the accuracy of the results. Recently, an approach was construced for the automatic hexahedral meshing of arbitrary geometries [5] by using the concept of a fundamental mesh and viewing mesh generation as an optimization problem. This work extended the class of geometries which can be meshed with hexahedral elements and implemented an algorithm to mesh geometries composed of a single surface. The algorithm was later extended to begin considering geometries with multiple surfaces, curves and vertices. This work became the embedding mesh generation algorithm [2], which is a promising approach to generating hexahedral meshes for arbitrary geometries. The hexahedral mesh generation process used by the embedding algorithm [2, 5] will be the basis of this work because of its initial use of a Cartesian grid. Many calculations can be performed more quickly with a Cartesian grid whereas other mesh generation algorithms have used a full octree lattice as a base [6, 1, 4]. Also, the memory requirements are smaller when using a Cartesian Grid than when using an octree lattice [4]. Smaller memory requirements improve the ability to scale this algorithm to create larger meshes for large geometry models or models where a fine mesh size is required. Enriching a Cartesian grid only when necessary can provide the benefit of quick calculations for most elements and also the enriched mesh necessary to capture all features in a geometry model.

2. Previous Work on the Embedding Algorithm. The embedding mesh generation algorithm [2] attempts to generate a hexahedral mesh by embedding geometry features in a Cartesian grid. Each vertex is represented by a corresponding node, curves are represented by a series of edges, surfaces are assigned faces, and volumes are assigned hexes. The association of edges with curves and nodes with vertices can be seen in Figure 2.1.

Following the embedding process, it is necessary to project the nodes, edges, and hexes to the corresponding geometry. A problem arises when small features are encountered in the model. The previously proposed embedding algorithm [2] fails when it encounters too many vertices in one area of the model. The standard size of a Cartesian grid cell may not always be able to capture all of the geometry features contained within it. In Figure 2.2, there are not enough nodes to capture each vertex in the Cartesian grid. In that case, it becomes necessary to refine the Cartesian grid cells to provide more nodes and edges for the embedding algorithm. As shown in Figure 2.2, refinement of the Cartesian grid cells provides enough

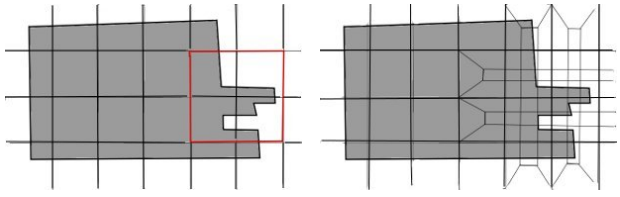
*The Pennsylvania State University, jmk5332@psu.edu
†Sandia National Laboratories, sjowen@sandia.gov



(a) Node Mapping Process (b) Curve Mapping Process

FIG. 2.1. *A potential solution for vertex mapping and curve mapping in the embedding algorithm.*

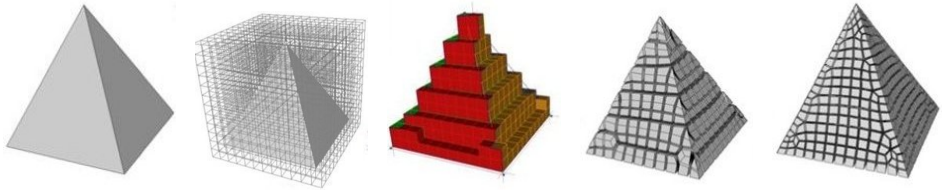
information to capture the curves and vertices of the two-dimensional geometry model. This work focuses primarily on how to automatically choose which Cartesian grid cells are refined and how to store the resulting refinement data.



(a) Problematic area of a geometry model. (b) A Cartesian grid enriched through refinement.

FIG. 2.2. *Enrichment of the Cartesian grid ensures that an acceptable number of nodes are present to capture all geometry features*

3. Algorithm Overview. As shown in Section 1, there is a need to intelligently refine a Cartesian grid and store the resulting data in order to improve the handling of small geometric features in the embedding mesh generation algorithm [2]. Figure 3.1 demonstrates the entire automatic mesh generation process.

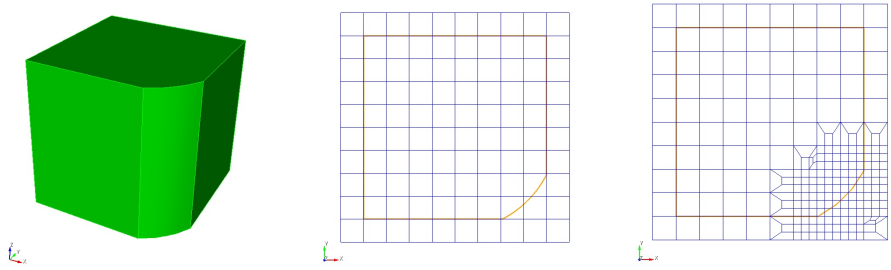


(a) Geometry (b) Cartesian Grid (c) Embedding (d) Projection (e) Sheet Insertion

FIG. 3.1. *The hexahedral mesh generation process for an arbitrary volume.*

The work to be done in this paper occurs between Figure 3.1(b) and 3.1(c). At this point a Cartesian grid has been established around the geometry, but the embedding process has not yet occurred.

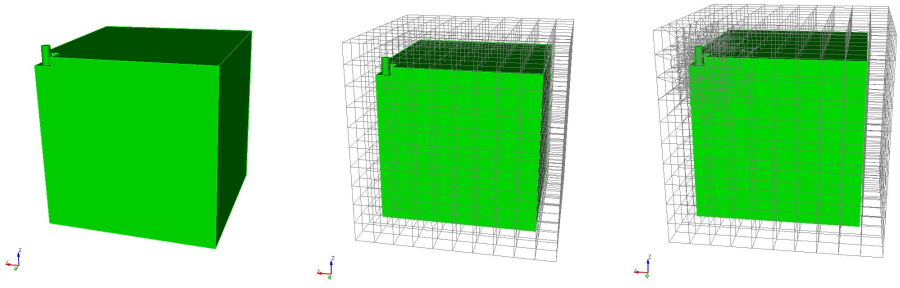
3.1. Initial Refinement Process. The first problem is determining where refinement is necessary in a given geometry model. The Cartesian grid cells are unable to capture all geometric features of a model when too many vertices exist in an area, indicating the relative size of the model is small in the region.



(a) Original geometry (b) View of the Cartesian Grid from the Top (c) View of the Refined Grid from the Top

FIG. 3.2. *Comparison of enriched mesh and Cartesian grid mesh.*

Geometry adaptive mesh sizing algorithms [3] generate mesh sizing information based on an octree lattice [4]. These sizing algorithms may be queried at an arbitrary point on a volume to determine what the mesh size should be at that point. This information can then be compared to the existing Cartesian grid cell size. If the Cartesian grid cell size is significantly larger than the sizing algorithm's suggested size, the Cartesian grid cell must be refined. An initial refinement process is executed based on data from geometry adaptive sizing algorithms [3] and the resulting mesh is stored in an auxiliary data structure. Then an interface is provided for specific refinement cases encountered by the embedding mesh generation algorithm. Figure 3.2 shows the results of an initial refinement process on a geometry model. In Figure 3.2, refinement was not necessary for the completion of the embedding algorithm process; however, the adaptive sizing algorithm indicated that a better mesh would be produced by using a smaller size. In other cases, refinement will be necessary to completing the embedding process such as in Figure 3.3.



(a) Original geometry (b) View of the Cartesian Grid (c) View of the Refined Grid

FIG. 3.3. *Comparison of enriched mesh and Cartesian grid mesh.*

3.2. Storing Refinement Information. If the initial refinement process produced a modified mesh, the enriched Cartesian grid cell information must be stored in a data structure. Full data structures exist for the storage of complex hexahedral meshes; however, the embedding mesh generation algorithm utilizes mesh refinement in a specialized way. Original mesh sizes for the Cartesian grid may be chosen well enough to avoid refinement in most areas of the grid. Only a few Cartesian grid cells will be refined in most cases. As a result,

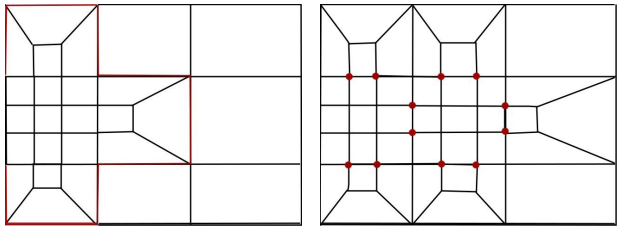
it is beneficial to preserve the Cartesian grid data structure. Most of the mesh can be created using the Cartesian grid and necessary computations can be performed on the Cartesian grid much more quickly than in a full data structure representation of the mesh.

An auxiliary data structure was created to hold enriched information for individual Cartesian grid cells. C++ Standard Template Library maps were created and added to the Cartesian grid structure implementation to hold the relationship between Cartesian grid cells and the auxiliary data structure. This allowed for a simple modification of the embedding algorithm. While traversing the Cartesian grid cells to embed information, a check of the map will show whether or not a grid cell has refinement data attached.

The auxiliary data is created after the completion of the refinement process. Newly created elements from the refinement process are grouped according to the grid cell that contains the element. These groups of cells are sent in to the auxiliary data structure constructor. A new auxiliary data structure is created to hold the refinement information present for each Cartesian grid cell that has been modified.

One task that the auxiliary data structures must perform is the assignment of continuous and unique ids for each element. This becomes a problem because the auxiliary data creates its own internal representation of data. Where the Cartesian grid and auxiliary data structure overlap or where two auxiliary data structures overlap, the same mesh element can be assigned two different id numbers. The node, edge, face, and cell ids need to remain consistent throughout the data structures.

Figure 3.4 demonstrates when data can be duplicated between the Cartesian grid structure and the auxiliary data structure. The highlighted edges are a part of both a new quadrilateral element and an existing quadrilateral element. The auxiliary data structure could attempt to create a second representation of the highlighted edges because the edges are also part of a newly created element from refinement. There should not be two ids representing the same element. Figure 3.4 also demonstrates the potential for duplication of entities between auxiliary data structures. The highlighted nodes are new; however, if another auxiliary data structure assigns an id number, the id number should be maintained in all auxiliary data structures.



(a) Duplicated edges (b) Duplicated nodes

FIG. 3.4. *Edges, nodes and faces could be duplicated in the auxiliary data cells if new ids are not assigned carefully. Edges and nodes that are at risk for duplication are highlighted in red.*

This problem may be resolved through correctly ordered searches of existing data to ensure that ids are assigned correctly. For example, data from the refinement operation may be queried to determine if the nodes associated with an edge were newly created. If they were not, the existing id may be determined from the Cartesian grid. If one or two of the nodes are new, checks must be made to ensure that the edge is not already in an auxiliary data structure. If it is already in an auxiliary data structure, the preexisting id may be assigned. Otherwise, the next available edge id is assigned.

3.3. Accessing Refinement Information during the Embedding Process. Many types of information are necessary for the embedding algorithm. For example, the embedding algorithm may need to determine a tangent vector or a face normal vector. These functions have been implemented in the Cartesian grid data structure representation. A second implementation of these functions was also placed in the auxiliary data structure. The Cartesian grid functionality was modified to query the auxiliary data if information is needed about refinement cells, faces, edges and nodes. These calculations become more complex in the auxiliary data structure due to the representation of the grid no longer being a simple Cartesian grid. Some of the information may be contained in multiple locations. For example, when determining cell adjacencies for a cell in auxiliary data, the adjacent cell might be in a different auxiliary data object or even within the Cartesian grid. If these calculations are done only in the Cartesian grid, many of the results may be determined implicitly. Element ids, associativity, and many other properties are implicitly determined because of the Cartesian grid structure. When these calculations are done on auxiliary data cells, the properties can no longer be determined implicitly. Table 3.3 lists a few of the functions that must work for any grid cell whether it was a part of the Cartesian grid or auxiliary data.

Selected Grid Functions	
cell_cells	Determines all cells adjacent to a grid cell
mid_cell	Calculates the centroid of a grid cell
face_normal	Calculates the normal vector of a cell face
node_in_cell	Determines if a node is a part of a cell
location	Returns the location of a node
edge_tangent	Calculates the tangent vector of an edge

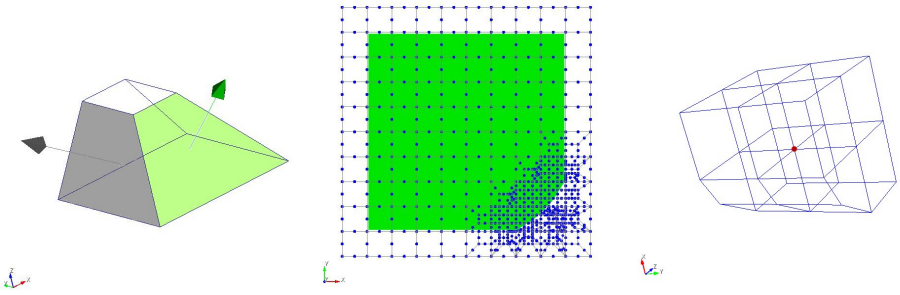
TABLE 3.1
A limited number of functions that must be implemented for the Cartesian grid cells and auxiliary data cells

3.4. Providing for Multiple Levels of Refinement. There may be rare cases where a single level of refinement may not enrich the Cartesian grid enough to adequately capture all geometry features. In this case, multiple levels of refinement are necessary. Two approaches are taken to handle this. First, the initial refinement is able to run for more than one iteration if the geometry adaptive sizing algorithm results indicate that some Cartesian grid cell sizes remain too large. The total number of times that initial refinement may run is limited to ensure that the mesh is not excessively refined by an unusual geometry case. The second way to provide for this is to allow the embedding algorithm to trigger the refinement process on a selected set of Cartesian grid cells.

The procedure called by the embedding algorithm is also equipped to handle the extraction of a currently refined mesh or plain Cartesian grid mesh, refine it and store the results in the auxiliary data structures. The extraction of the mesh occurs by maintaining a map of a grid cell to its auxiliary data structure. When extracting the mesh for the refiner, it is easy to tell which original grid cells are used and also which grid cells contain new data that must be included in the refinement process. After the refinement has occurred, the process to store the data is similar to that described in Section 3.2.

4. Examples. The examples provided in this section are problems solved by this algorithm or of calculations it performs. The figures demonstrate either scenarios where the original embedding approach [2] would fail or some of the special functionality required by the embedding algorithm [2].

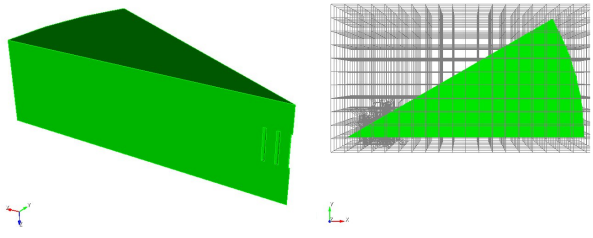
Figure 4.1 demonstrates some of the functions necessary to the embedding process [2].



(a) Quadrilateral Face Normal Vectors (b) Edge Midpoints (c) Cells Adjacent to a Node

FIG. 4.1. *Various required functions for the embedding algorithm [2].*

The normal vectors for the quadrilateral faces shown in figure 4.1 were computed by averaging the cross product of each set of two edges that share a node. Edge midpoints were calculated using a simple midpoint formula. Adjacencies in the grid are determined through properties of the arrangement of the Cartesian grid and auxiliary data structures.



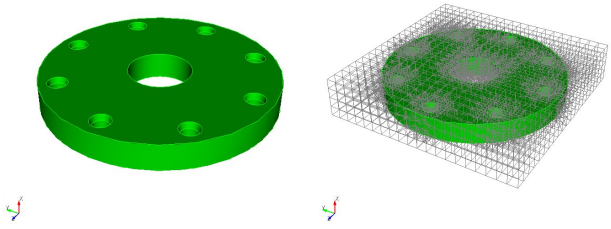
(a) Original Geometry (b) Refined Cartesian Grid

FIG. 4.2. *A wedge-shaped model that requires an enriched Cartesian grid.*

A geometry model which requires an enriched Cartesian grid is shown in Figure 4.2. Figure 4.2(a) shows that there are several small details present near the tip of the wedge-shaped geometry where a Cartesian grid could not capture all of the features without refinement. The enriched Cartesian grid for this model, shown in Figure 4.2(b), provides the necessary resolution to capture the small features.

In figure 4.3, smaller holes and more curvature in the geometry causes the Cartesian grid to be enriched at specific locations in order to more accurately capture the geometry. The hole in the center of the geometry model may not require enrichment to be captured using the existing grid cells at that location; however, the quality of the resultant mesh around that boundary is improved through refinement.

5. Conclusions. The use of a Cartesian grid in the generation of a hexahedral mesh is preferable because of the ease of calculations and the relatively small memory usage. Several cases exist where the Cartesian grid may no longer provide sufficient information to adequately capture each geometry feature. At this point, it becomes necessary to refine Cartesian grid cells based on sizing information provided by the geometry adaptive sizing algorithm [3] or based upon the mesh generation algorithm's failure to capture the features in a specific



(a) Original Geometry (b) Refined Cartesian Grid

FIG. 4.3. *A disk-shaped model that requires an enriched Cartesian grid.*

area of the geometry. Features that are small enough to require enrichment of the Cartesian grid are likely to be confined to a small section of the geometry because the initial choice of mesh size can be done such that no more than a certain percentage of Cartesian grid cells must be refined. Consequently it is beneficial to maintain the Cartesian grid structure and attach auxiliary data to the grid cells that have been refined. This solution allows the majority of calculations to be done quickly by using properties of the light-weight Cartesian grid data structure. Significantly fewer calculations are done on the refined cells contained within the auxiliary data structures. Also, the Cartesian grid structure has significant advantages over using an octree lattice as a base because of memory requirements [4]. The Cartesian grid requires much less memory, which allows for better scalability to create larger meshes for large geometries and geometries that require fine meshes.

This work will allow the embedding mesh generation algorithm to handle a wider range of geometry models. It will provide an enriched mesh base so that all features may be captured by the embedding algorith [2]. Further modifications are still needed to provide a robust mesh generation algorithm. Future work on this algorithm will include integrating the code written into the current hexahedral mesh generation algorithm. Work may also be done to include boundary sheet insertion and mesh optimization.

REFERENCES

[1] Y. ITO, A. SHIH, AND B. SONI, *Octree-based reasonable-quality hexahedral mesh generation using a set of new refinement templates*, International Journal for Numerical Methods in Engineering, 77 (2008), pp. 1809–1833.

[2] S. OWEN AND J. SHEPHERD, *Embedding features in a cartesian grid*. International Meshing Roundtable Paper, Oct. 2009.

[3] W. QUADROS, K. SHIMADA, AND S. OWEN, *3d discrete skeleton generation by wave propagation on pr-octree for finite element mesh sizing*, Engineering with Computers, 20 (2004), pp. 249–264.

[4] R. SCHNEIDERS, R. SCHINDLER, AND F. WEILER, *Octree-based generation of hexahedral element meshes*, in Proceedings of the 5th International Meshing Roundtable, 1996, pp. 205–215.

[5] J. SHEPHERD, *Topologic and Geometric Constraint-Based Hexahedral Mesh Generation*, PhD thesis, The University of Utah, Salt Lake City, UT, 2007.

[6] Y. ZHANG, T. HUGHES, AND C. L. BAJAJ, *Automatic 3d mesh generation for a domain with multiple materials*, in IMR, 2007, pp. 367–386.

A PRELIMINARY INVESTIGATION INTO UNCERTAINTY QUANTIFICATION METHODS APPLIED TO NETWORK COUPLED SYSTEMS

HAYES F. STRIPLING* AND ERIC T. PHIPPS†

Abstract. This paper summarizes an Uncertainty Quantification (UQ) analysis for a simple coupled network problem. In this framework physical components interact along an intermediate barrier (the network) instead of directly as in traditional multi-physics coupling. This additional barrier allows for another level of UQ analysis on the system, as both the individual components and global network are subject to variability arising from uncertainty in the underlying governing equations. We present a simple example of a network problem using neutron transport and heat conduction. Then, we outline the development of a UQ method using polynomial representations of the problem and the ability of this method to predict statistics and probabilities computed from the problem itself. The major finding of this report is that, given proper assumptions and choices of method parameters, the UQ methods described herein are capable of producing accurate representations of the problem and its outputs while affording considerable computational savings.

1. Introduction. The past decade has seen rapid advancement in large computational projects and increasing dependencies on these projects to support high-consequence decisions. An immediate result of this trend is the need for improved uncertainty quantification (UQ) methods in the models and algorithms which constitute these scientific simulations. Increased research and development efforts have produced more rigorous UQ methods and large-scale stochastic representations of systems which ultimately support decisions requiring probability or statistical estimates.

The idea of uncertainty quantification becomes increasingly complex in software which couples multiple sets of physics in overlapping and/or spatially separate domains. Besides adding another dimension to the error in the model (that resulting from the coupling itself), the computational burden of tracking uncertain or random parameters across component domains can grow much faster than that of the computation itself. Thus, the idea of reducing the dimension of uncertainty quantification methods in coupled systems without sacrificing accuracy in reported results is an area of interest to many computational teams.

With this goal in mind, we present a project designed to investigate uncertainty quantification and error propagation in coupled physical systems. Ultimately, the project aims to use a combination of random field modeling techniques and global polynomial representations to lessen the computational burden mentioned above. For preliminary investigation, a nuclear engineering example has been chosen in which the neutron diffusion equation and heat equation govern the neutron flux and temperature profile (respectively) in a simple one dimensional domain. These two sets of physics interact along a scalar dimension, and the global problem consists of a network joining the two separate, independently operating systems.

The goal of the analysis presented in this paper is to understand how the perturbation due to random uncertainty may effect certain quantities of interest. For example, common safety parameters associated with nuclear engineering systems include maximum fuel temperature and heat flux at fuel boundaries, both of which are directly affected by variation in material or physical parameters. Our goal is to introduce these random variations in a simple problem, propagate their effects through the network coupling, and develop some metric to quantify the resulting system response. Then, specific questions such as "What is the probability that a certain temperature in the medium or a certain heat flux at the fuel boundary will exceed a safety threshhold?" can be asked and studied through rigorous numerical techniques.

*Texas A&M University, h.stripling@tamu.edu
†Sandia National Laboratories, etphipp@sandia.gov

2. The Coupled Physical System. The simple nuclear engineering example problem is defined on a one-dimensional slab that is infinite in the y and z directions and has thickness $L = 10$cm in the x direction. At the left and right boundary of the slab, we assume fluid coolant with constant bulk fluid temperature, T_∞, but with potentially different convective heat transfer coefficients, h_L and h_R. The relevant physics is coupled neutron transport and heat transfer. The material is metallic Uranium Dioxide, UO_2, with a neutron source distributed throughout. The neutron diffusion equation [9]

$$-\frac{d}{dx}\left(D\frac{d\Phi(x)}{dx}\right) + \Sigma_a\Phi(x) = \nu\Sigma_f\Phi(x) + S_0(x), \tag{2.1}$$

governs the neutron scalar flux $\Phi(x)$ in the medium, where D is the diffusion coefficient, Σ_a and Σ_f are the neutron absorption and fission cross sections, respectively, ν is the number of neutrons emitted per fission, and $S_0(x)$ denotes the distributed neutron source. Heat transport is governed by the heat conduction equation

$$\frac{d}{dx}\left(k(x)\frac{dT}{dx}\right) = -q', \tag{2.2}$$

where $T(x)$ is the temperature, q' is a linear heat generation rate, and $k(x)$ is the thermal conductivity of the medium.

The diffusion equation is coupled to the heat equation through the temperature dependence of the nuclear cross sections. Defining the spatially averaged temperature of the slab as

$$\bar{T} = \frac{1}{L}\int_x T(x)dx, \tag{2.3}$$

then a simple approximation for the temperature dependence of neutron absorption cross sections can be written as

$$\Sigma(\bar{T}) = \Sigma(T_0)\sqrt{\frac{T_0}{\bar{T}}}, \tag{2.4}$$

where $\Sigma(T_0)$ is a measured nuclear cross section at temperature T_0 and is meant to represent any absorption cross section (e.g. fission, neutron capture, etc...). Equation 2.4 is based on the approximate inverse relationship of nuclear absorption with neutron velocity, which is proportional to the square root of the temperature of the medium.

To couple the heat equation to the diffusion equation, the linear heat generation term q' is calculated from the heat generated by fission in the medium:

$$q' = \frac{1}{L}\int_x \Phi(x)\Sigma_f E_f dx, \tag{2.5}$$

where E_f is the energy produced per fission.

At the boundaries of the domain, we apply the 'Marshak Vacuum' or 'extrapolated' boundary condition to the diffusion equation, which stipulates no neutron 'in-flow' and an extrapolation of the flux to zero at a distance 2D from the boundary:

$$\Phi(0) = 2D\frac{d\Phi(0)}{dx}, \quad \Phi(L) = -2D\frac{d\Phi(L)}{dx}. \tag{2.6}$$

Because of the surrounding fluid coolant, convective heat transfer boundary conditions are applied to the temperature T at each slab boundary:

$$k(0)\frac{dT(0)}{dx} = h_L(T(0) - T_\infty), \quad -k(L)\frac{dT(L)}{dx} = h_R(T(L) - T_\infty). \tag{2.7}$$

TABLE 2.1
Fission-Spectrum Cross Sections of UO_2 at 300K

Σ_t	$.7788cm^{-1}$
Σ_a	$.0156cm^{-1}$
Σ_f	$.0111cm^{-1}$
ν	2.2

TABLE 2.2
Physical Constants Chosen for the Deterministic System

h_1,h_2	$5\frac{W}{cm^2K}$
T_∞	$300K$
Thermal Cond, k	$10\frac{W}{cmK}$
Neutron Source, S	$10^{12}\frac{n}{cm^2s}$

To maintain realism, the physical parameters of the system (nuclear cross sections, temperatures, physical properties, etc...) are chosen to be physically realistic. Here, UO_2 is chosen as a representative medium because its thermal and neutronic properties are well documented. For preliminary analysis, a relatively cold (in a temperature sense) system is modeled. The macroscopic cross sections [8] of 10 weight-percent enriched UO_2 at 300K are summarized in Table 2.1 while values for the convection heat transfer coefficients, bulk fluid temperature, thermal conductivity, and neutron source are summarized in Table 2.2.

3. Numerical Solution to the Deterministic Coupled System. The nature of the absorption cross section temperature dependence described by Eq. 2.4 yields what the nuclear engineering community refers to as a 'negative temperature/power coefficient', and is the fundamental basis of what is sometimes called the 'inherent safety' of a reactor design. This relationship stipulates that any event resulting in a large temperature increase will reduce the absorption (fission) rate, thereby automatically shutting the reaction down. The net result of this behavior is the existence of equilibrium points at which the simple system described above is stable. To compute these equilibrium points, we apply Galerkin finite-element discretizations using linear, continuous basis functions to Eqs. 2.1 and 2.2 (for spatially uniform neutron source $S_0(x)$ and thermal conductivity $k(x)$, analytic solutions to Eqs. 2.1 and 2.2 can be derived; however to incorporate random, spatially varying uncertainty later, numerical solutions are needed). After discretization, Eqs. 2.1–2.7 represent a simple example of a network coupled system:

$$\begin{aligned} \bar{T} - g_1(u_1) = 0 \quad \text{s.t.} \quad f_1(u_1, q') = 0, \\ q' - g_2(u_2, \bar{T}) = 0 \quad \text{s.t.} \quad f_2(u_2, \bar{T}) = 0 \end{aligned} \tag{3.1}$$

where u_1 and u_2 denote the finite-element approximations to T and Φ, f_1 and f_2 are the corresponding implicit finite-element equations defining u_1 and u_2, and g_1 and g_2 are given by Eqs. 2.3 and 2.5. We refer to this as a network system because u_1 and u_2 are not directly coupled, but instead interact through the network defined by Eq. 3.1. To solve these equations, we applied a nonlinear elimination scheme [10, 15] where the NOX nonlinear solver package in Trilinos [7] is applied to the equations $f_1 = 0$ and $f_2 = 0$ to 'eliminate' u_1 and u_2 from the system:

$$\begin{aligned} \bar{T} - \tilde{g}_1(q') = 0, \quad \tilde{g}_1(q') = g_1(u_1(q')), \\ q' - \tilde{g}_2(\bar{T}) = 0, \quad \tilde{g}_2(\bar{T}) = g_2(u_2(\bar{T}), \bar{T}) \end{aligned} \tag{3.2}$$

(while the equations $f_1 = 0$ and $f_2 = 0$ are linear, a nonlinear solver was chosen for generality and flexibility). A third instance of NOX is then used to solve the (nonlinear) network defined by Eq. 3.2. The immediate advantage to this type of approach is that the 'inner' solves (those for the neutron flux and temperature profiles) are not required to exist in the same software regime. Differences in linear/nonlinear solver packages, discretization schemes, or

TABLE 3.1
Equilibrium Heat Generation and Average Temperature Values

Heat Generation Rate	$13.8144 \frac{W}{cm}$
Average Temperature	$325.3252K$

other numerical techniques do not complicate the solution process. This independence can be of great advantage for a problem involving vastly different physics which may operate more efficiently in different software architectures. However, the nonlinear elimination approach does introduce some computational costs because it may require calculation of many inner solves while the outer network solver is far away from the solution.

Table 3.1 summarizes the numerical solution to the network system 3.1 using the technique described above, the values of the physical constants in Tables 2.1 and 2.2, and a uniform spatial mesh size of $h = 0.1$cm. Moreover, Fig.3.1 illustrates the convergence of the numerical solution to an approximate analytical solution obtained by replacing the finite element discretizations with known analytic solutions. The solution does converge with decreasing mesh size, and this convergence occurs at a rate of h^2, as is expected.

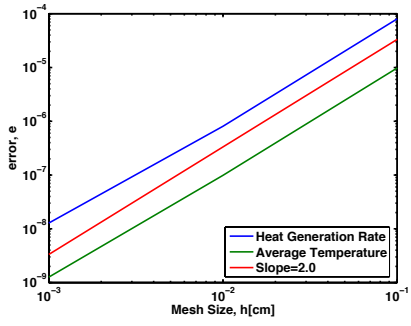


FIG. 3.1. *Convergence of the Finite-Element Solution*

4. Uncertainty Quantification of the Network System. As described in the introduction, we are interested in understanding the effects of uncertainty in simulation input data on the computed heat generation and average temperature values for this coupled system. In reality, all of the input data listed in Tables 2.1 and 2.2 are subject to uncertainty arising from material inhomogeneities and/or statistical models; for simplicity, however, we only treat the thermal conductivity k and neutron source S_0 as uncertain parameters. To quantify the effects of this uncertainty, we must make a mathematical model to describe it. An accepted strategy for doing so is a probabilistic model, where the thermal conductivity and neutron source are described by functions that vary randomly at each point in the domain. To this end, we introduce an abstract probability space $(\Omega, \mathcal{B}, P)$ where Ω is the set of outcomes, $\mathcal{B}$ is a $\sigma-$algebra consisting of subsets of Ω specifying events, and $P : \mathcal{B} \to \mathbf{R}$ is a given probability measure. We assume $k = k(x, \omega)$ and $S = S(x, \omega)$ are random fields where for each $x \in [0, L]$, $k(x, \cdot), S(x, \cdot) \in L^2_P(\Omega)$. We now seek random functions $\Phi, T : [0, L] \times \Omega \to \mathbf{R}$ such that the following equations hold for any $\omega \in \Omega$, except possibly a set of measure zero (P-almost

everywhere):

$$
\begin{aligned}
-\frac{d}{dx}D(\bar{T}(\omega))\frac{d}{dx}\Phi(x,\omega) + \Sigma_a(\bar{T}(\omega))\Phi(x,\omega) &= \nu\Sigma_f(\bar{T}(\omega))\Phi(x,\omega) + S_0(x,\omega),\\
\frac{d}{dx}k(x,\omega)\frac{d}{dx}T(x,\omega) &= -q'(\omega),\\
\bar{T}(\omega) &= \frac{1}{L}\int_x T(x,\omega)dx,\\
\Sigma(\bar{T}(\omega)) &= \Sigma(T_0)\sqrt{\frac{T_0}{\bar{T}(\omega)}},\\
q'(\omega) &= \frac{1}{L}\int_x \Phi(x,\omega)\Sigma_f E_f dx.
\end{aligned}
\tag{4.1}
$$

By modeling the heterogeneity in the domain as random fields for k and S, we obtain information as to how this heterogeneity changes the solution to the coupled system. Typically there is smoothness associated with material heterogeneity, i.e., values of k and S at some point x are likely to be close to those at a nearby point x'. Thus we assume the random fields are correlated with covariance functions given by

$$
\mathrm{cov}(x,x') = A\,\exp\left(\frac{-(x-x')^2}{L_c^2}\right) \tag{4.2}
$$

where L_c is a correlation length such that random variables $k(x,\cdot)$ and $k(x',\cdot)$ are essentially uncorrelated if $|x-x'| >> L_c$ (with a similar definition for S). It follows, then, that a correlation length on the same order as the domain width will yield a lightly varying random field, whereas a smaller correlation length will result in a field with higher-frequency oscillations.

It is well-known that the random fields for k and S can be represented through the Karhunen-Loève (KL) expansion [12], e.g.,

$$
k(x,\omega) = k_0 + \sum_{n=0}^{\infty}\sqrt{\zeta_n}\varphi_n(x)Y_n(\omega) \tag{4.3}
$$

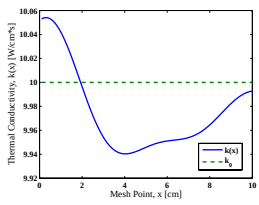
where k_0 is the mean about which the expansion should vary, and ζ_n and $\varphi_n(x)$ are eigenvalues and orthogonal eigenfunctions of the covariance function satisfying,

$$
\int_x \mathrm{cov}_k(x,x')\phi_n(x)dx = \zeta_n\phi_n(x'), \quad x' \in [0,L] \tag{4.4}
$$

and Y_n are zero-mean, uncorrelated random variables defined by

$$
Y_n(\omega) = \frac{1}{\sqrt{\zeta_n}}\int_x \phi_n(x)(k(x,\omega)-\bar{k}(x))dx. \tag{4.5}
$$

The resulting eigenvalues are non-negative and decreasing and therefore the expansion can be truncated at some $n = N$ for a given level of fidelity [4]. Thus the random fields can be approximated by a small number of random variables. Doing so requires computing numerically the KL eigenvalues and eigenfunctions that satisfy Eq. 4.4. The approach used in this paper was to approximate each eigenfunction $\phi_n(x)$ with the finite element representation $\phi_n(x) = \sum_{i=1}^{N_e}\phi_{ni}B_n(x)$, where $B_n(x)$ is the same linear continuous basis originally used to discretize the diffusion and heat equations and N_e is the number of finite elements. The weak form of Eq. 4.4 is then generated using this finite-element representation, resulting in



(a) Thermal Conductivity (b) Neutron Source

an N_e by N_e generalized eigenvalue problem [14], which we solved using an ARPACK [11] wrapper supplied by John Red-Horse of Sandia National Labs.

For a given $n = N_1$, let $\xi^1 = (Y_1, \ldots, Y_{N_1})$ be the random vector comprised of the KL random variables of the random field k. Similarly, after applying a KL expansion to the random field S truncated at a given $n = N_2$, let ξ^2 be the corresponding random vector. Figures 4.1(a) and 4.1(b) illustrate one possible realization (using Eq. 4.3) of the truncated KL expansion on $[0, L]$ for $L_c(k) = 6.0\ cm$, $L_c(S) = 4.0\ cm$, N_1=N_2=5, and uniform Y_n. After applying the same finite-element discretizations to Eq. 4.1 with k and S replaced by their respective truncated KL expansions, we obtain the stochastic version of the network problem (3.1): Find $q', \bar{T}, u_1, u_2 \in L^2_P(\Omega)$ such that P-almost everywhere

$$\begin{aligned} \bar{T} - g_1(u_1, \xi^1) = 0 \ \text{ s.t. } \ f_1(u_1, q', \xi^1) = 0, \\ q' - g_2(u_2, \bar{T}, \xi^2) = 0 \ \text{ s.t. } \ f_2(u_2, \bar{T}, \xi^2) = 0. \end{aligned} \tag{4.6}$$

For simplicity in notation in what follows, define $v = (q', T)$, $F = (\bar{T} - g_1, q' - g_2)$, and $\xi = (\xi^1, \xi^2)$. With this, Eq. 4.6 can be written

$$F(v(\omega), \xi(\omega)) = 0, \quad \omega \in \Omega. \tag{4.7}$$

4.1. Polynomial Chaos Approximation. To approximate solutions to Eq. 4.7, a polynomial chaos representation [6, 16] of v as a function of ξ is developed. Let $N = N_1 + N_2$ be the total number of random variables and let $\psi_i : \mathbf{R}^N \to \mathbf{R}$, $i = 0, \ldots, P$ be a set of N-variate polynomials that are orthogonal with respect to the inner product

$$(f, g) = \int_\Omega f(\xi(\omega)) g(\xi(\omega)) dP(\omega) = \int_\Gamma f(y) g(y) d\mu_\xi(y), \quad f, g \in L^2_{\mu_\xi}(\mathbf{R}^N), \tag{4.8}$$

where $\mu_\xi = P \circ \xi^{-1}$ is the distribution of ξ and $\Gamma = \xi(\Omega) \subset \mathbf{R}^N$. We assume μ_ξ is absolutely continuous with respect to Lebesgue measure on $\mathbf{R}^N$ so that $d\mu_\xi(y) = \rho(y)dy$ where $\rho : \mathbf{R}^n \to \mathbf{R}$ is the probability density function of ξ. Furthermore, we assume the components of ξ are independent so that $\rho(y) = \rho_1(y_1) \ldots \rho_N(y_N)$ factors. This assumption is actually true for random variables arising from a KL expansion if each ξ_i is normally distributed (i.e., if the random fields are Gaussian) since the KL representation guarantees the random variables are uncorrelated. For simplicity, however, we will extend this assumption to uniform random variables in order to proceed with the analysis. In any case, the polynomials ψ_i can be written as tensor products of univariate orthogonal polynomials of total degree at most M (for some M), i.e.,

$$\psi_i(y) = \psi_{i_1}(y_1) \ldots \psi_{i_N}(y_N), \quad i_1 + \cdots + i_N \le M, \tag{4.9}$$

where ψ_{i_j} is a degree i_j polynomial orthogonal with respect to

$$(f, g) = \int_{\Gamma_i} f(y_i) g(y_i) \rho_i(y_i) dy_i, \quad f, g \in L^2_{\rho_i}(\mathbf{R}), \tag{4.10}$$

and where $\Gamma_i = \xi_i(\Omega)$. The collection of all such polynomials are dense in $L^2_P(\omega)$ and thus v can be represented by its Polynomial Chaos Expansion (PCE):

$$v(\omega) = \sum_{i=0}^{\infty} v_i \psi_i(\xi(\omega)), \quad \omega \in \Omega \tag{4.11}$$

where convergence of the above series is taken in the L^2_P sense. Thus an approximation to v can be be obtained by truncating the above series at some fixed order P. In the case of a tensor product basis generated from univariate polynomials of order M as described above we have

$$P = \frac{(M+N)!}{M!N!}. \tag{4.12}$$

Several approaches are available for numerically approximating the coefficients v_i in the above expansion. The simplest to implement computationally is the so-called non-intrusive polynomial chaos method [1, 3, 13] where orthogonality of the basis polynomials ψ_i is exploited to obtain

$$v_i = \frac{1}{||\psi_i||^2} \int_\Gamma v(y)\psi_i(y)\rho(y)dy \approx \frac{1}{||\psi_i||^2} \sum_{j=0}^{Q} w_j v(y_j)\psi_i(y_j), \quad i = 0, \ldots, P, \tag{4.13}$$

where $\{y_j : j = 0, \ldots, Q\}$ and $\{w_j : j = 0, \ldots, Q\}$ are a set of quadrature points and weights defined by ρ. For each j, $v(y_j)$ is computed by solving $F(v_j, y_j) = 0$ for v_j. Such a method is called non-intrusive because it merely requires solving the network system F at a given set of realizations of the random variables ξ. Note however that $Q >> P$, so many more samples of the network system must be generated than there are coefficients in the expansion.

Another more intrusive approach is to formulate Galerkin residual equations [6] for the expansion coefficients $\{v_i\}$:

$$F_i(v_0, \ldots, v_p) = \int_\Gamma F(\hat{v}(y), y)\psi_i(y)\rho(y)dy = 0, \quad i = 0, \ldots, P, \tag{4.14}$$

where

$$\hat{v}(y) = \sum_{i=0}^{P} v_i \psi_i(y). \tag{4.15}$$

This defines a new nonlinear system that must be solved to obtain all of the chaos coefficients $v_0, \ldots v_P$ simultaneously. To apply Newton's method to such a system we also require its Jacobian (see, e.g., [5]):

$$\begin{aligned} \frac{\partial F_i}{\partial v_j} &= \int_\Gamma \frac{\partial F}{\partial v}(\hat{v}(y), y)\psi_i(y)\psi_j(y)\rho(y)dy \\ &\approx \sum_{k=0}^{P} J_k \int_\Gamma \psi_i(y)\psi_j(y)\psi_k(y)\rho(y)dy \quad i, j = 0, \ldots, P, \end{aligned} \tag{4.16}$$

where

$$J_k = \frac{1}{||\psi_k||^2} \int_\Gamma \frac{\partial F}{\partial v}(\hat{v}(y), y)\psi_k(y)\rho(y)dy, \quad k = 0, \ldots, P, \tag{4.17}$$

are the coefficients of the polynomial chaos expansion of the Jacobian matrix $\partial F/\partial v$. Using the definition of F we have

$$F_i = \begin{bmatrix} \bar{T}_i \|\psi_i\|^2 - \int_\Gamma g_1(u_1(y), \pi_1(y))\psi_i(y)\rho(y)dy \\ q'_i \|\psi_i\|^2 - \int_\Gamma g_2(u_2(y), \hat{\bar{T}}(y), \pi_2(y))\psi_i(y)\rho(y)dy \end{bmatrix}, \quad i = 0, \ldots, P, \tag{4.18}$$

where $\pi_1(\xi) = \xi^1$ and $\pi_2(\xi) = \xi^2$. In the semi-intrusive approach, these integrals are evaluated via quadrature, e.g.,

$$\int_\Gamma g_1(u_1(y), \pi_1(y))\psi_i(y)\rho(y)dy \approx \sum_{j=0}^{Q} w_j g_1(u_1(y_j), \pi_1(y_j)) \tag{4.19}$$

where $u_1(y_j)$ is given by solving $f_1(u_1, \hat{q}'(y_j), \pi_1(y_j)) = 0$.

The Polynomial Chaos expansion used on the example problem depicted in this paper is actually a hybrid of intrusive and non-intrusive techniques. The network PCE is computed intrusively by solving Eq. 4.18. The expansions on the discretized Eqs. 2.1 and 2.2, however, are computed entirely non-intrusively using Eq. 4.19. Therefore, we term the PCE technique applied to this problem as 'semi-intrusive'. The Stokhos package in Trilinos was used to perform these calculations based on sparse-grid quadrature methods [2] provided by Dakota [3].

5. Selection of Stochastic Expansion Parameters. As mentioned in the introduction, the goal of this analysis is to understand the procedure for reducing the cost of a given Uncertainty Quantification method while maintaining a required level of accuracy. The cost of the PCE method is directly related to Eq. 4.1, which is symmetric in both the stochastic dimension, N, and univariate polynomial degree, M. Therefore, we search for the minimum values of these parameters such that the UQ method maintains the desired level of accuracy.

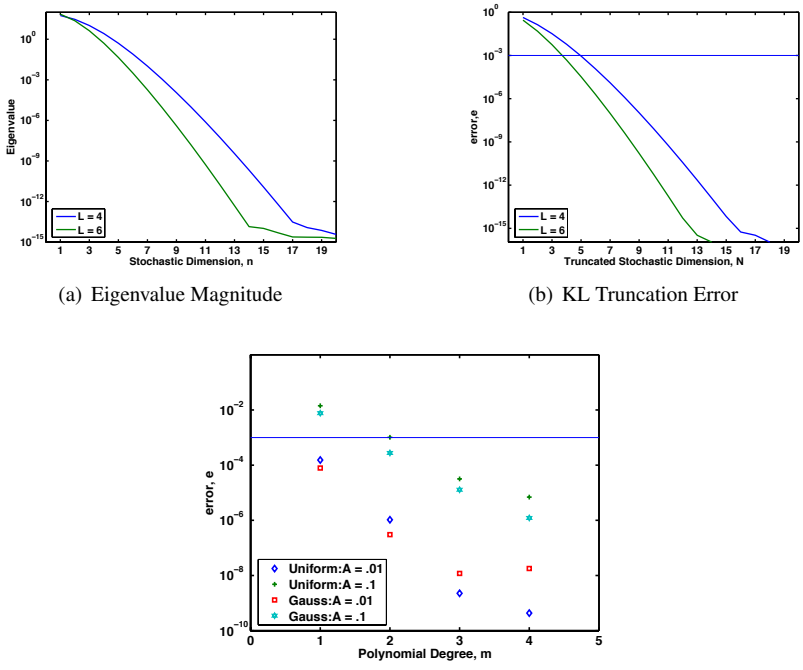
To help define this desired level of accuracy, we propose a Monte Carlo probability estimate of exceeding a predefined domain-averaged temperature, $\bar{T}_{lim}$, AND linear heat generation rate, q'_{lim}. The expected error in such a calculation decays proportionally to $\frac{1}{\sqrt{N_s}}$, where N_s is the number of Monte Carlo samples. For the purposes of illustration, we let N_s=1,000,000, making the error in the Monte Carlo probability on the order of 10^{-3}. Thus, the goal is to minimize M and N while maintaining at least this level of accuracy from the stochastic system and its polynomial representation.

5.1. Choice of Minimum Stochastic Dimension using KL Truncation Analysis. As discussed in Section 4, the KL expansion (Eq. 4.3) of the random field may be truncated at some $n = N$ to achieve a desired level of accuracy. Figure 5.2(a) illustrates the decay rate of the eigenvalues for the two random fields pictured in Figures 4.1(a) and 4.1(b). Recall that these random fields differ in their correlation lengths ($L_c(k) = 6.0, L_c(S) = 4.0$), and note that the decay rate is larger in magnitude for increasing correlation length.

The L^2 truncation error for stochastic dimension $n = N$ can be calculated by computing the proportion of the total sum of the KL expansion contributed by the truncated terms, $n = N + 1 \ldots \infty$. To approximate this proportion, we approximate the infinite sum as a sum of the first 100 eigenvalues and plot the truncation error calculated as a function of N:

$$e = \frac{\sum_{n=N+1}^{100} \zeta_n}{\sum_{n=1}^{100} \zeta_n}. \tag{5.1}$$

Figure 5.2(b) illustrates the results for the two correlation lengths defined in this problem. The horizontal line at $e = 10^{-3}$ indicates the maximum allowable error from the KL truncation. For these specific fields and this specific problem, we have chosen to use a minimum



(a) Eigenvalue Magnitude (b) KL Truncation Error

FIG. 5.2. *Error in Truncation of KL Expansion*

stochastic dimension of $N = 5$ for each of the inner solutions to the heat and diffusion equations. Thus, the stochastic dimension of the global problem is 10. At this truncation level, we believe that the error in the KL expansion will not dominate the error in the Monte Carlo results.

5.2. Choice of Minimum Polynomial Degree using Point-Wise Error Analysis. After we choose the minimum stochastic dimension, we must decide upon the minimum required polynomial degree, M, in the chaos expansion to adequately represent the problem. This value is a function of the non-linearity or complexity of the specific problem. In the case of this example, the required polynomial degree mostly depends on the amplitude, A, of the variability in the random fields of the thermal conductivity and neutron source terms (at a given oscillation frequency governed by L_c). We control this parameter by changing the range of the random variables Y_n for a given KL realization.

For a given KL expansion with a set amplitude, A, a point error analysis can determine the degree M required for a desired level of accuracy. This procedure involves the evaluation and comparison of the system and its polynomial representation (for increasing m) at a specified value of ξ. We repeat this process twice, once using Gaussian and once using uniform random variables. Figure 5.2 illustrates this point error analysis for the semi-intrusive polynomial expansion applied to the stochastic system.

As Fig. 5.2 indicates, the network problem with the larger amplitude requires a higher order polynomial to resolve the larger range of the random parameters to a desired level of accuracy. An interesting characteristic of the plot is that the Gaussian random variables are more accurate than uniform random variables at lower polynomial orders, but the opposite is true at higher polynomial orders. We believe this behavior occurs because the higher order polynomials require more quadrature points for evaluation, and in the case of the unbounded Gaussian variables, it is likely that more extreme (i.e. closer to zero) values of the KL real-

TABLE 6.1
Mean and Standard Deviation of System Solution

Value	Network Problem [-1 1]	PCE, Uniform [-1 1]	PCE, Gauss. ($\sigma = \frac{1}{3}$)	PCE, Gauss. ($\sigma = \frac{1}{\sqrt{3}}$)	PCE, Gauss. ($\sigma = 2.0$)
Mean, $q'[\frac{W}{cm}]$	13.81174	13.81224	13.81368	13.81224	13.78728
St. Dev., q'	0.59364	0.59359	0.34273	0.59360	2.05457
Mean, $\bar{T}$ [K]	325.35293	325.35302	325.33447	325.35308	325.69672
St. Dev., $\bar{T}$	1.19375	1.19375	0.68832	1.19443	4.29663

izations are occuring. In the case of the heat equation, if the thermal conductivity approaches zero, the problem becomes increasingly ill-posed. This error, then, would propagate through the network and affect the accuracy of the entire solution.

Figure 5.2 indicates that the minimum allowable univariate polynomial order (i.e. that which affords an error less than 10^{-3}) is M=3. Thus, we believe that polynomial chaos expansions or univariate order M=3 and stochastic dimension N=5 will be accurate to, at least, the order of the error in a 1,000,000 sample Monte Carlo run.

6. Results from Monte Carlo Probability Calculations. The purpose of the Monte Carlo calculations is two part: First, we are interested in comparing a probability calculation arising from samples of the problem itself to that computed from samples of its polynomial representation. Second, we are interested in understanding how the computations are affected by the difference between Gaussian and uniform random input variables. First, the network system will be sampled 1,000,000 times using uniform random variables on the interval [-1 1]. Then 20 batches of 1,000,000 samples each will be taken from each of four polynomial representations: one expanded upon uniform random variables on the interval [-1 1] and three from Gaussian random variables, each with different standard deviations: σ=$\frac{1}{3}$ such that ~99% of the random values fall on the interval [-1 1]; σ=$\frac{1}{\sqrt{3}}$ to match the first two moments of the uniform random distribution; and σ=2.0 to illustrate problem behavior with highly varying inputs. In the end, we are interested to see the degree of accuracy of the UQ problem and the extent to which the problem (which is defined with uniform random variables) is sensitive to the use of uniform or Gaussian random variables.

Table 6.1 lists the mean and standard deviation of temperature and heat generation solutions computed from the samples of the problem and computed from first two moments of the four different polynomials. As expected, the reported means of the first two columns (those generated from uniform distributions) agree to three digits of accuracy. The means computed from expansions on Gaussian random variables seem to be on the right order, but are diverging with increasing σ. Also of interest is the changes in the standard deviation with changes in the Gaussian input standard deviation. As expected, if the input random variables are given a higher standard deviation, then the output parameters will also vary about a wider margin. Additionally, this relationship appears to be linear through the range of standard deviations tested in this report.

Of further interest, however, is an example calculation of the probability that a single instance of the problem will result above $\bar{T}_{lim}$ and q'_{lim}. In an effort to limit the number of required samples, we choose to calculate a probability that we know to be on the order of 5%: the probability of exceeding temperature and power limits of 327.2K and 14.51$\frac{W}{cm}$ respectively. Table 6.2 summarizes the results of this calculation.

TABLE 6.2
Example Probability of Exceeding Operational Limits

Value	Network Problem [-1 1]	PCE, Uniform [-1 1]	PCE, Gauss. ($\sigma = \frac{1}{3}$)	PCE, Gauss. ($\sigma = \frac{1}{\sqrt{3}}$)	PCE, Gauss. ($\sigma = 2.0$)
Probability	5.1661%	5.1950%	0.00%	0.31854%	35.7000%
St. Dev.	n/a	0.01655%	0.00%	0.00520%	0.04933%

Tables 6.1 and 6.2 reveal that, in the case of this problem, we are able to compute the statistics and probability estimates to the desired level of accuracy given the correct choice of stochastic dimension, polynomial degree, and random input distribution. The choice of input distribution is especially important; in this case, we know that the input distribution to the problem itself is uniform and therefore expect a polynomial representation expanded about uniform random variables to produce accurate calculations. In testing the Gaussian random variables, we see that the difference in the distribution affects the probability calculation despite the first two moments being relatively accurate. Therefore, we find that the correct choice of UQ input distribution must carefully consider the original problem definition and the distribution of its uncertain parameters.

Another important result from the course of this study is that, given the proper choice of UQ method and parameters, the use of the Polynomial Chaos representation of the network problem affords massive computational savings while preserving the desired level of accuracy. For example, the time required to perform 1,000,000 samples and compute statistics for this (simple) network system is on the order of days. Alternatively, the time required to compute the polynomial expansion about the system, take 20,000,000 samples, and compute the same statistics is on the order of minutes. Yet, as illustrated by Tables 6.1 and 6.2, the statistics for the valid UQ method are within the accuracy bounds specified in Section 5. We predict that the same cost savings would extend to more complex systems as well.

7. Conclusions. This simple coupled network problem serves as a baseline example of the kinds of systems in which UQ methods can provide large advantages and new capabilities. Using a careful definition and understanding of the underlying problem and goals, we were able to develop a UQ problem with specific parameters and features to achieve a desired level of accuracy in the final result. We then tested our choices of these parameters by comparing the actual system behavior against both the behavior of the chosen UQ method and a different UQ method based on a different set of assumptions. Our finding was that the proper choice of UQ parameters and assumptions is crucial to adequate problem representation and that, given the proper choice of these parameters, we achieved our desired level of accuracy while saving orders of magnitude in computational time.

REFERENCES

[1] S. ACHARJEE AND N. ZABARAS, *A non-intrusive stochastic galerkin approach for modeling uncertainty propagation in deformation processes*, Computers & Structures, 85 (2007), pp. 244–254.

[2] V. BARTHELMANN, E. NOVAK, AND K. RITTER, *High dimensional polynomial interpolation on sparse grids*, Advances in Computational Mathematics, (2000).

[3] M. ELDRED, C. WEBSTER, AND P. CONSTANTINE, *Evaluation of non-intrusive approaches for Wiener-Askey generalized polynomial chaos*, in 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, April 2008.

[4] P. FRAUENFELDER, C. SCHWAB, AND R. A. TODOR, *Finite elements for elliptic problems with stochastic coefficients*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 205–228.

[5] R. Ghanem and R. Kruger, *Numerical solution of spectral stochastic finite element systems*, Computer Methods in Applied Mechanics and Engineering, 129 (1996), pp. 289–303.

[6] R. G. Ghanem and P. D. Spanos, *Stochastic finite elements: a spectral approach*, Springer-Verlag, New York, 1991.

[7] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, and K. Stanley, *An overview of the Trilinos package*, ACM Trans. Math. Softw., 31 (2005).

[8] K. A. E. R. Institute, *Table of nuclides*. website: http://atom.kaeri.re.kr, 2000.

[9] J. R. Lamarsh and A. J. Baratta, *Introduction to Nuclear Engineering*, Prentice-Hall, Upper Saddle River, New Jersey, third ed., 2001.

[10] P. J. Lanzkron, D. J. Rose, and J. T. Wilkes, *An analysis of approximate nonlinear elimination*, SIAM J. Sci. Comput., 17 (1996), pp. 538–559.

[11] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[12] M. Loève, *Probability theory*, Springer-Verlag, New York, fourth ed., 1977. Graduate Texts in Mathematics, Vol. 45 and 46.

[13] M. Reagan, H. Najm, R. Ghanem, and O. Knio, *Uncertainty quantification in reacting-flow simulations through non-intrusive spectral projection*, Combustion and Flame, 132 (2003), pp. 545–555.

[14] C. Schwab and R. A. Todor, *Karhunen-Loeve approximation of random fields by generalized fast multipole methods*, Journal of Computational Physics, 217 (2006), pp. 100–122.

[15] J. T. Wilkes, *A new method for solving systems of nonlinear equations in circuit simulation*, tech. rep., CiteSeer [http://cs1.ist.psu.edu/cgi-bin/oai.cgi] (United States), 1994.

[16] D. Xiu and G. E. Karniadakis, *The Wiener-Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644 (electronic).

A FAST ILU PRECONDITIONING-BASED SOLVER FOR THE CHARGE EQUILIBRATION PROBLEM

HASAN M. AKTULGA*, ANANTH Y. GRAMA†, STEVE PLIMPTON‡, AND AIDAN THOMPSON§

Abstract. Charge equilibration (QEq) is the problem of assigning partial electrostatic charges to individual atoms in a molecular dynamics (MD) simulation. It utilizes the neighborhood information of atoms to determine partial charges which minimize the electrostatic energy in the system subject to the constraint of fixed total charge. The formulation of the QEq problem gives us a large sparse linear system of the form $Ax = b$ which can be solved using well-known Krylov subspace methods uch as *CG* or *GMRES*. However, application of these solvers to the QEq problem only with simple optimizations such as a diagonal preconditioner and a good initial guess does not yield satisfactory results. Instead, we present an ILU preconditioning-based algorithm which can solve the QEq problem much faster while still using the *CG* and *GMRES* solvers. We demonstrate the performance of the ILU pre-conditionaer on some sample systems. Due to the poor scaling of ILU factorization algorithms, we restrict our attention to single-processor calculations and small ($< 10^4$ atoms) systems. We briefly point out some potential problems on a large-scale parallel simulation ($> 10^6$ atoms) and some solution approaches that can be used to deal with these large systems.

1. Introduction. Molecular simulation methods has been an attractive field of study for many scientists and researchers in various areas. Molecular simulation methods span a wide scale ranging from *ab-initio* methods to classical MD methods. While *ab-initio* methods treat electrons explicitly making it possible to observe reactions in the system of interest and to analyze it in great detail, classical MD simulations model electrons together with the nuclei as a single point in space. Fixed bonds and fixed partial charges assumptions used in most classical MD methods together with the lost electronic degrees of freedom restrict the application areas of these methods but gives them speed-ups that can never be achieved by *ab-initio* methods while still producing valuable results.

ReaxFF is a force field developed by Adri van Duin et. al [7] to bridge the gap between *ab-initio* and classical MD methods. It still treats electrons together with the nuclei as a single point in space but it does not have any fixed bonds or fixed charges restrictions making it possible to study systems with reactions, hence the name *ReaxFF*. In a reactive system the partial charges on atoms keep changing as a result of reactions. Indeed, even in the absence of reactions partial charges constantly fluctuate as the neighborhood of atoms change. In *ReaxFF*, we use the charge equilibration (QEq) method to address the problem of determining the partial charges at any given timestep during the course of an MD run. The basic idea behind the QEq method is to utilize the neighborhood information of atoms to determine partial charges with the objective of minimizing the electrostatic energy in the system subject to the constraint of fixed total charge. We refer interested readers to [6] for more details.

Polarizable force fields which are essentially classical MD methods without the fixed charges assumption have started becoming popular in recent years [1]. We believe that polarizable force field methods can also benefit from the QEq method, therefore we have implemented the QEq method as a *fix* in the popular LAMMPS package from Sandia National Laboratories [4]. Readers who want to use the QEq method embedded in LAMMPS can refer to the LAMMPS documentation website [5].

In Section 2 we present the mathematical formulation of the QEq problem. Later in Section 3 we show how the QEq solver used in our current implementation of *ReaxFF* starts dominating the computation time as the tolerance value used as a stopping criteria for the

*Purdue University Computer Science Department, haktulga@cs.purdue.edu
†Purdue University Computer Science Department, ayg@cs.purdue.edu
‡Sandia National Laboratories, sjplimp@sandia.gov
§Sandia National Laboratories, athomps@sandia.gov

iterative solver is decreased In Section 4, we propose a new algorithm to solve the QEq problem for moderate sized systems and analyze the algorithm in detail. Later in Section 5, we summarize the results of some experiments we have performed in *Matlab* to compare the performances of the proposed algorithm and the method currently being used. Finally, in Section 6, we briefly discuss some potential problems that can be encountered while solving the QEq problem for a large-scale system in a parallel setting and we point out to some possible solution approaches that can be used to generalize our new algorithm for dealing with large scale systems.

2. Theory. In this section we present the formulation of the charge equilibration problem as presented in [3]. QEq is an approximation to the problem of determining partial charges on atoms for a given configuration. It is a simplified alternative to the much more expensive (but also highly accurate) way of computing partial charges using an *ab-initio* method. The basic idea behind the QEq method is to assign partial charges to atoms so as to minimize the electrostatic energy of the system subject to the constraint that the net charge remains constant (the net charge of a neutral system is 0 which is what we will be assuming for the rest of this section). In mathematical terms, the problem can be stated as:

$$\begin{aligned}
&\text{Minimize } E_{ele} = \sum_i \chi_i q_i + \frac{1}{2}\sum_{i<j} H_{ij} q_i q_j \\
&\text{where } H_{ij} = J_i \delta_{ij} + \frac{1-\delta_{ij}}{r_{ij}^3 + \gamma_{ij}^{-3\,\frac{1}{3}}} \\
&\text{subject to } \sum_i q_i = 0
\end{aligned}$$

The method of Lagrange multipliers is used to solve the above minimization problem and we obtain the following linear systems:

$$-\chi_k = \sum_i H_{ik} s_i \tag{2.1}$$

$$-1 = \sum_i H_{ik} t_i \tag{2.2}$$

Finally, charges are computed from the solutions to these two linear systems:

$$q_i = s_i - \frac{\sum_i s_i}{\sum_i t_i} t_i \tag{2.3}$$

3. Performance. Charge equilibration is just a precursor for computing the electrostatic energy of the system and the resulting linear systems given in (2.1) and (2.2) can certainly be solved using a direct solver. But the coefficient matrix H is an $N \times N$ matrix where N is the number of atoms in a given system and a direct solver would scale with $O(N^3)$ in this case. This makes direct solvers unsuitable for even moderate sized systems on a single processor which have a few thousand atoms only. The fact that H matrix is not diagonally dominant makes basic iterative schemes such as *Jacobi*, *Gauss-Seidel* or *SOR* methods also infeasible.

However, H is a sparse linear system so we can use well-known Krylov subspace methods such as *CG* or *GMRES* algorithms for solving the linear systems shown in (2.1) and (2.2). The sparsity of the coefficient matrix comes from the fact that no matter what the system size

is, we use the neighboring atom information within a given cutoff distance, r_{cut} (typically r_{cut} = 10 Å). Even though r_{cut} and the number of atoms that can be found inside the sphere with the radius r_{cut} will change from system to system, we can safely say that it will be on the order of a few hundred atoms. Therefore the number of non-zeros in H will be on the order of a few hundred entries per row.

Based on our observations on many different molecular systems, we can say that H carries a heavy diagonal and this property gives us an easy to apply preconditioner, namely the diagonal preconditioner. Another observation is that in MD simulations timestep lengths are chosen to be very short (on the order of a few femtoseconds, in fact less than a femtosecond in *ReaxFF*) and therefore positions of atoms change very slightly between consecutive timesteps. This observation leads to the fact that the solutions to (2.1) and (2.2) in the previous timestep give very good initial guesses about the solutions to the systems at the present timestep. Indeed, making linear or quadratic extrapolations on the solutions of the last few steps, better initial guesses can be made.

We come up with a quite simple linear solver in the light of the observations above. In the current implementation of *ReaxFF*, we are using the *GMRES* algorithm with a diagonal preconditioner which does a linear extrapolation from the solutions of two previous timesteps to make a good initial guess about the problem at hand. We have also implemented the *CG* algorithm with the same optimizations to see how it compares to the *GMRES* method for the QEq problem. As can be seen in tables in the upcoming sections, *GMRES* is a much better solver for our needs. But the *CG* algorithm can be made parallel very easily, so it has its own advantages.

3.1. QEq Dominance. The downside of using a Krylov subspace method over a direct solver is that the solution we get is actually an approximate solution within the tolerance specified. For this reason it is crucial that we can solve the QEq problem with great accuracy because small errors in atomic charges can accumulate as the simulation progresses and remove energy from the system. In addition, we need to come up with a good solution in a reasonable amount of time so that the linear solve routines do not become the bottleneck in the simulation.

To illustrate how QEq calculations can start dominating the computational time, we have chosen to work with a 343 molecule hexane system, C_6H_{14}. We have taken a snapshot of the hexane system equilibrated under 1 atm and 200 K. In table 3.1, we show how the amount of time spent in QEq increases as we decrease the QEq tolerance. In order to achieve the lower tolerance, *GMRES* has to make more iterations thus increasing the time spent in QEq and the percentage of QEq calculations in total computation time. The fact that we need to spend 68% of the total computational time in QEq when we lower the tolerance to $1e-8$ demonstrates the potential QEq dominance in *ReaxFF* simulations and the need for much better solvers.

TABLE 3.1

QEq Dominance: *Columns* 3-5 *show the amount of time (in sec.) spent on each major component of our* ReaxFF *code: neighbors generation, bonded interactions, non-bonded interactions. Since QEq is a pre-cursor to electrostatic energy computations, we have included the QEq part within non-bonded interactions. The last* 3 *columns give detailed information about the QEq part: time spent in QEq (in sec.), average number of matrix-vector multiplications at each step, percentage of QEq calculations in total computation time.*

tolerance	total	neighbors	bonded	nonb	QEq	matvecs	QEq%
$1e-4$	4.49	0.46	0.31	3.67	1.74	10.36	39%
$1e-5$	4.71	0.47	0.32	3.87	1.95	11.42	41%
$1e-6$	5.16	0.47	0.33	4.30	2.38	14.94	46%
$1e-7$	6.32	0.46	0.32	5.49	3.58	27.03	57%
$1e-8$	8.30	0.43	0.29	7.54	5.64	55.64	68%

4. Proposed Algorithm. One of the most popular preconditioning techniques is using the L and U matrices from the incomplete LU (ILU) factorization of the coefficient matrix. Table 4 summarizes how we can improve the number of iterations of our linear solvers by applying an ILU preconditioner on the H matrix. Using an ILU preconditioner gives us an excellent improvement over the diagonal preconditioner when we look at the number of iterations required to solve the QEq problem. However, we cannot see any improvement on the time required to solve the same problem and timing is what we are ultimately interested in. The reason for the large numbers in ILU preconditioned solvers' timing columns is that given times include the time required for performing the ILU decomposition in addition to the time taken by the linear solve procedures themselves. *Matlab* provides the luinc($\bullet$, $\bullet$) routine where the first argument is the matrix to perform the ILU decomposition on, and the second argument defines the threshold for the ILU decomposition algorithm. In our case, times required for completing luinc(H, $1e-1$), luinc(H, $1e-2$) and luinc(H, $1e-3$) on the coefficient matrix were 0.33, 4.20, 15.1 seconds, respectively.

Based on the data at Table 4, there is another advantage of using an ILU preconditioner to solve the QEq problem. As can be seen, when we use an ILU preconditioner, we can lower the QEq tolerance at the expense of much fewer extra iterations compared to using the diagonal preconditioner.

TABLE 4.1

Effect of the ILU preconditioner on the number of iterations and total time required for solving the QEq problem using Matlab 7.6 *on a machine with a* 2.66*GHz Intel i7 processor and* 6 *GB of memory. These data was obtained using a snapshot from a 6540 atom bulk water system equilibrated at* 200 *K under* 1 *atm pressure.*

	tol=$1e-6$		tol=$1e-10$	
solver	iterations	time (sec.)	iterations	time (sec.)
pcg, diagonal	34	0.78	98	2.11
pcg, luinc($1e-1$)	–	–	–	–
pcg, luinc($1e-2$)	5	4.56	16	4.86
pcg, luinc($1e-3$)	3	15.5	8	15.7
gmres(50), diagonal	12	0.22	74	1.07
gmres(50), luinc($1e-1$)	6	0.76	37	1.38
gmres(50), luinc($1e-2$)	3	4.47	11	4.63
gmres(50), luinc($1e-3$)	2	15.5	7	15.6

Despite the gains of reduced number of iterations and a much well-behaved coefficient matrix, using the ILU preconditioning technique is still not preferable over a QEq solver with a diagonal preconditioner. This is solely because of the large amount of time required to compute the L and U factors. However, the context of the QEq problem that we are dealing with gives us an opportunity to offset this disadvantage. Like in any other MD simulation, in *ReaxFF* the displacement of atoms between consecutive timesteps is very small (on the order of *picometers*???). This means that the coefficient matrix H, which is computed based on the pairwise distances between atoms, will most likely retain its structure and contain very similar values in its entries between consecutive timesteps. This presents us the opportunity to use the L and U matrices computed at a timestep as nice preconditioners in subsequent timesteps as well.

Let us denote the matrices resulting from an ILU factorization at timestep t by L_t and U_t, and the number of iterations required to solve the QEq problem at timestep t by using L_t and U_t as preconditioners by $n_{t,t}$. Similarly, let $n_{t2,t1}$ denote the number of iterations for solving the QEq problem at timestep t_2 by using L_{t1} and U_{t1} as preconditioners. Clearly, for $t_2 \neq t_1$ we would expect to have $n_{t_2,t_2} \leq n_{t_2,t_1}$. This is because of the fact that H matrix changes over time

(even if it is a very slow change) and the best preconditioner for solving the QEq problem at any timestep can most likely be obtained by performing the ILU decomposition of the H matrix at that timestep. We will refer to the difference $n_{t_2,t_1} - n_{t_2,t_2}$ as "excessive iterations" at timestep t_2. We denote by ins_{t_0} the ideal maximum number of subsequent steps following step t_0 so that $\forall t\colon t_0 \leq t \leq t_0 + ns_{t_0}$, n_{t,t_0} is not more than some threshold determined by $n_{t,t}$ and c where c is a parameter denoting the acceptable number of excessive iterations at any timestep. More precisely,

$$ins_{t_0} = t_{max_0} - t_0 \text{ where } t_{max_0} = \max_{t_m, t_m \geq t_0} \{\forall t \in [t_0, t_m], n_{t,t_0} \leq n_{t,t} + c\} \tag{4.1}$$

Ideally, we would compute the ILU factorization of the H matrix when we start an MD simulation and use L_{t_0} and U_{t_0} matrices as preconditioners for the first ins_{t_0} timesteps. Then we would recompute the ILU(H) at timestep $t_{ins_{t_0}}$ and start using the resulting upper and lower triangular matrices as preconditioners in the following timesteps. But as noted, this is only an ideal scenario. It is ridiculous to first compute L_t, U_t at timestep t and use them as preconditioners to find out what $n_{t,t}$ is and then solve the QEq problem once more at the same timestep to see how well n_{t,t_0} compares to $n_{t,t}$. Instead, we propose using n_{t_0,t_0} as an estimate for $n_{t,t}$ in order to determine whether we should recompute ILU(H) or not. So we introduce ns_{t_0} to replace ins_{t_0}:

$$ns_{t_0} = t_{max_0} - t_0 \text{ where } t_{max_0} = \max_{t_m, t_m \geq t_0} \{\forall t \in [t_0, t_m], n_{t,t_0} \leq n_{t_0,t_0} + c\} \tag{4.2}$$

Consequently, we propose the method outlined in algorithm 1 to solve the QEq problem.

Algorithm 1 The new ILU preconditioning-based algorithm that we propose for solving the QEq problem.

```
ComputeH(H_0)
L,U ← ILU(H_0)
{use the initial ILU preconditioner typically for 2-3 timesteps}
{until we start obtaining good initial guesses}
for t = 0 to t_init do
   QEq_solve(H_t, L, U, tol)
   t ← t + 1
   ComputeH(H_t)
end for
{now that we can make good initial guesses}
{we can record our initial estimate for n_{t,t}}
n ← QEq_solve(H_t, L, U, tol)
est_n ← n
for t = t_init + 1 to max_t do
   ComputeH(H_t)
   if n > est_n + c then
      L,U ← ILU(H_t)
      n ← QEq_solve(H_t, L, U, tol)
      est_n ← n
   else
      n ← QEq_solve(H_t, L, U, tol)
   end if
   t ← t + 1
end for
```

4.1. Analysis of the New Algorithm. The effectiveness of algorithm 1 ultimately depends on two things: first how does the number of iterations required to solve a given system using the new algorithm with pre-computed L, U matrices compare to the performance of the method outlined in section 3, and second how large is ns_t typically for a timestep t that we compute the ILU factorization in.

It is apparent why we are interested in fewer number of iterations performed by the linear solver: fewer number of iterations means fewer number of matrix-vector multiplications and fewer cpu cycles required to solve the problem at hand. However, we have to note that the application of an ILU preconditioner is more costly than the simple diagonal preconditioner because resulting L, U matrices have more non-zero entries. The number of non-zero entries in L, U matrices depend on the threshold chosen: a lower threshold means more non-zero entries. On the other hand, as can be seen in table 4, as we decrease the ILU factorization threshold, we get fewer number of iterations required to solve the QEq problem no matter which linear solver we use or how accurate we want the solution to be. So there is a clear trade-off here and one needs to choose the ILU threshold carefully.

Our experiments, not shown in this paper, suggest that $1e-2$ is the best choice among the thresholds used in table 4. Let $L(h)$ and $U(h)$ denote ILU factors using a threshold h. $L(1e-1)$, $U(1e-1)$ is far from nicely approximating the real LU factors, so it requires a lot more iterations than using $L(1e-2)$, $U(1e-2)$ as preconditioners. Also the number of non-zeros in $L(1e-2)$, $U(1e-2)$ is not significantly larger than the non-zeros in $L(1e-1)$, $U(1e-1)$ matrices, so there is not a significant difference in the time required to apply both preconditioners. On the other hand, lowering the threshold to $1e-3$ introduces too many non-zeros making $L(1e-3)$, $U(1e-3)$ more expensive preconditioners to apply than $L(1e-2)$, $U(1e-2)$. And the gain in the iteration count does not really compensate for the extra cost in the application of $L(1e-3)$, $U(1e-3)$ as preconditioners, therefore we do not see a clear improvement in the running time of the linear solver by decreasing the threshold from $1e-2$ to $1e-3$. Hence we have decided that $1e-2$ is the best choice among the three of them.

Now we discuss the second question we have raised at the beginning of this section. The typical value of ns_t will vary depending on a number of factors that determine the mobility of atoms in a system such as the very own 'characteristic of the system, the phase that the system is in, whether the system has reached equilibrium or not, the temperature of the system, etc. In addition to these, the QEq tolerance we have chosen and the number of acceptable excessive iterations to us (denoted by c) will definitely affect the frequency of ILU factorizations to be performed during the course of a simulation. For instance, the mobility of atoms in solid systems is very low. Therefore we would expect to have very large ns_t values, on the order of thousands of steps, while working with solid systems. But still a low QEq tolerance or a low value for c would induce a lower ns_t value.

Even though ILU factorization is cheaper than computing the real LU factors, the algorithm is essentially the same: A Gaussian elimination where resulting entries lower than the specified threshold are dropped. So even a sparse implementation of the ILU factorization algorithm is still very expensive and not quite scalable. Therefore the value of ns_t is crucial to offset the cost associated with doing the ILU factorization at step t. The larger the ns_t is, the lower the average cost of ILU factorization per timestep would be for subsequent steps. In the next section, we will be showing by experiments that ns_t is large enough to amortize the cost of the ILU factorization during a simulation.

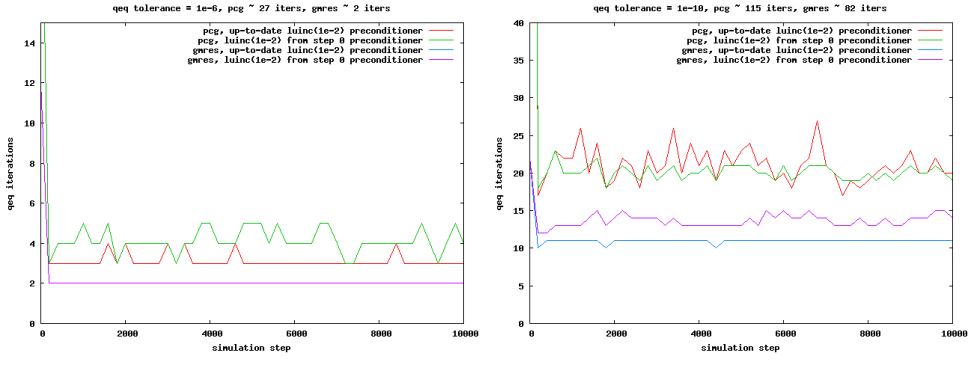
On a side note, we would like to mention that in our new algorithm if we were to pick up n_{t_0,t_0} as our first estimate, we would most likely never be recomputing the preconditioning matrices L and U again during the entire simulation. Because we do not have a good initial guess at the start of the simulation and n_{t_0,t_0} is most likely very large compared to the number

of iterations that will be required in the upcoming steps. Therefore our algorithm would start suffering from large number of excessive iterations as the simulation progresses. To alleviate this problem, in algorithm 1 we do not record the number of iterations performed in the first few timesteps of the MD simulation.

5. Experiments. In this section, we seek the answers to the two questions mentioned in section 4.1 by working with two systems that has very different characteristics, a PETN crystal and a bulk water system. Our goal is to demonstrate how effective the new algorithm could be on real scenarios.

First we present our results on the PETN crystal which is a solid material. Figures 5.1(a), 5.1(b) compare the values of $n_{t,0}$ and $n_{t,t}$ for a PETN crystal which contains 3712 atoms using the *PCG* and *GMRES* solvers in *Matlab*. For *PCG* at both $1e-6$ and $1e-10$ QEq tolerances, we get a significant improvement in terms of the iteration count using the new algorithm over using the diagonal preconditioner (27 vs 4 at $1e-6$, and 115 vs 21 at $1e-10$ on average). For *GMRES*, we do not get any improvement at $1e-6$ QEq tolerance (2 vs 2 iterations) but again when we decrease the tolerance to $1e-10$ the gain in iteration counts is significant (82 vs 13 iterations).

Throughout the entire simulation (and most likely after the 10000 steps shown here) $n_{t,0}$ follows $n_{t,t}$ very closely. This gives us the chance to perform the ILU factorization at a low frequency so that the long time required to compute it will not be noticable over the entire simulation period. As expected, there is a clear increase in the number of excessive iterations when we lower the QEq tolerance to $1e-10$ but this increase is quite acceptable and its side-effects might be offset with a clever choice of c.

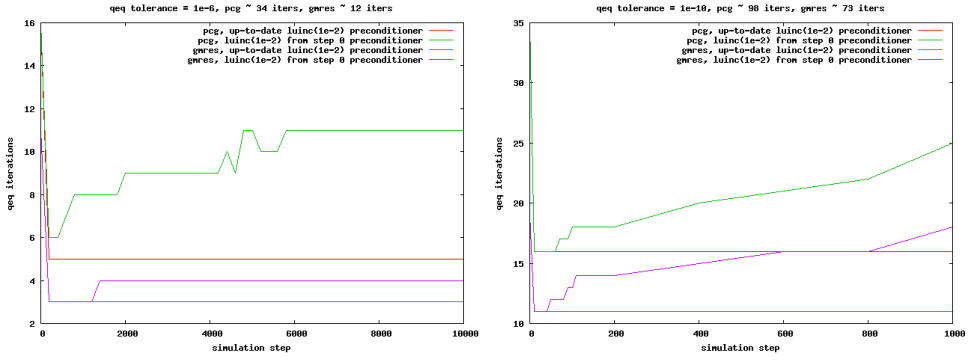


(a) QEq tolerance = $1e-6$ (b) QEq tolerance = $1e-10$

FIG. 5.1. $n_{t,0}$, $n_{t,t}$ *values of* PCG *and* GMRES *solvers for the 3712 atom PETN crystal. Experiments are performed in* Matlab *7 for data obtained from every 200 steps of a 10000 step long NVE simulation with dt*=0.0625 *fs. Temperature fluctuates between* 350 *K and* 400 *K. Performance of the solvers using diagonal preconditioners are given at the title of each figure for comparison purposes.*

We have performed the same type of analysis on a 6540 atom bulk water system. Characteristics of bulk water is quite different from those of a PETN crystal. This is clearly reflected in figures 5.2(a), 5.2(b) where we plot the values of $n_{t,0}$ and $n_{t,t}$ during the course of a 10000 steps NVE simulation. As we can see, excessive iterations appear much more quickly and their number increases more by time. Therefore we would have to perform an ILU factorization more frequently while working with a water system compared to a PETN crystal and the performance degradation due to ILU factorizations would be more apparent. But comparing the number of iterations required by solvers using a diagonal preconditioner, we can easily

say that we would still benefit a lot from using the algorithm 1 for bulk water, too. For instance, using *PCG* at QEq tolerance $1e-6$ and assuming $c = 2$, ns_{t_0} would be approximately 1800 steps and we would perform only 8 iterations at each timestep with the new algorithm compared to the 34 iterations required by the *PCG* method with a diagonal preconditioner. Using *GMRES* at QEq tolerance $1e-10$ and assuming $c = 3$, ns_{t_0} would be approximately 200 steps and we would perform only 8 iterations at each timestep with the new algorithm compared to the 73 iterations required by the *GMRES* method with a diagonal preconditioner.



(a) QEq tolerance = $1e-6$ (b) QEq tolerance = $1e-10$

FIG. 5.2. $n_{t,0}$, $n_{t,t}$ *values of* PCG *and* GMRES *solvers for the* 6540 *atom bulk water system. Experiments are performed in* Matlab 7.6 *for data obtained from every* 10 *steps of the initial* 200 *steps and then every* 200 *steps afterwards. The entire simulation was a* 10000 *step long NVE simulation with dt*=0.25 *fs. Temperature fluctuates around* 200 *K.*

6. Future Research. All results we have shown in the experiments section come from moderate sized systems. To be able to work with large scale systems, a parallel implementation is a must. However, the parallelization of *ReaxFF* introduces additional problems regarding the QEq solvers. First of all, we have to switch from *GMRES* to *PCG* even though the former one is a much better solver for the QEq problem as shown in our experiments in section 5. This decision was made in the light of the fact that coming up with a scalable parallel implementation of the *GMRES* algorithm is not quite possible, on the other hand *PCG* algorithm can be made parallel very easily.

Still, the parallel version of *PCG* requires 4 global communication operations (which can be reduced to 3 by cleverly checking for the stopping criterion) at each iteration and this situation can render the parallel *ReaxFF* code unscalable even if all other components are scalable. QEq calculations already start dominating the total computation time in the serial version as we decrease the tolerance, see section 3.1. Noting that scalability issues arise in solving the QEq problem in a parallel setting, it is evident that the dominance of QEq calculations will be even more apparent in a parallel implementation of *ReaxFF*.

All experiments we have presented so far were on small systems and clearly we can solve the "QEq dominance" problem outlined in section 3.1 by using algorithm 1 for those systems. However, as we have mentioned before, ILU algorithm is not really scalable since it is a Gaussian elimination in essence. There are parallel implementations for the ILU factorization (see for example [2]) but they do not really suit our needs. These algorithms are scalable upto a point but they certainly do not scale upto billions of unknowns which is the point that we ultimately want to reach.

Therefore we apparently need a more clever approach for solving the scalability problem. A straightforward extension of our new algorithm would be to let every processor perform

an ILU factorization within the diagonal block it owns. At the end, ILU factorization is an approximation to the actual LU factors, and letting each processor perform ILU factorization independently may still produce good preconditioners. We are also planning to try a flavor of the SPIKE algorithm which uses ILU preconditioning within the diagonal blocks while treating the off-diagonals specially.

7. Conclusions. *ReaxFF* is a force field developed for bridging the gap between *ab-initio* systems and classical MD methods. What makes *ReaxFF* an important method is that it makes the study of large scale reactive systems possible even on today's common computing platforms. However, the charge equilibration method that *ReaxFF* depends for accurately computing the partial charge distribution on atoms at any timestep may degrade the performance of *ReaxFF* simulations and render the method unscalable as we have discussed. This is due to the enormous number of CPU cycles required for solving the linear systems associated with the QEq problem when good solvers are not used. In a parallel setting, large number of global communications are also required by the same QEq solvers making the scalability problem even worse.

So we propose an ILU preconditioner based algorithm for solving the QEq problems for moderate sized systems (around 10000 atoms) on a single processor. This new algorithm outperforms Krylov subspace methods using a diagonal preconditioner in terms of both iteration counts and running times. The expense associated with computing the ILU factorization at each timestep is avoided by cleverly making use of the context of the QEq problem that we are dealing with. The observation that the structure of the coefficient matrix in our QEq formulation does not change drastically between subsequent timesteps lets us use the preconditioner computed at a timestep quite effectively for a number following steps, too. Our experiments on various systems confirm that enormous gains can be achieved by this new scheme on quite different systems.

We anticipate that the scalability problems associated with the ILU factorization algorithm can be overcome by using a simplified ILU factorization at each node independently or by making use of the SPIKE algorithm. In this way, the method we have presented in this paper can be generalized to deal with much larger systems, too.

REFERENCES

[1] T. A. Halgren and W. Damm, *Polarizable force fields*, Current Opinion in Structural Biology, 11 (2001), pp. 236–242.

[2] D. Hysom and A. Pothen, *A scalable parallel algorithm for incomplete factor preconditioning*, SIAM J. Sci. Comput., 22 (2001), pp. 2194–2215.

[3] A. Nakano, *Parallel multilevel preconditioned conjugate-gradient approach to variable-charge molecular dynamics*, Computer Physics Communications, 104 (1997), pp. 59–69.

[4] S. J. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, J Comp Phys, 117 (1995), pp. 1–19.

[5] S. J. Plimpton, P. Crozier, and A. Thompson, *Lammps users manual*. `http://lammps.sandia.gov/doc/Manual.html`.

[6] A. K. Rappe and W. A. G. III, *Charge equilibration for molecular dynamics simulation*, J Phys Chem, 95 (1991), pp. 3358–3363.

[7] A. C. T. van Duin, S. Dasgupta, F. Lorant, and W. A. G. III, *Reaxff: A reactive force field for hydrocarbons*, J Phys Chem A, 105 (2001), pp. 9396–9409.

A STUDY OF MULTILEVEL ILU TECHNIQUES FOR CIRCUIT SIMULATION

ELLEN C. DURANT* AND HEIDI K. THORNQUIST†

Abstract. Numerical linear algebra is at the heart of scientific computing. In large-scale simulations, the linear solvers often account for more than 80% of computational time. For large, sparse problems iterative methods are essential and preconditioning is key to iterative solver performance. Although a number of general-purpose preconditioners perform well on discretized partial differential equations, they struggle with the differential algebraic equations generated during circuit simulation. Here, we investigate various multilevel incomplete LU techniques and compare them to classic incomplete LU methods for preconditioning circuit matrices.

1. Introduction. While advances in manufacturing enable the fabrication of integrated circuits containing hundreds of millions of devices, the time-sensitive modeling and simulation necessary to design these circuits pose a significant computational challenge. Currently available Electrical Design Automation (EDA) tools such as Xyce, a highly parallel circuit simulator developed at Sandia National Laboratories [1], often struggle with these complex simulations. The principal reason for this is the time required for the non-linear solver to compute the solutions of large linearized systems. To address this problem we investigate more effective preconditioners for circuit matrices.

Iterative methods play a crucial role in solving large, sparse problems. Solver performance depends on intelligent preconditioning. Several general-purpose preconditioners perform well on discretized partial differential equations; however, specialized preconditioners are necessary for solving the differential algebraic equations generated through circuit simulation. This is due to the conditioning and highly heterogeneous structure of circuit matrices. It has been documented that preconditioners based on distributed Schur complements [3][4] and multilevel incomplete LU (ILU) factorizations [12][7] are effective on this class of matrices.

Here we investigate the multilevel ILU preconditioners generated by the Algebraic Recursive Multilevel Solvers (ARMS) [11] package and ILU++ [6] and compare them with a classic ILUT preconditioner to determine their effectiveness on circuit simulation matrices. These preconditioners will be used with the generalized minimal residual (GMRES)[10] and the stabilized bi-conjugate gradient (Bi-CGSTAB)[13] methods.

In this paper, except when specified otherwise, upper case letters (A, B, etc.) denote matrices and lower case letters (x, y, etc.) vectors. Transpose is denoted by A^T. We use MATLAB notation to refer to elements, rows, and columns of matrices, respectively, $A(i, j) = a_{i,j}$ the element of A at row i, column j; $A(i, :)$ is the entire row i of A; similarly, $A(:, j)$ is the entire column j of A.

2. Preconditioners. Preconditioners transform linear systems into equivalent systems with drastically improved properties for solution by an iterative method. Consider the original linear system

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$. A preconditioner, $P \in \mathbb{R}^{n\times n}$, seeks to reduce the condition number of A by its application on the left,

$$P^{-1}Ax = P^{-1}b, \tag{2.2}$$

*Texas Tech University, Dept. of Mathematics & Statistics, edurant@ttu.edu
†Electrical and Microsystems Modeling Dept., Sandia National Laboratories, hkthorn@sandia.gov

right,

$$AP^{-1}Px = b, \tag{2.3}$$

or both left and right (split),

$$P_L^{-1}AP_R^{-1}P_Rx = P_L^{-1}b. \tag{2.4}$$

A variety of different methods can be used to compute preconditioners.

Algebraic preconditioning methods, which include the ILU factorizations we will consider in this paper, use the entries and underlying graph structure of A to generate a preconditioner. The graph structure of a matrix is vital to the generation of an efficient preconditioner, so to improve this structure permutations are often applied. These permutations seek to improve diagonal dominance, remove dense rows and columns, or reduce fill in from the incomplete factorization of the original matrix. For robustness, row and column scalings are also used in generating preconditioners. Permutations and scalings, combined, can provide a matrix that is easier to factor than the original, resulting in a more efficient and robust approximation. It is necessary to acknowledge the separation between the permutations and scalings from the factorizations, because certain types of permutations and scalings work well on certain classes of problems. The fact that the preconditioner is a composition of permutations and factorizations, $P = P_1 \cdot P_2 \cdot \ldots \cdot P_n, n \in \mathbb{Z}$, gives flexibility to the algebraic methods.

2.1. Preprocessing. Preprocessing refers to the analysis of the original matrix that leads to the generation of permutations and scalings that are combined with factorizations to give the resulting preconditioner. Preprocessing techniques can be structure based, requiring only the graph connectivity, or value based, requiring the matrix entries. For instance, normalizing the rows and/or columns of a matrix is a value based preprocessing technique. Other examples of common preprocessing techniques that will be discussed in this section are singleton filtering, I-matrix reordering, reverse Cuthill-McKee (RCM) reordering, and PQ reordering. It will be noted in this discussion which preprocessing techniques are structure based and which are value based because the structure based techniques are preferable, as they can be reused throughout a circuit simulation.

2.1.1. Singleton Filtering. A singleton row (or column) is a dense row (or column) that corresponds to a column (or row) with exactly one non-zero entry. Such rows and columns are frequently found in circuit simulation matrices, because power supply and ground nodes are widely connected. Singleton filtering eliminates these rows and columns. Such filtering is structure based and is sometimes required for generating effective preconditioners, especially in parallel.

2.1.2. I-Matrix Reordering. I-matrix preprocessing [7] generates a non-sym-metric permutation of the original matrix that results in maximal diagonal dominance. An I-matrix, $A \in \mathbb{R}^{n\times n}$, has the following properties:

$$\begin{array}{rcll} a_{i,i} & = & 1, & \forall i \in n \\ a_{i,j} & \leq & 1, & \forall i, j \in n, i \neq j. \end{array}$$

The goal of I-matrix reordering is to maximize the product of the diagonal entries of the matrix. To compute the I-matrix reordering, we solve a multiplicative maximization problem via an additive minimization problem by applying the negative logarithm to the absolute value of the coefficient matrix. This preprocessing method is value based.

2.1.3. Reverse Cuthill-McKee Reordering. The goal of the reverse Cuthill-McKee (RCM) [5] algorithm is bandwidth reduction. It uses a breadth-first search (BFS) of the associated graph to reorder a matrix by grouping connected nodes together. The algorithm will systematically find all the nodes connected to a root node. An example of the algorithm is illustrated in figure 2.1. The nodes are numbered in the order that each node is found. In the example, the algorithm starts with the root node, 1, and searches for its neighbors—nodes 2, 3 and 4 in (2.1). This search is repeated on the neighbor nodes (2, 3 and 4), continues until all nodes connected to node 1 have been found.

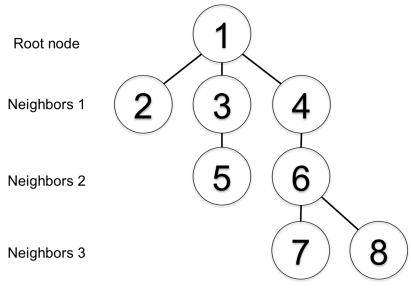


FIG. 2.1. *Breadth-first Search Illustration*

When we preprocess A with RCM, whenever the algorithm has exhausted all possible neighbors to the root node, we permute the corresponding rows and columns so that they are rearranged together in the matrix. This procedure is usually structure based and generates a symmetric permutation that can improve the numerical stability of the ILU factorization.

2.1.4. PQ Reordering. PQ reordering is a non-symmetric permutation algorithm that is both structure and valued based. Unlike I-matrix processing, PQ accounts for structural, in addition to numerical, diagonal dominance. This is accomplished by computing a weight vector:

$$w(i) = \frac{|\max(A(i,i))|}{\|A(i,:)\|} \times \frac{1}{\mathrm{nz}(A(i,:))} \forall i \in n, \tag{2.5}$$

where $A \in \mathbb{R}^{n \times n}$ and $\mathrm{nz}(A(i,:))$ denotes the number of non-zeros in row i of A. Using this weight vector, the algorithm permutes the rows of A to have decreasing values of w and permutes the columns of A to have the dominant elements on the diagonal.

2.2. Incomplete LU Factorization. ILU factorizations are considered to be excellent all-purpose preconditioners [8]. ILU can be tailored to specific computational issues, such as memory limitations, by implementing a drop tolerance (ILUT) or specifying an allowed fill-in level (ILU(k)). ILUT can be implemented with diagonal modification, which replaces zeros on the diagonal with a local drop tolerance, $droptol{*}A(:, j)$, where j denotes the column index of the zero on the diagonal. This modification makes the overall preconditioner more robust. However, ILU has its limitations.

2.3. Multilevel ILU. As systems become larger and more complex, ILU becomes less practical, because it factors the entire matrix A. This may result in too crude of a factorization. That is, as A gets larger, the residual $R = LU - A$ becomes exponentially larger, because ILU drops too many elements during the factorization.

Multilevel ILU alleviates this issue by first breaking A into block form:

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix}. \tag{2.6}$$

Then it allows dropping to occur when computing the Schur complement, block C. Once blocks B, F, E, and C are determined, we calculate block B's LDU factorization, which results in a complete multilevel LDU factorization:

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ E_B & I \end{pmatrix} \times \begin{pmatrix} D_B & 0 \\ 0 & S \end{pmatrix} \times \begin{pmatrix} U_B & F_B \\ 0 & I \end{pmatrix}, \tag{2.7}$$

where L_B, U_B and D_B are, respectively, the lower, upper, and diagonal matrices from the LDU factorization, $E_B = EU_B^{-1}D_B^{-1}$, $F_B = D_B^{-1}L_B^{-1}F$, and the Schur complement is $S = C - E_B D_B F_B$. Next, we let $A_2 = S$ and repeat the process at the next level with $A = A_2$. This procedure continues until the maximum level is reached or until S is determined to have sufficient properties for completely factoring the Schur complement.

The multilevel LDU factorization offers a framework in which to explore a number of variations of preconditioners, such as the multilevel ILU factorization. Like ILU, the multilevel ILU factorization drops elements in order to keep matrices sparse. However, while an ILU factorization may be too coarse to reach a solution, a multilevel ILU factorization may be sufficient. That is because a multilevel ILU factorization can flexibly choose the type of permutations and dropping strategy for each level, giving it the ability to balance quality, computational cost, and robustness of the preconditioner. For our experiments we considered the multilevel ILU factorizations provided by the ARMS and ILU++ software packages.

2.3.1. Algebraic Recursive Multilevel Solver. ARMS is a software package, written for MATLAB, that generates two multilevel ILU preconditioners: ARMS2 and ARMS-C. ARMS2 takes the matrix A_l, where l denotes the level, and uses a symmetric permutation matrix, P_l, to reorder A_l as follows:

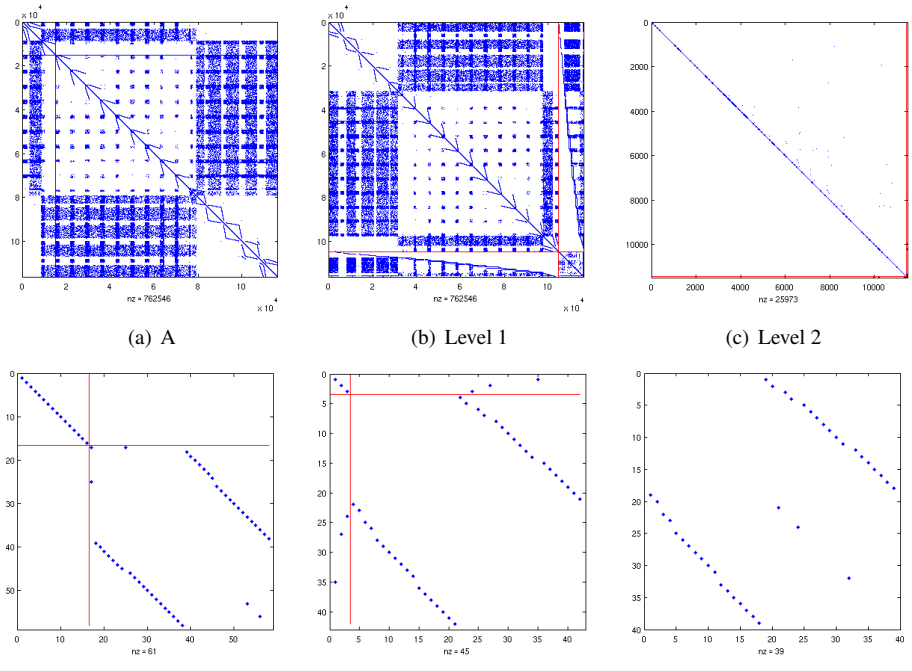
$$P_l A_l P_l^T = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix}, \tag{2.8}$$

where $C_l = A_{l+1}$ represents the Schur complement. The algorithm to determine the blocks B_l, F_l, E_l, and C_l utilizes an independent set reordering algorithm [9], inspired by a multigrid technique that separates unknowns into two sets: coarse and fine [12]. In both ARMS2 and ARMS-C, we group together independent sets and permute these sets to block B, similar to creating a coarse set, leaving the leftover points to a fine set for further processing.

The permutation algorithm in the MATLAB version of ARMS2 begins by computing weights, w, for each row of A,

$$w(i) = \frac{|A(i,i)|}{\|A(i,:)\|_1}. \tag{2.9}$$

The size of block B is determined by the number of normalized weights, $w/\max(w)$, greater than a relative tolerance. Whenever a weight is greater than the relative tolerance, that row is considered to have sufficient diagonal dominance. Then, the diagonally dominant rows are permuted using a modification of the RCM algorithm (see section 2.1.3). The leftover rows and columns are moved to blocks F_l, E_l, and C_l. The procedure then begins again, letting $A_{l+1} = C_l$ until the maximum number of levels is reached, or until the Schur complement has sufficient properties for solution. The latter stopping criterion occurs when, at the next level,



(a) A (b) Level 1 (c) Level 2

(d) Level 3 (e) Level 4 (f) Schur Complement

FIG. 2.2. *Matrix structure plots from ARMS2 preconditioner. Figure 2.2(a) represents the original circuit matrix. Figures 2.2(b) through 2.2(e) show how each level is broken down into the block form, as in equation 2.8. The red lines in the figures separate the blocks B_l, F_l, E_l and C_l. Figure 2.2(f) is the Schur complement.*

all the elements in A_{l+1} get permuted to block B_{l+1} (in other words, whenever C_{l+1} is empty). Figure 2.2 shows the application of ARMS2 to a test matrix from Xyce.

By comparison, ARMS-C implements a non-symmetric permutation of A_l:

$$P_l A_l Q_l^T = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix}, \tag{2.10}$$

where P_l permutes the rows, Q_l^T permutes the columns and $C_l = A_{l+1}$ is the Schur complement. ARMS-C uses a PQ-type reordering to determine the blocks B_l, F_l, E_l and C_l.

The permutation algorithm implemented in the MATLAB version of ARMS-C uses a variation of the PQ reordering described in section 2.1.4. The PQ-type reordering algorithm in ARMS-C begins by checking to see whether or not the the largest value of each row is greater than a relative tolerance, τ. The row and column indices of elements that pass this test are stored in vectors, *row* and *col*. Next, we pass the vectors *row* and *col* to calculate a weight vector, w, accounting for both numerical and structural diagonal dominance, as explained in section 2.1.4. The MATLAB implementation of this algorithm is:

$count = 1$

For $i = 1, 2, ..., n$

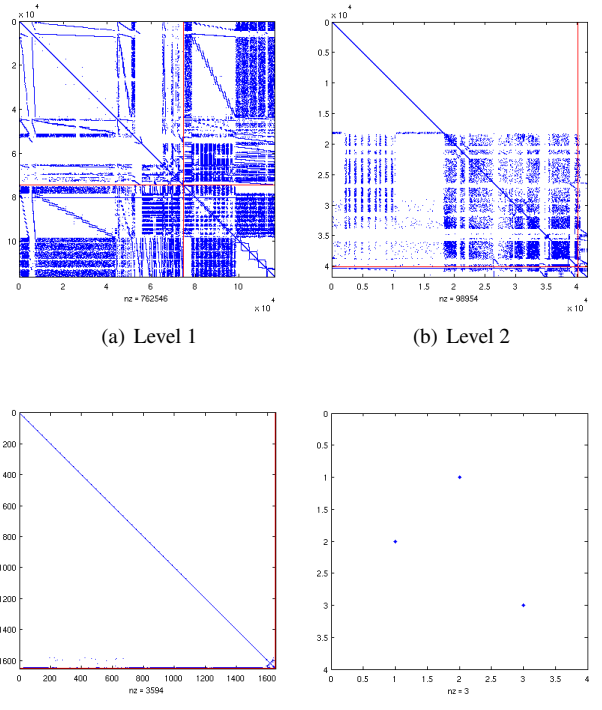
  If $|\max(A(i,:)| = |a_{i,j}| > \tau$ then $row(count) = i, col(count) = j$

  $count = count + 1$

For $k = 1, 2, ..., count$

$$w(i) = \frac{|a_{row(k),col(k)}|}{\|A(row(k),:)\|_1} \times \frac{1}{\text{nz}(A(i,:))}, \tag{2.11}$$

We then reorder w in descending order, taking care to also reorder *row* and *col* to reflect the new order of w. The PQ-type reordering then permutes the rows and columns of A_l that are stored in w to block B_l by creating independent sets and accounting for diagonal dominance with ordered vectors *row* and *col*, as described in [11]. The reordering moves rejected rows and columns to blocks F_l, E_l and C_l. This process iterates, letting $A_{l+1} = C_l$, until the maximum number of levels is reached, or the Schur complement has sufficient properties for solution. Figure 2.3 shows the application of ARMS-C to the same test matrix from Xyce as in Figure 2.2(a).



(a) Level 1 (b) Level 2

(c) Level 3 (d) Schur Complement

FIG. 2.3. *Matrix structure plots from ARMS-C preconditioner using same circuit matrix in Figure 2.2(a). Figures 2.3(a) through 2.3(c) show how each level is broken down into the block form, as per Equation 2.10. The red lines in the figures show the separation the blocks B_l, F_l, E_l and C_l. Figure 2.3(d) is the Schur complement.*

2.3.2. ILU++. ILU++ is a software package, written in C++, that provides a framework for generating multilevel ILU preconditioners. Compared to ARMS, ILU++ offers many options for the ILU factorization, level termination strategy, preprocessing, and drop tolerances [6]. For example, Crout ILU (ILUC) and ILUC with dual-pivoting (ILUCDP) is available. Crout ILU factors a matrix $A \approx LU$ so that U is unit upper triangular. When dual-pivoting is allowed, rows and columns can be permuted independently—to encourage sparsity and to avoid small pivots. Additionally, in ILU++ several level termination options are accessible, such as ending the level once a maximum amount of fill-in has been reached or stopping whenever a pivot is too small. Furthermore, ILU++ offers multiple preprocessing techniques, such as normalization of rows and columns, PQ reordering and I-matrix reordering. Finally, drop tolerances can be easily edited by the user, like those specifying fill-in for the overall preconditioner and the Schur complement or choosing a pivoting tolerance. Where ILU++ excels in flexibility, ARMS only offers ILUT, with either RCM reordering or

PQ reordering, and limited customization features.

3. Results. In this section we will present the performance results from using the multilevel preconditioners provided by ARMS and ILU++ with GMRES to solve linear systems generated by the Xyce circuit simulator. We assess the performance of each preconditioner using GMRES iterations, instead of computational time, for fairness since ARMS and ILU++ are written in two distinctly different languages, MATLAB and C++, respectively. The matrices under consideration come from a circuit that Xyce has trouble simulating using iterative solvers.

To give our results some context, we will offer the GMRES iterations necessary for Xyce to solve each linear system, using the default solver configuration, as well. By default, Xyce uses an Additive Schwartz preconditioner, which is a one-level overlapping domain decomposition preconditioner. With these types of preconditioners the unknowns are partitioned, usually across processors, and the matrix is divided up in a compatible manner. The local matrix is one where the unknowns designated to a particular processor match up with the rows of the matrix also designated to that processor. Xyce uses no overlapping, by default, resulting in a block diagonal preconditioner. Furthermore, for the local matrix solve, Xyce uses an incomplete LU factorization (ILUT).

3.1. Voter Circuit. The CircuitSim90 circuit simulator benchmark suite is one of the very few defacto standard circuit simulator benchmark suites that is in the public domain [2]. The Voter circuit is one of the larger Level 2 MOSFET benchmark tests, generating a

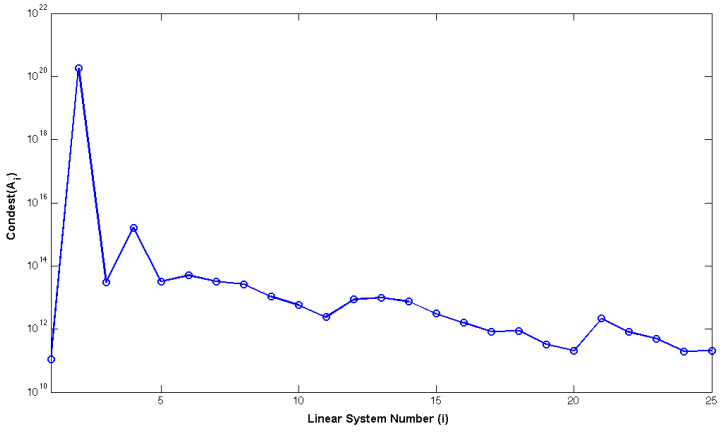


FIG. 3.1. *Condition estimates of the first 25 Voter matrices generated by Xyce*

linear system with 10217 unknowns. To simulate this circuit through a full transient takes 4 hours with a serial build of Xyce, which uses direct solvers. Xyce has yet to complete a full transient simulation of this circuit using iterative solvers.

The most dominant issue with the linear systems generated during the simulation of the Voter circuit is their ill-conditioning. Figure 3.1 shows the estimated condition numbers of the first 25 matrices generated by Xyce during the simulation of the Voter circuit. While one of the goals of preconditioning is to improve the conditioning of the original matrix to accelerate convergence of the iterative method, not all preconditioners are successful.

3.2. Xyce Performance. The default preconditioner in Xyce is not always effective in solving these linear systems. Figure 3.2 shows (top) the number of GMRES iterations, maximum 500, required to achieve the requested relative residual, 10^{-9}, and (bottom) what relative residual was achieved. In 8 of the first 25 linear solves, GMRES was not able to reach 10^{-9}

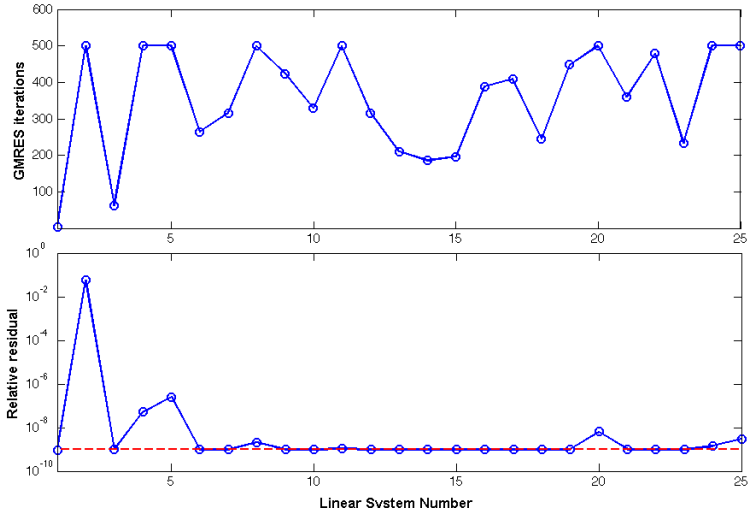


FIG. 3.2. *Xyce iterative solver behavior for the first 25 Voter matrices*

in 500 iterations. In fact, when GMRES did converge, it usually required more than 200 iterations. The default preconditioner generated by Xyce is clearly not consistently effective on these linear systems.

3.3. ARMS Performance. ARMS is a software package, written for MATLAB, that generates two multilevel ILU preconditioners: ARMS2 and ARMS-C. These two preconditioners generate a symmetric and non-symmetric permutation, respectively. The Voter matrices are non-symmetric, so the ARMS-C preconditioner may be more effective in solving these linear systems. However, in the course of our studies, we have found the implementation of the ARMS-C preconditioner provided by the ARMS software package is not robust enough to handle these matrices. We routinely received errors from the code, such as those shown in Figure 3.3. So, regrettably, we will be reporting results for GMRES preconditioned with the ARMS2 preconditioner.

```
??? Improper assignment with rectangular empty matrix.

Error in ==> presel at 17
    jcor(i) = row(k);

Error in ==> PQ at 8
 [icor, jcor, count] = presel(A, tol) ;

Error in ==> armsC at 20
    [ip, iq, nB] = PQ(S,tolInd) ;

Error in ==> demoArmsVoter at 172
PRE = armsC(A,ARMSopt) ;
```

FIG. 3.3. *ARMS-C error output*

ARMS2 offers a minimal set of options for the user to specify. The preconditioning options mostly pertain to dropping tolerances, levels of fill, and maximum number of levels. The GMRES solver options are for specifying the size of the Krylov subspace, maximum number of restarts, requested relative residual tolerance, and maximum number of iterations. The results presented here are obtained when the ILU tolerance for B (see Equation 2.7)

and the Schur complement (`ilutolB` and `ilutolS`, respectively) is changed to 10^{-3} and the maximum number of levels (`nlev`) is set to 3. For the iterative solver, the size of the Krylov subspace (`k`) and maximum iterations (`maxits`) was set to 500, while the maximum number of restarts (`maxrestarts`) was set to 0. The convergence tolerance (`tol`) was set to 10^{-9}. These settings were chosen to allow comparison with Xyce's default preconditioner.

Considering the fact that it uses a symmetric permutation, it is promising that the ARMS2 preconditioner is quite effective in solving a majority of these linear systems. Figure 3.4 shows that it only failed to obtain the requested residual of 10^{-9} in 2 of the first 25 linear

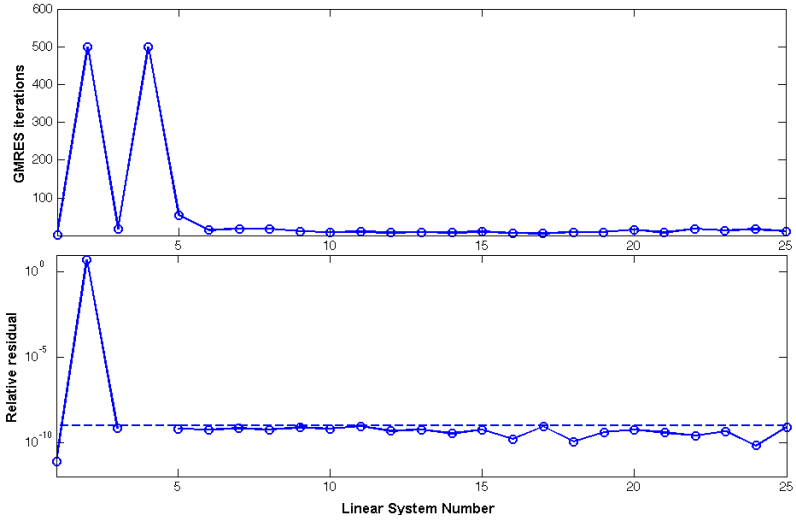


FIG. 3.4. *ARMS2 preconditioned GMRES behavior for the first 25 Voter matrices*

solves and, when it did converge, it usually required less than 25 iterations. However, when it did fail, the resulting residuals were larger than with Xyce's default preconditioner and in one case GMRES returned NaNs. The two failures were on the second and fourth linear systems, which have the most ill-conditioned (see Figure 3.1) matrices of the first 25. So, given the fact that the condition number of the Voter matrix improves as the simulation proceeds, the ARMS2 preconditioner will, overall, be more effective in accelerating the convergence of GMRES than Xyce's default preconditioner.

3.4. ILU++ Performance. ILU++ provides a framework for generating multilevel ILU preconditioners. Compared to ARMS, ILU++ offers many more options for the ILU factorization, level termination strategy, preprocessing, and drop tolerances. Performing a study with all of the possible combinations offered by this software package is beyond the scope of this paper. Instead, we chose three recommended default configurations that result in preconditioners that are somewhat comparable to ARMS2, ARMS-C, and Xyce's default preconditioner. The first configuration (A) computes an I-Matrix reordering (see section 2.1.2) before factorizing the matrix using full pivoting (row and column). The second configuration (B) normalizes both rows and columns, then applies PQ preprocessing (see section 2.1.4) before factorizing the matrix using full pivoting. The third configuration (C) is the same as the first, but it performs an additional symmetric reordering to produce an initial diagonally dominant submatrix. All three configurations use a drop tolerance of 10^{-3} and relative residual tolerance for GMRES of 10^{-9}.

The three multilevel preconditioners tested from ILU++ were very effective in solving a majority of the linear systems. The least effective of the three preconditioners, as shown

in Figure 3.5, is the second configuration (B) that uses the PQ preprocessing. This preconditioner fails to solve 3 of the first 25 linear systems and, in all of those failures, the GMRES solver returned NaNs. This is a noteworthy result since the ARMS-C preconditioner we were having trouble configuring uses PQ reordering. We would expect to see similar behavior from the ARMS-C preconditioner on these linear systems, given a more robust implementation. When GMRES does converge using this second configuration, it usually required less than 50 iterations.

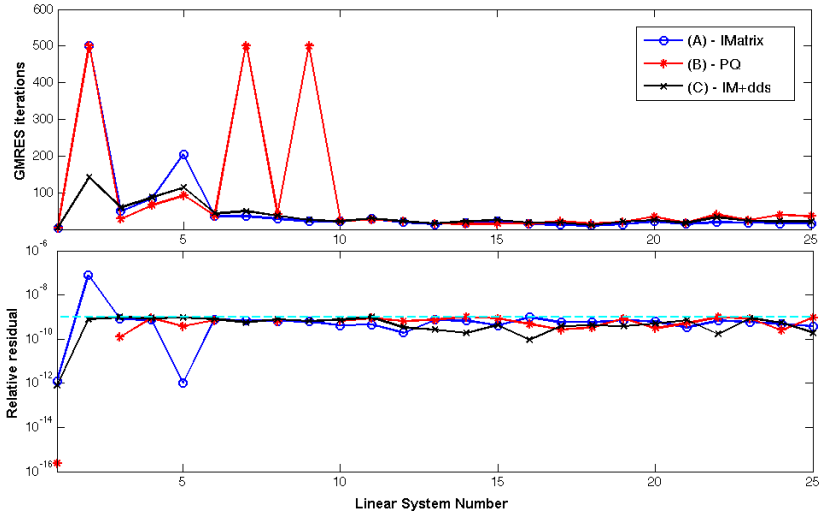


FIG. 3.5. *ILU++ preconditioned GMRES behavior for the first 25 Voter matrices*

The most effective of all the preconditioners tested on these linear systems from either the ARMS or ILU++ software packages use I-Matrix reordering. The preconditioner generated using the first configuration fails only once, on the second linear system. The third configuration, that combines I-Matrix reordering with an additional reordering for diagonal dominance, does not fail to solve any of the first 25 linear systems. In fact, even the most ill-conditioned matrices required less than 150 iterations and all the others required much less than 50.

4. Conclusion. Multilevel incomplete LU preconditioners have shown promise for accelerating the convergence of GMRES on linear systems generated through circuit simulation. This paper presents a small subset of the available multilevel ILU preconditioners and compared them on a sequence of linear systems generated by Xyce for a circuit it has not yet been able to simulate using iterative methods. The results obtained give unequivocal evidence that Xyce would benefit from the integration of multilevel ILU preconditioners. In particular, future directions of this work should include the study of I-Matrix reordering, which uses maximum weighted matching. It should be noted that both ARMS and ILU++ are serial software packages and Xyce will require a parallel implementation of a multilevel ILU preconditioner. While only a few of the multitude of options were presented here, our experience from this study makes it clear that finding the best parameters for preconditioning circuit simulation matrices will be difficult and will require time and experience.

REFERENCES

[1] *Xyce parallel circuit simulator*. http://xyce.sandia.gov.

[2] J. Barby and R. Guindi, *CircuitSim93: A circuit simulator benchmarking methodology case study*, Proc. IEEE Int., ASIC Conf., (1993), pp. 531–535.

[3] A. Basermann, U. Jaekel, and K. Hachiy, *Preconditioning parallel sparse iterative solvers for circuit simulation*, in Proceedings of the 8th SIAM Proceedings on Applied Linear Algebra, Williamsburg VA, July 15-19 2003.

[4] A. Basermann, U. Jaekel, and M. Nordhausen, *Parallel iterative solvers for sparse linear systems in circuit simulation*, Future Generation Computer Systems, 21 (2005), pp. 1275–1284.

[5] W. Liu and A. Sherman, *Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices*, SIAM J. Numerical Analysis, 13 (1976), pp. 198–213.

[6] J. Mayer, *ILU++*. http://iamlasun8.mathematik.uni-karlsruhe.de/~ae04/iluplusplus.html.

[7] ———, *A numerical evaluation of preprocessing and ILU-type preconditioners for the solution of unsymmetric sparse linear systems using iterative methods*, ACM Transactions on Mathematical Software, 36 (2009), pp. 1:1–1:26.

[8] Y. Saad, *ILUT a Dual Threshold Incomplete LU Factorization*, Numerical Linear Algebra with Applications, 1 (1994), pp. 387–402.

[9] ———, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Scientific Computing, 17 (1996), pp. 830–847.

[10] ———, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, second ed., 2003.

[11] ———, *Multilevel ILU with reorderings for diagonal dominance*, SIAM J. Scientific Computing, 27 (2006), pp. 1032–1057.

[12] Y. Saad and B. Suchomel, *ARMS: An Algebraic Recursive Multilevel Solver for general sparse linear systems*, Tech. Rep. UMSI 99/107, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1999.

[13] H. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Scientific and Statistical Computing, 13 (1992), pp. 634–644.

COMPARISONS BETWEEN FINITE ELEMENT AND FC-AD METHODS WITH APPLICATIONS TO DRUG DELIVERY

CATHERINE E. BENI*, OSCAR P. BRUNO *, PAVEL B. BOCHEV†, DENIS RIDZAL†, AND KARA J. PETERSON†

Abstract. In this paper we compare two different methods of solving the basic equations that arise from the problem of magnetic drug delivery: Finite Element methods and a new method, the so-called Fourier Continuation-Alternating Directions (FC-AD) algorithm. Although there was not enough time to obtain numerical results, we discuss the analytical properties of the two methods.

1. Introduction. The goal of magnetic drug delivery is to direct and confine magnetically-responsive particles coated by or containing therapeutic agents to specific regions in the body by applying external magnetic fields. This is useful for treatment of diseases, including cancer, strokes, and infections, because it concentrates treatments to disease sites, thereby reducing their impact on the rest of the body and in turn allowing for a higher dose of treatment to be applied.

To rationally design a method of confining the magnetically-responsive particles (also referred to as ferrofluid) to a particular region of the body, a mathematical model of how the external magnetic forces will transport the fluid through the bloodstream is required. One such model has been developed by Grief and Anderson [7] and was implemented numerically (in COMSOL) at the University of Maryland [9]. The model characterized by the following hyperbolic convection-diffusion partial differential equation (PDE):

$$\frac{\partial}{\partial t}C(\vec{r},t) = \\ -\nabla\cdot\left[C(\vec{r},t)\ \vec{V}_{\text{blood}}(\vec{r},t) - D(\vec{r})\ \nabla C(\vec{r},t) + k(\vec{r})\ C(\vec{r},t)\nabla\left(|\vec{H}(\vec{r},t)|^2\right)\right], \tag{1.1}$$

where $C(\vec{r},t)$ is the concentration of magnetic particles in the blood, $\vec{V}_{\text{blood}}(\vec{r},t)$ is the blood convection, $D(\vec{r})$ is the diffusion coefficient of particles within the blood stream, $k(\vec{r})$ is the magnetic drift coefficient, and $\vec{H}(\vec{r},t)$ is the externally applied magnetic field intensity. The term $\nabla\left(|\vec{H}(\vec{r},t)|^2\right)$ is referred to as the *control*.

For the preliminary tests of simulating the fluid flow and diffusion rates, an extremely simplified model of the vasculature is used. This allows us to better understand how the flow is affected by the relationships between the magnitude of the magnetic force, the diffusion coefficient of the surrounding tissue and the blood velocity without added complexities. The idealized geometry we use in this paper consists of a lateral cross section of a blood vessel, including the endothelial layer (vessel wall) and some of the surrounding tissue, with a magnet situated below, see figure 1. Each layer has a different diffusion coefficient, with the vessel having the largest value and the endothelial layer having the smallest. The magnetic drift coefficient in each layer is the ratio between the layer's diffusion coefficient and the diffusion coefficient in the vessel. We simplify the model further by setting the control to be constant.

Because of the discontinuity in the diffusion coefficients, when the magnetic forces acting on the particles are strong enough to overcome the blood velocity, a sharp build-up of concentration, also known as a *boundary layer*, appears at the interface between the vessel and the endothelial layer. This provides a challenge for numerical algorithms: for the values of diffusion coefficients used in realistic models of capillaries, the number of grid points

*California Institute of Technology, beni@caltech.edu, bruno@acm.caltech.edu
†Sandia National Laboratories, pbboche@sandia.gov

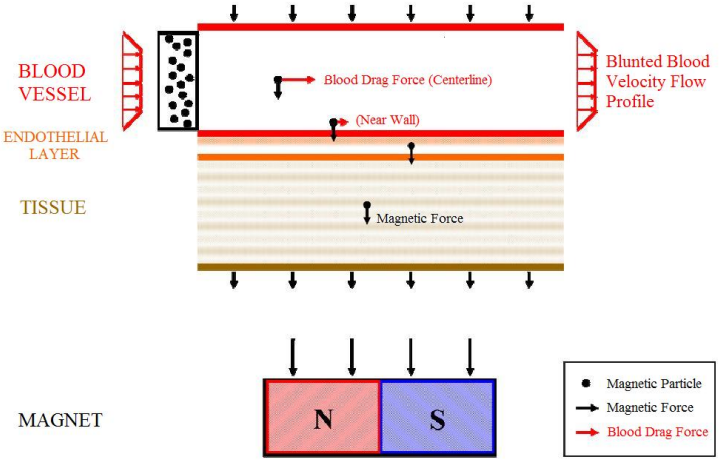


FIG. 1.1. *Vessel geometry used in numerical simulations consisting of the lateral cross section of a blood vessel and some surrounding tissues, including the endothelial layer (vessel wall). The magnet situated below the vessel demonstrates how the external magnetic field affects the fluid flow by generating a downward "velocity".*

needed to accurately resolve the boundary layer is large enough that the computational times required by standard methods make them unfeasible for use when multiple scenarios are being simulated.

In this paper, we solve equation (1.1) with piecewise constant coefficients by using two different numerical algorithms, Fourier Continuation-Alternating Direction (FC-AD) and Finite Element (FEM) methods. The goal of this paper is to compare the performance of the two methods when resolving a boundary layer, with a focus on convergence and accuracy.

2. Finite Element Method. For the purposes of better modeling the basic relationships between the multiple parameters present in equation (1.1), in this paper we consider the simplified equation

$$\begin{aligned} &C_t = D\nabla^2 C - v_x C_x + v_y C_y, && (x,y,t) \in \Omega \times (0,T], \\ &C = 1, && x = x_l,\ y \geq y_v,\ t \in (0,T], \\ &C_x = 0, && x = x_l, x_r,\ y < y_v,\ t \in (0,T], \\ &C_y = 0, && y = y_l, y_r,\ t \in (0,T], \\ &C(x,y,0) = \begin{cases} 1\,; & y \geq y_v \\ 0\,; & y < y_v \end{cases} \end{aligned} \tag{2.1}$$

defined on the domain $\Omega = [x_l, x_r] \times [y_l, y_r]$, where y_v is the location of the interface between the vessel and the endothelial layer. This is derived from (1.1) by noticing that, for the geometry currently being considered, v_{blood}, $D(\vec{r})$ and $k(\vec{r})$ are constant in the x-direction and piecewise constant in the y-direction. It is important to note that while $D(\vec{r})$ and $k(\vec{r})$ are truly piecewise constant everywhere, in the vessel, v_{blood} takes on the form of a blunted, parabolic velocity profile. However, any complications arising from a variable blood velocity term are overcome by a simple discretization of v_{blood}, effectively making it piecewise constant. As stated previously, the control is set to be constant. For notational convenience, we re-label $v_{\text{blood}} = v_x$ and $k(\vec{r})\nabla\left(|\vec{H}(\vec{r},t)|^2\right) = -v_y$.

The Finite Element Method (FEM) [8] is a general technique introduced in the late 50's and early 60's for the numerical solution of differential and integral equations in science

and engineering. The basic idea of any numerical method used for computing the solution of a differential equation is to discretize the given continuous problem (with infinitely many degrees of freedom), thus forming a finite system of equations. In Galerkin methods, the finite element method considered in this paper, a discretization is generated after a reformulation of the given differential equation into its *variational form* is performed. Because a time derivative is present in our PDE, before computing the variational form, we must discretize in time. For this, we use Backward Euler's method:

$$\frac{C^{n+1} - C^n}{\triangle t} = D\nabla^2 C^{n+1} - v_x C_x^{n+1} + v_y C_y^{n+1}.$$

Grouping together like terms, we obtain the implicit scheme

$$C^{n+1} - \triangle t\left(D\nabla^2 C^{n+1} + b \cdot \nabla C^{n+1}\right) = C^n, \tag{2.2}$$

where $b = (-v_x, v_y)$. Because we chose an implicit method, the FEM described below will be unconditionally stable in time.

Consider the Hilbert space $H_0^1(\Omega)$ of test functions

$$H_0^1(\Omega) = \{w \in L_2(\Omega) : w = 0 \text{ on } \Gamma, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y} \in L_2(\Omega)\},$$

where Γ is the boundary of Ω. The variational formulation of (2.1) is given as

$$\text{Find } C \in H_0^1(\Omega) \text{ such that } a(C, w) = f(w) \ \ \forall w \in H_0^1(\Omega), \tag{2.3}$$

where

$$a(C, w) = \int_\Omega \nabla C \cdot \nabla w \, dx + \int_\Omega b \cdot \nabla w \, dx$$
$$f(w) = \int_\Omega C_t w \, dx.$$

Next, a finite-dimensional subspace V_h of $H_0^1(\Omega)$ is constructed through the triangulation of Ω: the division of Ω into a set $T_h = K_1, ..., K_m$ of non-overlapping cells K_i,

$$\Omega = \bigcup_{K \in T_h} K = K_1 \cup K_2 ... \cup K_m,$$

such that no vertex of one cell lies on the edge of another cell. Cells are typically chosen to be triangular although other two-dimensional shapes, such as squares, are also commonly used. For the numerical results presented in this paper, the cells K_i are chosen to be quadratic. Note that, although it is true for our case of a rectangular domain, in general Ω does not necessarily exactly equal its triangulation. Next, we define V_h as

$$V_h = \{w \in H_0^1 : w|_K \text{ is linear for K} \in T_h\}$$

where $w|_K$ denotes the restriction of w to K. By considering the *nodes* $N_i, i = 1, ..., M$ of T_h, we can easily generate a basis $\{\phi_j\}$ for V_h through the following definition:

$$\phi_j(N_i) = \delta_{ij} \equiv \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}, \quad i, j = 1, ..., M.$$

Because the support of ϕ_j is given by the cells with the common node N_j, a function $w \in V_h$ can be represented by the linear combination

$$w(x) = \sum_{j=1}^{M} \eta_j \phi_j(x), \ \eta_j = w(N_j), \ \text{for } x \in \Omega \cup \Gamma.$$

The Galerkin method for the boundary value problem (2.1) is then formulated as

$$\text{Find } C_h \in V_h \text{ such that } a(C_h, w) = f(w) \ \ \forall w \in V_h(\Omega), \tag{2.4}$$

and can be viewed as the projection of the variational form (2.3) to the finite dimensional space C_h. In particular, we have

$$a(C_h, \phi_j) = f(\phi_j).$$

By noting that C_h has the representation

$$C_h = \sum_{i=1}^{M} \xi_i \phi_i(x), \ \xi_i = C_h(x_i),$$

we can re-write (2.4) as

$$\sum_{i=1}^{M} \xi_i a(\phi_i, \phi_j) = f(\phi_j), \tag{2.5}$$

or, more simply, as the linear system of equations:

$$A\xi = b, \ \text{ with } \ a_{ij} = a(\phi_i, \phi_j), \ b_j = f(\phi_j). \tag{2.6}$$

The above system can then be solved for using a variety of numerical methods. It is important to note that due to the small support of each of the basis functions ϕ_j, the matrix A is very sparse and can therefore be inverted efficiently.

In this paper, the Galerkin method described above was implemented in C++ through the use Intrepid, a high-end Finite Element package developed by P.B. Bochev, et al. at Sandia National Laboratories. For the inversion of the linear system shown in (2.6), the direct sparse solver package Amesos was used. The details of the numerical implementation are listed in section 4.

3. FC-AD. The Fourier-Continuation Alternating-Direction (FC-AD) algorithm [3] is a new methodology created by O.P. Bruno and M. Lyon for the numerical solution of PDEs in general spatial domains based on using the well-known Alternating Direction Implicit (ADI) approach in conjuction with the Fourier Continuation (FC) method. Alternating directions algorithms, introduced in [4, 6, 5] have the attractive feature of being capable of yielding unconditional stability at roughly the same computational cost as explicit (conditionally stable) Finite Difference approximations. However, the application of alternating directions methods has been hindered by the fact that they cannot be directly applied to PDEs on general domains without reducing the accuracy near the boundary to first order. FC-AD methods can produce high-order accuracies with unconditionally stable solutions for general geometries in essentially linear time. These features make the FC-AD method well-suited for not only the simple geometry currently being considered, but also for future, more complex models of the vasculature.

In section 3.1 we introduce the alternating direction scheme used in our numerical simulations of (2.1). The FC-based algorithm for the solution of the resulting ordinary differential equations (ODEs), called FC-ODE is presented in section 3.2.

3.1. Alternating Directions Implicit Approach. As in section 2, we again consider the PDE described by equation (2.1). Let t^n and C^n be equal to $n\triangle t$ and $C(x, y, t^n)$ respectively. By discretizing the time derivative through a centered finite difference scheme around $t = (n + \frac{1}{2})\triangle t$, we obtain

$$\frac{C^{n+1} - C^n}{\triangle t} = \frac{D}{2}\nabla^2(C^{n+1} + C^n) - \frac{v_x}{2}\frac{\partial}{\partial x}(C^{n+1} + C^n) + \frac{v_y}{2}\frac{\partial}{\partial y}(C^{n+1} + C^n) + E_1(x, y, \triangle t),$$

where $E_1(x, y, \triangle t) \sim O(\triangle t^2)$ is the remainder. Grouping together the terms for C^n and C^{n+1} in the above equation yields

$$\begin{aligned}\left(1 - \frac{\triangle t}{2}\left(D\nabla^2 - v_x\frac{\partial}{\partial x} + v_y\frac{\partial}{\partial y}\right)\right)C^{n+1} \quad &= \\ \left(1 + \frac{\triangle t}{2}\left(D\nabla^2 - v_x\frac{\partial}{\partial x} + v_y\frac{\partial}{\partial y}\right)\right)C^{n} \quad &+ \triangle t E_1(x, y, \triangle t).\end{aligned} \tag{3.1}$$

Equation (3.1), can then be expressed in the following factored form:

$$\begin{aligned}&\left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial x^2} + \frac{v_x\triangle t}{2}\frac{\partial}{\partial x}\right)\left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial y^2} - \frac{v_y\triangle t}{2}\frac{\partial}{\partial y}\right)C^{n+1} = \\ &\left(1 + \frac{D\triangle t}{2}\frac{\partial^2}{\partial x^2} - \frac{v_x\triangle t}{2}\frac{\partial}{\partial x}\right)\left(1 + \frac{D\triangle t}{2}\frac{\partial^2}{\partial y^2} + \frac{v_y\triangle t}{2}\frac{\partial}{\partial y}\right)C^{n} \\ &+ O(\triangle t^3) + \triangle t E_1(x, y, \triangle t).\end{aligned} \tag{3.2}$$

In order to solve for C^{n+1} in equation (3.2), it is necessary to invert the differential operators $\left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial x^2} + \frac{v_x\triangle t}{2}\frac{\partial}{\partial x}\right)$ and $\left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial y^2} - \frac{v_y\triangle t}{2}\frac{\partial}{\partial y}\right)$. The application of the inverse operators on a function f is equivalent to computing the solutions of the one-dimensional boundary value problem

$$-\alpha u'' - \beta u' + u = f, \;\; u'(x_l) = 0, \;\; u'(x_r) = 0, \tag{3.3}$$

where $\alpha = \frac{D\triangle t}{2}$ and β is either $-\frac{v_x\triangle t}{2}$ or $\frac{v_y\triangle t}{2}$.

By noticing that the differential operators are commutable, we can obtain the following scheme for computing $\tilde{C}^n$, an approxmation of the exact function C^n:

$$\begin{aligned}\left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial x^2} + \frac{v_x\triangle t}{2}\frac{\partial}{\partial x}\right)\tilde{C}^{n+\frac{1}{2}} \quad &= \left(1 + \frac{D\triangle t}{2}\frac{\partial^2}{\partial y^2} + \frac{v_y\triangle t}{2}\frac{\partial}{\partial y}\right)\tilde{C}^{n} \\ \left(1 - \frac{D\triangle t}{2}\frac{\partial^2}{\partial y^2} - \frac{v_y\triangle t}{2}\frac{\partial}{\partial y}\right)\tilde{C}^{n+1} \quad &= \left(1 + \frac{D\triangle t}{2}\frac{\partial^2}{\partial x^2} - \frac{v_x\triangle t}{2}\frac{\partial}{\partial x}\right)\tilde{C}^{n+\frac{1}{2}}.\end{aligned} \tag{3.4}$$

3.2. Fourier Continuation and the FC-ODE algorithm. The last element needed to implement the FC-AD solver is a suitable discrete operator that approximates the inverse of the simple differential operator shown in equation (3.3):

$$S_{\alpha,\beta,x} = \left(1 - \alpha\frac{\partial^2}{\partial x^2} - \beta\frac{\partial}{\partial x}\right)^{-1}. \tag{3.5}$$

Consider a smooth function $f \in C^k[x_l, x_r]$ for some positive integer k or $k = \infty$. We assume that approximate values of the function f are given over a discrete grid $x_j, \; j = 1, ..., n$

contained in $[x_l, x_r]$. For the purposes of explanation, and without loss of generality, we replace the interval $[x_1, x_n]$ with the interval $[0, 1]$ and use the discrete grid

$$x_j = (j-1)h, \quad j = 1, ..., n, \quad h = 1/(n-1); \tag{3.6}$$

The FC-ODE method described in this section applies, after a change of variables, to the general intervals $[x_1, x_n]$ and $[x_l, x_r]$. Because (3.3) is a linear ODE with constant coefficients, we can compute the application of the inverse operator $S_{\alpha,\beta,x}$ to f by approximating f with a Fourier series and performing some simple algebraic operations on its Fourier coefficients. However, in the case that f is not periodic over the interval $[0, 1]$, approximating f with a Fourier series of period 1 gives rise to the Gibbs phenomenon, a well-known ringing effect. The Fourier continuation method is a new method introduced in [1, 2] for the resolution of the Gibbs phenomenon. It relies upon constructing a periodic function in a domain significantly larger than the interval $[x_1, x_n] = [0, 1]$ while still containing the discretization points $x_1, ..., x_n$.

Following the method described in [2, 3], we define a function $f^c(x)$ periodic over a given period b, with $b > 1$, through the series

$$f^c(x) = \sum_{k \in t(M)} a_k e^{\frac{2\pi i}{b} kx}, \tag{3.7}$$

where $t(M) = \{j \in \mathbb{N} : -M/2 + 1 \le j \le M/2\}$ for M even and $t(M) = \{j \in \mathbb{N} : -(M-1)/2 \le j \le (M-1)/2\}$ for M odd. The coefficients a_k are then obtained from the solution of the least-squares system of equations

$$\min_{\{a_k:\ k \in t(M)\}} \sum_{j=0}^{n-1} \left| \sum_{k \in t(M)} a_k e^{\frac{2\pi i}{b} k x_j} - f(x_j) \right|^2 . \tag{3.8}$$

Numerical results [2] have shown that it is advantageous to solve the least-squares system by using Singular Value Decompositions (SVD). In this paper, we refer to the SVD-based method of Fourier continuation as FC(SVD). A typical result produced by our implementation of the FC(SVD) method is shown in figure 3.2.

However, while the $O(n^3)$ computational cost of FC(SVD) is adequate for applications such as the high-order surface representations in [2], it is significantly higher than desirable for use in a PDE solver. A new method, called FC(Gram), based on function matching and Gram polynomials was presented in [3] and is described below.

Consider a pair of functions: the given function $f(x)$ and its translation $f(x - d - 1)$, defined over the intervals $[0, 1]$ and $[1 + d, 2 + d]$ respectively, see figure 3.2. As shown in the figure, we consider small portions taken from the right end of the graph of $f(x)$ and the left end of the graph of $f(x - d - 1)$. The intervals these portions are defined on are $[1 - \delta, 1]$ and $[1 + d, 1 + d + \delta]$ accordingly. The FC(SVD) algorithm can be applied to these two segments to generate a periodic function f_{match}, with periodicity interval $[1 - \delta, 1 + 2d + \delta]$, that simultaneously matches $f(x)$ on the interval $[1 - \delta, 1]$ and $f(x - d - 1)$ on the interval $[1 + d, 1 + d + \delta]$. From [2], we know this approximation is spectrally accurate. The function f_{match}, shown raised for visibility in figure 3.2, can then be used to provide a smooth transition from the right end of $f(x)$ to the left end of $f(x - d - 1)$ as follows:

$$f^{\mathrm{de}}(x) = \begin{cases} f(x), & \text{for } x \in [0, 1] \\ f_{\mathrm{match}}(x), & \text{for } x \in (1, 1 + d] \\ f^{\mathrm{de}}(x + 1 + d) = f^{\mathrm{de}}(x), & \text{for all } x \text{ in } \mathbb{R} \end{cases} . \tag{3.9}$$

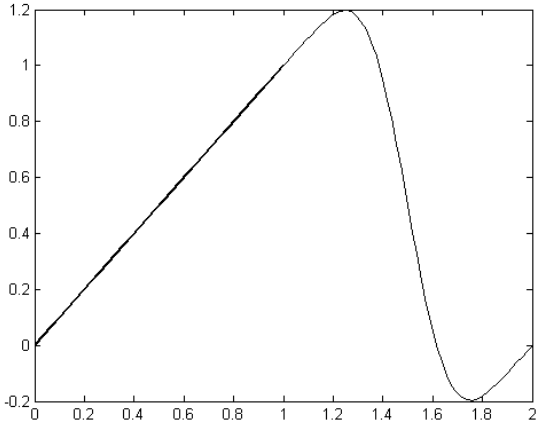


FIG. 3.1. *Smooth periodic function resulting from applying the FC(SVD) method with* 32 *modes to* $f(x) = x$ *over the interval* $[0, 1]$.

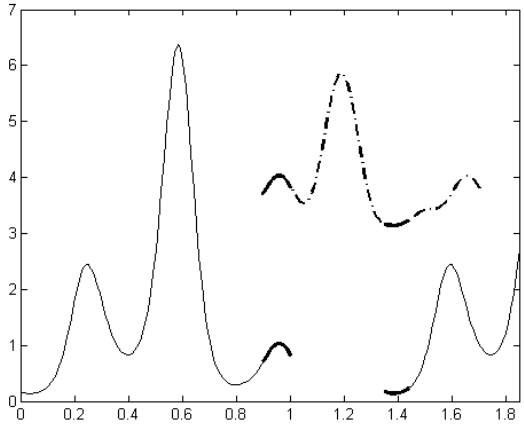


FIG. 3.2. *Calculation of a periodic extension of* $f(x) = e^{\sin(5.4\pi x - 2.7\pi x) - \cos(2\pi x)}$ *using the FC matching algorithm described in* (3.9) *with* $N_\delta = 10$ *and* $d = .35$. *The solid lines represent the function* $f(x)$ *and its translation* $f(x-d-1)$ *while the dark lines represent the segments used in the matching process. The dashed lines represent* f^{match}, *raised for clarity*

Due to the approximation errors arising in FC(SVD), the $(1 + d)$-periodic function f^{de} defined in (3.9) is piecewise discontinuous (the lable "de" stands for "discontinuous extension"). However, a spectrally accurate Fourier continuation for f can now be obtained as a discrete Fourier transform of the grid values of the function f^{de}. To do this, we require that $1 + d$ and δ be integer multiples of h. For appropriate positive integers N_δ and N_d, defining

$$\delta = (N_\delta - 1)h \quad \text{and} \quad d = (N_d - 1)h,$$

sampling the function f^{de} at the $n + N_d - 2$ evenly spaced points

$$x_j = (j - 1)h, \quad j = 1, ..., n + N_d - 2,$$

and then computing the discrete Fourier transform of the resulting discrete values of f^{de} produces, with FFT speed, a high-order accurate Fourier continuation of the function f.

However, a further variation on the FC(SVD) method is required: even for "reasonable" values of the number of one-dimensional samples, the number of discretization points contained in the boundary intervals, chosen small enough to maintain FFT-type computational speeds, is not large enough to yield optimally accurate SVD-based continuations. To overcome this, [3] proposes preceeding the FC(SVD) calculation with projecting the right hand and left hand segments of $f(x)$ and $f(x-d-1)$ respectively into their correpsonding Gram polynomial bases. One advantage of this procedure is that each of the polynomials in the orthogonal bases can be easily computed at a large number of points in its relevant interval. From this, highly accurate FC(SVD) continuations of the basis functions can be constructed and combined to form the continuation of the original function. In addition, the evaluation of the FC(SVD) of the various basis polynomials can be precomputed and stored, at insignificant memory cost, for each possible (reasonable) choice of N_δ, N_d, and the parameters of the FC(SVD) continuation.

The FC(Gram) algorithm, described in detail in [3], is based on the following: first a "discontinuous-projection" function, shown below, is constructed

$$f^{\mathrm{dp}}(x) = \begin{cases} f^p_{\mathrm{right}}(x-1-d), & \text{for } x \in [0,\delta] \\ f(x), & \text{for } x \in (\delta, 1-\delta) \\ f^p_{\mathrm{left}}(x), & \text{for } x \in [1-\delta, 1] \\ f_{\mathrm{match}}(x), & \text{for } x \in (1, 1+d) \\ f^{\mathrm{dp}}(x+1+d) = f^{\mathrm{dp}}(x), & \text{for all } x \text{ in } \mathbb{R} \end{cases}, \tag{3.10}$$

where f^p_{left} and f^p_{right} are the orthogonal projections of $f_{\mathrm{left}}(x) = f(x),\ x \in [1-\delta, 1]$ and $f_{\mathrm{right}} = f(x-d-1),\ x \in [1+d, 1+d+\delta]$ respectively. Finally, the Fourier continuation f^c of f is constructed through the application of an FFT to f^{dp}. Note that f^c is the unique $n+N_d-2$ term Fourier series that interpolates f^{dp} at the points x_j for $j = 1, ..., n+N_d-2$ and is obtained at a total cost of $O(n \log n)$ operations. This concludes our description of the FC(Gram) continuation procedure.

Next, we use the above method of computing an FC(Gram) continuation series of a given function in conjunction with a transformation back to the interval $[x_l, x_r]$ to obtain the discrete operator $S_{\alpha,\beta,x}$. First, approximate the right-hand side discrete data f with its Fourier continuation

$$f^c(x) = \sum_{k \in t(n+N_d-2)} a_k e^{2\pi i \frac{x_k}{(x_r-x_l)(1+d)}}.$$

The solution of

$$-\alpha v''(x) - \beta v'(x) + v(x) = f^c(x), \quad v'(x_l) = 0, \quad v'(x_r) = 0, \tag{3.11}$$

is then obtained from a few simple calculations. After including the appropriate solutions of the associated homogenous problems, a step that is necessary to meet the boundary conditions, the solution v of (3.11) is given by

$$v(x) = \sum_{k \in t(n+N_d-2)} \gamma_k e^{2\pi i \frac{x_k}{(x_r-x_l)(1+d)}} + c_1 h_1(x) + c_2 h_2(x), \tag{3.12}$$

where

$$\gamma_k = \frac{a_k}{1 + \frac{2\beta\pi i k}{(x_r-x_l)(1+d)} + \frac{4\alpha^2\pi^2 k^2}{(x_r-x_l)^2(1+d)^2}},$$

c_1 and c_2 are constants chosen to fit the boundary conditions and where h_1 and h_2 are the homogeneous solutions

$$h_1(x) = e^{\omega^+ x} \text{ and } h_1(x) = e^{\omega^- x},$$

with $\omega^{\pm} = \frac{\beta \pm \sqrt{\beta^2+4}}{2\alpha}$.

Recalling the ADI algorithm described in section 3.1, we note that $\alpha = D\triangle t/2$ and $\beta = V\triangle t/2$, where V is either $-v_x$ or v_y depending on the step in the algorithm. In particular, $\alpha, \beta \to 0$ as $\triangle t \to 0$. However, due to the nature of the Fourier continuation approximation of f used in computing (3.12), the error between the approximate solution $v(x)$ and the exact solution $u(x)$ does not converge to 0 as $\alpha, \beta \to 0$. Instead, we obtain

$$u(x_j) - v(x_j) \to f_j - f^c(x_j) \neq 0 \text{ as } \alpha, \beta \to 0 \tag{3.13}$$

for x_j near the boundary points x_l and x_r. To overcome this, a correction is required. One such correction, $\eta = (\eta_j)$ can be generated from low-order Finite Difference techniques as follows:

$$\begin{aligned} -\alpha \frac{\eta_{j+1} - 2\eta_j + \eta_{j-1}}{h^2} - \beta \frac{\eta_{j+1} - \eta_{j-1}}{2h} + \eta_j &= f_j - f^c(x_j), \text{ for } j \in S \\ \eta_j &= 0 \text{ otherwise,} \end{aligned} \tag{3.14}$$

where $S = \{1, ..., N_\delta\} \cup \{n - N_\delta + 1, ..., n\}$, and with boundary conditions $\eta_0 = 0$, $\eta_{N_\delta+1} = 0$, $\eta_{n-N_\delta} = 0$, and $\eta_{n+1} = 0$. It is easy to verify that the solution $v(x_j) + \eta_j$ satisfies

$$u(x_j) - (v(x_j) + \eta_j) \to 0 \text{ as } \alpha, \beta \to 0.$$

However, the lack of convergence to zero shown in equation (3.13) followed by the introduction of Finite Difference-based corrections introduces certain types of conditional convergence of the overall FC-AD method. Although this typically has no impact for practical applications, a number of additional components are required to ensure the unconditional stability of the full FC-AD method. These components are not listed in this paper, but may be found in section 4 of [3]. For the purposes of this paper, we use the following discrete operator to compute the solution of (3.3):

$$(S_{\alpha,\beta,x}\, f)_j = \eta_j + v_j. \tag{3.15}$$

All the elements are now in place for the numerical implementation of the FC-AD methodology. We present our results of the application of the FC-AD method to equation (2.1) in section 4.

4. Numerical Results. All the numerical results shown were solved over a system of length 24 and width 1.85. To follow the geometry shown in figure 1, it was necessary to divide the system into appropriate layers: we chose the tissue to have width .5, the membrane to have width .35, and the vessel to have width 1. Figure 4 displays the concentration of the ferrofluid at steady state obtained from iterating the ADI method described in section 3.1 until the time derivative was less than 10^{-7}.

5. Future Work. Due to time constraints, we were unable to obtain numerical results from the Finite Element method described. However, we plan on further exploring the differences between FC-AD methods and Finite Element methods in the future.

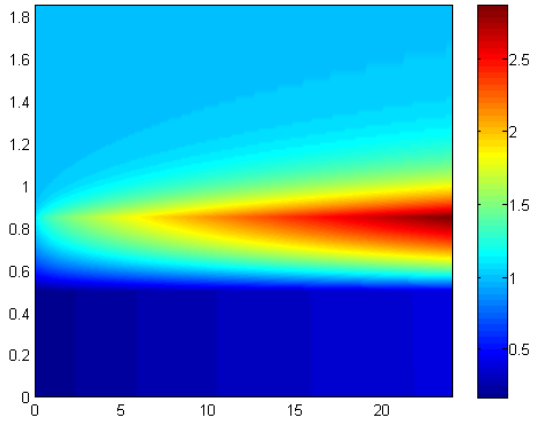


FIG. 4.1. *Concentration at steady state obtained from solving equation* (2.1) *using the ADI method described in section 3.1*

REFERENCES

[1] J. BOYD, *A comparison of numerical algorithms for Fourier extension of the first, second, and third kinds*, J. Comput. Phys., 178 (2002), pp. 118–160.

[2] O. BRUNO, Y. HAN, AND M. POHLMAN, *Accurate, high-order representation of complex three-dimensional surfaces via Fourier-continuation analysis*, J. Comput. Phys., 227 (2007), pp. 1094–1125.

[3] O. BRUNO AND M. LYON, *High-order unconditionally-stable FC-AD solvers for general smooth domains I. Basic elements.* `http://www.acm.caltech.edu/~bruno/FCADHApr17f.pdf`, 2009.

[4] J. DOUGLAS, JR., *Alternating direction methods for three space variables*, Numer. Math, 4 (1962), pp. 41–63.

[5] J. DOUGLAS, JR. AND J. GUNN, *A general formulation of alternating direction methods. Part I. Parabolic and hyperbolic problems.*, Numer. Math, 6 (1964), pp. 428–453.

[6] J. DOUGLAS, JR. AND C. PEARCY, *On convergence of alternating direction procedures in the presense of singular opeartors*, Numer. Math, 5 (1963), pp. 175–184.

[7] A. GRIEF AND G. RICHARDSON, *Mathematical modelling of magnetically targeted drug delivery*, J. Magn. Magn. Mater., 293 (2005), pp. 455–463.

[8] C. JOHNSON, *Numerical solution of partial differential equations by the finite element method*, Cambridge University Press, 1992.

[9] B. SHAPIRO, *Towards dynamic control of magnetic fields to focus magnetic carriers to targets deep inside the body*, J. Magn. Magn. Mater., 321 (2009), pp. 1594–1599.

ASSESSMENT OF COLLOCATION AND GALERKIN APPROACHES TO STOCHASTIC PARTIAL DIFFERENTIAL EQUATIONS

CHRISTOPHER W. MILLER*, RAYMOND S. TUMINARO†, ERIC T. PHIPPS‡, AND HOWARD C. ELMAN§

Abstract. Several algorithms have been proposed for the solution of partial differential equations with random input data including well-known Monte-Carlo based schemes. Recently, the stochastic Galerkin method [2], [3], [7] and the sparse grid collocation method [1], [10], [15] have received significant attention due to their potential to achieve high accuracy at reduced computational costs.

Our study employs Sandia's Trilinos software to implement both of the above methods. We assess the cost and performance of these two methods. The implementations in Trilinos are known to be efficient which allows for a realistic assessment of the computational complexity of the methods. We consider cases where the material uncertainties are modeled by a linear expansion and by a non-linear expansion. We also develop a cost model for both methods that allows us to examine asymptotic behavior. The results of this paper will be expanded in [9].

1. Problem Statement. We investigate the linear elliptic diffusion equation with zero Dirichlet boundary conditions, where diffusivity is given by a random field. If D is a open subset of $\mathbb{R}^n$ and (Ω, Σ, P) is a complete probability space then this can be written as

$$\begin{aligned} -\nabla \cdot (a(\mathbf{x}, \omega)\nabla u(\mathbf{x}, \omega)) &= f(\mathbf{x}, \omega) \quad (\mathbf{x}, \omega) \in D \times \Omega \\ u(x, \omega) &= 0 \qquad (\mathbf{x}, \omega) \in \partial D \times \Omega. \end{aligned} \tag{1.1}$$

The random input field is often given as a truncated Karhunen-Loève (KL) expansion [8] or by a polynomial chaos (PC) expansion [14]. The truncated KL-expansion of the random field is given by

$$a(\mathbf{x}, \omega) \approx a(\mathbf{x}, \boldsymbol{\xi}(\omega)) = a_0(\mathbf{x}) + \sum_{k=1}^{M} \lambda_k \xi_k(\omega) a_k(\mathbf{x}), \tag{1.2}$$

where (λ_i, a_i) are solutions to the integral equation

$$\int_D C(\mathbf{x}_1, \mathbf{x}_2) a_i(\mathbf{x}_2) d\mathbf{x}_2 = \lambda_i a_i(\mathbf{x}_1), \tag{1.3}$$

where C is the covariance kernel of the random field. The random variables are uncorrelated, mean zero, and are given by the formula,

$$\xi_k(\omega) = \frac{1}{\lambda_k} \int_D a(\mathbf{x}, \omega) f_k(\mathbf{x})\, d\mathbf{x}. \tag{1.4}$$

For the remainder we make the further modeling assumption that the ξ_k's are independent and admit a joint probability density of the form $\rho(\boldsymbol{\xi}) = \prod_{k=1}^{M} \rho_k(\xi_k)$. The covariance kernel is positive semi-definite and assume the eigenvalues are ordered to satisfy $\lambda_1 \geq \lambda_2 \geq ... \geq 0$. Define $\Gamma = \times_{k=1}^{M} \Gamma_k = \times_{k=1}^{M} Im(\xi_k)$.

Given a vector of random variables $\boldsymbol{\xi} = [\xi_1, ...\xi_M]^t$ with a joint probability distribution $\rho(\boldsymbol{\xi}) = \prod_{k=1}^{M} \rho_k(\xi_k)$ (not necessarily the same ones appearing in (1.2)), the PC-expansion of a

*University of Maryland at College Park: Department of Applied Mathematics and Scientific Computation, cmiller@math.umd.edu

†Sandia National Laboratories, rstumin@sandia.gov

‡Sandia National Laboratories, etphipp@sandia.gov

§University of Maryland at College Park: Department of Applied Mathematics and Scientific Computation, elman@cs.umd.edu

process projects the random field into the space S_p of multivariate polynomials in $\boldsymbol{\xi}$ of total degree p with dimension $N_\xi = \frac{(M+p)!}{M!p!}$. We can define an inner product on this space by,

$$\langle \Psi\Phi \rangle = \int_\Gamma \Psi(\boldsymbol{\xi})\Phi(\boldsymbol{\xi})\rho(\boldsymbol{\xi})\, d\boldsymbol{\xi} = \int_\Omega \Psi(\boldsymbol{\xi}(\omega))\Phi(\boldsymbol{\xi}(\omega))\, dP. \tag{1.5}$$

Given a basis $\{\Psi_i(\boldsymbol{\xi}) : i = 0 : N_\xi - 1\}$ for S_p which is orthogonal with respect to the above inner product, the PC-expansion of $a(\mathbf{x}, \omega)$ is given by

$$a(\mathbf{x}, \omega) \approx \sum_{k=0}^{N_\xi - 1} a_k(\mathbf{x})\Psi_k(\boldsymbol{\xi}) \tag{1.6}$$
$$a_k(\mathbf{x}) = \langle a(\mathbf{x}, \omega)\Psi_k \rangle.$$

When the random variables are independent then the multivariate orthogonal polynomials are tensor products of the univaritate polynomials orthogonal with respect to ρ_k. The generation of these polynomials is discussed later.

The expansions (1.2) and (1.6) are two ways of representing $a(\cdot, \cdot)$. The latter has the advantage of being more flexible and is capable of representing quantities that are nonlinear in the random component. Note that from an implementation perspective, a KL-expansion is equivalent to a degree one PC-expansion. Throughout this paper we use the notation of a PC-expansion without loss of generality.

Let $V = \{v(\mathbf{x}, \boldsymbol{\xi}) : \|v\|_V < \infty\}$, where $\|v\|_V^2 = \int_\Gamma a|\nabla v(\cdot, \boldsymbol{\xi})|^2\rho(\xi)\, d\boldsymbol{\xi}$. $\|\cdot\|_V$ is a norm if and only if there exist constants a_{min}, a_{max} such that

$$0 < a_{min} \le a(\mathbf{x}, \boldsymbol{\xi}) \le a_{max} < \infty, \tag{1.7}$$

almost everywhere, P-almost surely. Under this assumption, it is a straightforward application of the Lax-Milgram lemma to show that there exists a $u \in V$ satisfying,

$$-\int_\Gamma \nabla \cdot (a\nabla u)v\rho(\boldsymbol{\xi})\, d\boldsymbol{\xi} = \int_\Gamma f v\rho(\boldsymbol{\xi})\, d\boldsymbol{\xi}, \tag{1.8}$$

for all $v \in V$ provided that $f \in L_2(D \otimes \Gamma)$. This is the variational form of (1.1). One could now attain a full Galerkin formulation by integrating over D by parts. We choose not to do this because we intend to use finite differences for the spatial discretization.

2. Stochastic Galerkin Method. Let $\{\Psi_k\}_{k=0}^{N_\xi - 1}$ be a orthogonal basis for S_p with respect to the inner product (1.5). Substituting expansions for $u(\mathbf{x}, \boldsymbol{\xi})$, and $f(\mathbf{x}, \boldsymbol{\xi})$, restricting (1.8) to S_p gives

$$-\int_\Gamma \nabla \cdot (a(\mathbf{x}, \boldsymbol{\xi})(\sum_{i=0}^{N_\xi - 1} \nabla u_p(x)\Psi_i))\Psi_j\, d\boldsymbol{\xi} = \int_\Gamma \sum_{k=0}^{N_\xi - 1} f_k\Psi_k\Psi_j\, d\boldsymbol{\xi} \quad \forall\ j = 0 : N_\xi - 1, \tag{2.1}$$

where either (1.2) or (1.6) is used to expand $a(\mathbf{x}, \boldsymbol{\xi})$. This leads to a set of coupled second order differential equations for the unknown function which can then be discretized in a standard way (e.g. finite elements or finite differences) giving rise to a global linear system of the form,

$$A\vec{u} = \vec{f} \tag{2.2}$$

$$A = \sum_{k=0}^{N_\xi} G_k \otimes A_k, \quad \vec{f} = \sum_{k=0}^{N_\xi} \vec{g_k} \otimes \vec{f_k},$$

where the G_k matrices depend only on the stochastic basis as,

$$G_k(i,j) =< \Psi_k\Psi_i\Psi_j >;\ g_k(i) =< \Psi_k\Psi_i > \quad i,j,k = 0 : N_\xi - 1. \tag{2.3}$$

We note that (2.2) applies to the case when $a(\mathbf{x},\boldsymbol{\xi})$ is expanded using (1.6). For the KL-expansion case the index k runs from 0 to M. For the remainder of this paper we use the polynomial chaos notation.

In our experiments we elect to use finite differences so the A_k matrices correspond to a standard five point operator discretizing the problem $-\nabla\cdot(a_k\nabla u) = f_k$. The matrix A is of order N_xN_ξ where N_x is the number of degrees of freedom used in the spatial discretization. It is relatively sparse in the block sense due to the orthogonality of the stochastic basis functions. Specifically G_0 is diagonal and G_k has at most two entries per row for $k > 0$ in the case where diffusion is modeled by (1.2) and $\rho_k(\xi_k)$ is symmetric with respect to the origin.

One significant obstacle to the Galerkin method is that it requires the solution to the large linear system in (2.2). While that may appear prohibitively expensive this system does not need to be stored and often the number of iterations required by an iterative method is only slightly larger than that required for a single deterministic system. If the expansion of $a(\mathbf{x},\omega)$ satisfies (1.7) then the system matrix is symmetric and positive definite. Thus, the conjugate gradient method becomes the preferred solver. Since the CG method only requires matrix vector products the large global stiffness matrix is never fully assembled. Instead the matrix vector product is preformed implicitly following the procedure described in [6]. The A_k's are assembled and the product is expressed as $(Au)_j = \sum_{k=0}^{N_\xi} < \Psi_k\Psi_i\Psi_j > (A_ku_i)$. The terms A_ku_i are precomputed and then scaled as needed. This can save time and memory since most $< \Psi_k\Psi_i\Psi_j >$ are zero. It is most advantageous in the case where $a(\mathbf{x},\boldsymbol{\xi})$ is modeled by (1.2). It also saves a significant amount of memory since it only requires the assembly of N_ξ (or M in the case of a KL-expansion) order N_x stiffness matrices.

A preconditioner is also needed to solve the system. If the variance of the input process is not large then a natural and effective choice of preconditioner is the matrix $A_0^{-1}\otimes G_0^{-1}$ where A_0 is the mean stiffness matrix [11]. The action of the inverse of A_0 is approximated using a single iteration of algebraic multigrid which must be applied N_ξ times.

3. Sparse Grid Collocation. The collocation method is an alternative to the Galerkin scheme. It samples the input operator at a predetermined set of points $\Theta = \{\boldsymbol{\xi}^{(1)},...,\boldsymbol{\xi}^{(n)}\}$ and constructs a polynomial interpolant $u \approx \sum_{k=0}^{|\Theta|} u_k(x)L_k(\boldsymbol{\xi})$ where L_k is some Lagrange interpolating polynomial and u_k is found by solving the deterministic PDE

$$-\nabla\cdot(a(\mathbf{x},\boldsymbol{\xi}^{(k)})\nabla u(\mathbf{x},\boldsymbol{\xi}^{(k)}) = f(\mathbf{x},\boldsymbol{\xi}^{(k)}). \tag{3.1}$$

Since multidimensional interpolation is not necessarily well defined for a given set of points care must be taken to ensure that an interpolant exists. The natural choice of points consists of tensor products of one-dimensional point sets and the L_k's are tensor products of one dimensional Lagrange polynomials. Unfortunately the number of points is exponential in the number of random variables involved. A less costly alternative to this is the Smolyak sparse grid interpolation algorithm which produces an approximation of a similar accuracy to full tensor grids while using many fewer points [13]. An M dimensional level p grid is defined so that quadrature based on the Smolyak interpolation formula is exact for all M-variate polynomials of total degree p (Fig. (3.1)). It is shown in [2] and [1] that such a grid will produce a solution to (1.1) of similar accuracy to an order p Galerkin scheme. The sparse grid will have on the order of 2^p more points than there are stochastic degrees of freedom in the Galerkin scheme, $|\Theta| \approx 2^pN_\xi$ for $M \gg 1$ [15]. Our routines for the construction of sparse grids can be found in the *Dakota* package [4].

For a fully non-intrusive collocation method the diffusion coefficients would be sampled and discretized in space at the sample point. This repeated assembly can be very expensive. We elect in our implementations to take advantage of the fact that the stiffness matrix at a given value of the random variable is a linear combination of the stiffness matrices appearing in (2.2). We assemble these matrices first and then compute the sum at each sample point. This is somewhat intrusive but it greatly reduces the amount of time required to perform assembly.

In the case of collocation, the repeated cost of constructing an algebraic multigrid preconditioner (which can often be as costly as the conjugate gradient iterative solution phase) can be eliminated if one simply builds an algebraic preconditioner for the mean problem and uses this for all the deterministic systems. If the variance of the operator is small then the mean based AMG preconditioner is nearly as effective as doing AMG on each sub-problem and saves time in setup costs.

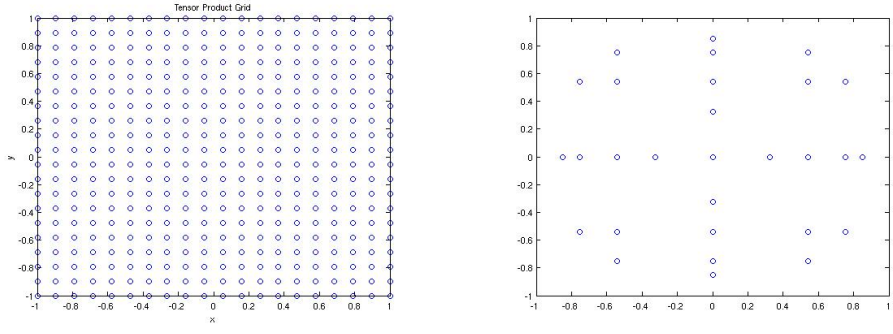


FIG. 3.1. *Two dimensional level 3 tensor and sparse grids.*

4. Modeling Computational Costs. From an implementation perspective, collocation is quite advantageous in that it requires only a modest interaction between an existing deterministic PDE application and a stochastic capability. Collocation samples the stochastic domain at a discrete set of points by solving a set of independent deterministic problems. This is often accomplished by repeatedly invoking a deterministic application with different input parameters determined by the collocation point-sampling method. A Galerkin method, on the other hand, is much more intrusive as it requires the solution of a system of equations with a large coefficient matrix which has been discretized in both spatial and stochastic dimensions. This can be implemented by repeatedly invoking a deterministic application to build and store the A_k's in (2.2).

From a computational cost perspective, there are trade-offs with both methods. The Galerkin method requires the computation of the double and triple products associated with the orthogonal polynomials, the assembly of the right hand side and the A_k matrices, and finally the solution to a very large $N_\xi N_x$ coupled matrix. Collocation requires the assembly of a sparse grid and the derivation of an associated sparse grid quadrature rule, and the assembly/-solution of a series of deterministic sub-problems. Further, as was stated above, the number of sample points needed for collocation is much larger than the dimension of the Galerkin system required to achieve comparable accuracy.

To better understand the relationship between these two methods, we develop a cost model. If we use a mean based preconditioner for the collocation method we can roughly

TABLE 4.1
Degrees of Freedom for Various Methods

M = 2	Level k Sparse Grid $\lvert\Theta\rvert$	Galerkin N_ξ	Non-Zero Blocks per row in SFEM γ	Tensor Grid
k = 1	5	3	2.33	4
k = 2	14	6	3.00	9
k = 3	30	10	3.40	16
k = 4	55	15	3.67	25
M = 10				
k = 1	21	11	2.82	1024
k = 2	231	66	4.33	59049
k= 3	1771	286	5.62	1048576
k = 4	10626	1001	6.71	9765625
M = 20				
k = 1	41	21	2.90	1.04×10^6
k = 2	861	231	4.64	3.49×10^9
k= 3	12341	1771	6.22	1.0995×10^{12}.

model its costs by

$$collocation\ cost = Z_C 2^p N_\xi(\alpha + 1) \tag{4.1}$$

where p is the Smolyak grid level, N_ξ is the number of degrees of freedom needed by an order p Galerkin system, Z_C is the number of PCG iterations needed to solve a single deterministic system, and α is the cost of a single preconditioning invocation relative to a matrix-vector product. In our application, we fix the multigrid parameters as follows. One V-cycle is performed at each iteration and within each V-cycle one symmetric Gauss-Seidel iteration is invoked for presmoothing and for postsmoothing. The coarsest grid is coarse enough so that a direct solver can be used without affecting the cost per iteration. These parameters were chosen to optimize the run time of a single deterministic solve. The cost to apply a single multigrid iteration is roughly equivalent to 5-6 matrix products (2 matrix-vector products for fine level presmoothing, another 2 for fine level postsmoothing, and 1 matrix-vector product for a fine level residual calculation). Thus α can be assumed to be 5 or 6.

The total cost of the Galerkin method can be modeled by

$$Galerkin\ cost = Z_{SG}(N_\xi\alpha + N_\xi\gamma) \tag{4.2}$$

where Z_{SG} is the number of PCG iterations required to solve the Galerkin system. $N_\xi\alpha$ is the mean based preconditioning cost for a single iteration of the stochastic Galerkin method. $N_\xi h$ is the cost of a single matrix vector product for (2.2). When an explicit matrix vector product is used γ is roughly the number of non-zero blocks per row in (2.2). When an implicit method is used with (1.6) the cost is identical to the explicit case. For the case of a KL-expansion with an implicit matrix-vector product γ is roughly $M + 1$. The cost of the two methods is identical when (4.1) and (4.2) are equated. After canceling terms and ignoring the matrix vector cost in (4.1), this gives

$$2^p\alpha \approx (Z_{SG}/Z_C)(\alpha + \gamma) \tag{4.3}$$

when the cost of two methods are roughly comparable. If, for example, the growth in iterations (Z_{SG}/Z_C) is order 1 and the preconditioning costs dominate the matrix vector costs (i.e.,

$\alpha \gg \gamma$). Then, we can expect that Galerkin significantly outperforms collocation due to the 2^p factor. However, when γ is large compared to the preconditioning cost, collocation is more attractive. Table 4.1 gives values of N_ξ, $|\Theta|$, and γ for various values of M and p. One can observe that the estimate $2^p N_\xi \approx |\Theta|$ is a slight overestimate but improves as M grows larger. For reference the number of points used by a full tensor product grid is shown.

The remainder of this paper seeks to understand the model and assess the validity of assumptions. In particular, we compare the accuracy of a level p Smolyak grid with a p^{th} degree polynomial approximation of the Galerkin approach. We also investigate the cost of matrix-vector products, and the convergence behavior of mean based preconditioning.

5. CG Convergence. In this section we explore the ratios in (4.3). When algebraic multigrid is used for an elliptic PDE, we expect that the number of iterations is bounded independent of the mesh spacing in the spatial dimension. For well-posed Poisson problems multigrid converges rapidly. Since collocation solves repeated deterministic systems we expect multigrid to behave well. For Galerkin systems the performance of mean based preconditioning is more complicated. To understand this we investigate the problem

$$-\nabla \cdot (a(x,\xi)u(x,\xi)) = f \tag{5.1}$$

in the domain $[-.5,.5]^2$ with zero Dirichlet boundary conditions. The diffusion coefficient given in a one term KL expansion is

$$a(\mathbf{x},\xi) := 1 + \sigma \frac{1}{\pi^2} \xi cos[\frac{\pi}{2}(x^2+y^2)]. \tag{5.2}$$

We choose the function

$$u := exp(-|\xi|^2)16(x^2-.25)(y^2-.25) \tag{5.3}$$

as the exact solution and the forcing term f is defined by applying (5.1) to u.

The diffusion coefficient must remain positive for the problem to remain well-posed. The problem remains well-posed when

$$|\sigma \frac{1}{\pi^2} \xi cos(\frac{\pi}{2} r^2)| < 1 \Rightarrow |\xi| < \frac{\pi^2}{\sigma}. \tag{5.4}$$

Unfortunately (5.4) cannot be guaranteed for unbounded random variables. There are various ways this can be addressed. In this paper we choose to model random variables by a truncated Gaussian distribution

$$\rho(\xi) = \frac{1}{\int_{-c}^{c} \exp(-\frac{\xi^2}{2})\, d\xi} \exp(-\frac{\xi^2}{2}) \chi_{[-c,c]} \tag{5.5}$$

This is a modeling assumption made for the purpose of investigation. For a certain range of values of c the problem is guaranteed to remain well-posed. Polynomials orthogonal to a truncated Gaussian measure are referred to as Rys polynomials. As the parameter c is increased, the measure approaches the standard Gaussian measure and the Rys polynomials are observed to approach the behavior of the Hermite polynomials. For our implementation of collocation, the sparse grids are based on the zeros of the orthogonal polynomials. This leads to an efficient multidimensional quadrature rule.

All orthogonal polynomials can be written using a three term recurrence relation

$$\psi_{i+1}(\xi) = (\xi - \alpha_i)\psi_i(\xi) - \beta_i\psi_{i-1}(\xi), \tag{5.6}$$
$$\text{where } \alpha_i = \frac{\int_\Gamma \xi\psi_i(\xi)^2\rho(\xi)\,d\xi}{\int_\Gamma \psi_i(\xi)^2\rho(\xi)\,d\xi},$$
$$\beta_i = \frac{\int_\Gamma \psi_i(\xi)^2\rho(\xi)\,d\xi}{\int_\Gamma \psi_{i-1}(\xi)^2\rho(\xi)\,d\xi},$$
$$\text{and } \psi_0 = 1\ \psi_{-1} = 0.$$

In the case of Hermite polynomials there exist closed forms for the recurrence coefficients. Such a closed form is not known to exist for the Rys polynomials so a numerical method must be employed. The generation of orthogonal polynomials by numerical methods is discussed extensively in [5]. We compute the α and β recurrence coefficients via the discretized Steltjies procedure [12] where integrals are approximated by quadrature. This has been observed to be effective for the distributions which we study in this paper.

Testing for both the sparse grid collocation method and the stochastic Galerkin method was performed using the truncated Gaussian PDF with several values of c using the Rys polynomials and with the full Gaussian measure using Hermite polynomials. The linear solver for both methods stop when $\frac{\|r\|_2}{\|r^{(0)}\|_2} < 10^{-12}$. Table 5.1 shows the average number of iterations required by each deterministic sub-problem as a function of grid level and the Gaussian truncation point for fixed $\sigma = 3$. The last column uses full Gaussian measure. The PDE becomes ill-posed when $c > \frac{\pi^2}{\sigma} \approx 3.29$. Problems to the right of the double line are ill-posed and are guaranteed to fail for a high enough grid level as some of the collocation points will be placed in the region of ill-posedness. If any of the individual sub-problems failed to converge the method is reported as having failed. We also report the error in the norm $\langle \|u_{exact} - \hat{u}\|^2_{l_\infty} \rangle$.

TABLE 5.1

Average iterations for the deterministic sub-problems in the stochastic collocation method as a function of the Rys cutoff For $\sigma = 3$

	c=1	2	3	4	∞
level = 1	Error = 0.0382 Iterations = 10	0.1083 10.50	0.1024 10.50	0.0838 10.50	0.0764 11.33
level = 2	5.6381e-04 10.33	0.0154 10.67	.0342 11.00	0.0411 11.33	0.0466 16.40
level = 3	5.7703e-4 10.25	.0171 10.75	.0460 11.75	0.0639 12.25	Did Not Converge
level = 4	3.8297e-6 10.20	.0013 11.00	.0078 12.2	0.0134 14.60	DNC
level = 5	3.9185e-6 10.17	.0014 11.00	.0117 12.67	DNC	DNC
level = 6	1.49417e-8 10.30	6.6308e-5 11	0.0013 13.29	DNC	DNC

Table 5.2 shows iteration counts for the Galerkin method. For problems in one random variable, the stochastic collocation method produces the exact same result as the Galerkin method. Since this is the case we do not report errors for the Galerkin method. Again, problems to the right of the double line are ill-posed and the linear systems involved are guaranteed to become indefinite as p increases [11].

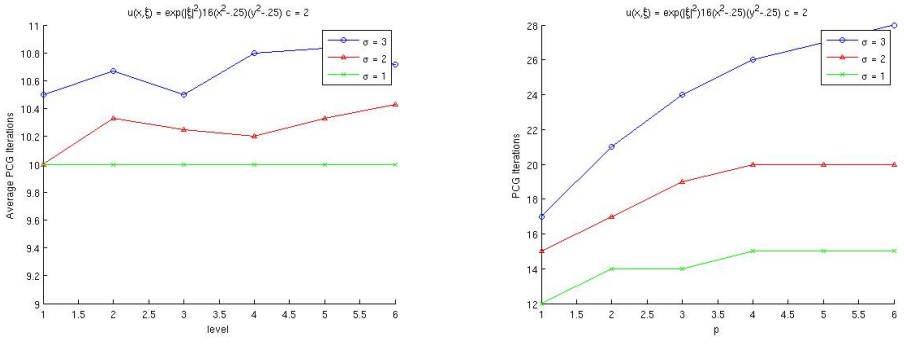


FIG. 5.1. *PCG iterations vs grad level and vs polynomial degree for first model problem*

TABLE 5.2
Iterations for the stochastic Galerkin method as a function of the Rys cutoff For $\sigma = 3$

	c=1	2	3	4	∞
p = 1	Iterations = 14	17	17	17	17
p = 2	16	21	24	25	28
p = 3	17	24	31	35	47
p = 4	17	26	38	51	95
p = 5	18	27	44	94	DNC
p = 6	18	28	49	DNC	DNC

Tables 5.1 and 5.2 show that the iteration counts are fairly well behaved when mean-based preconditioning is used. Fig. 5.1 illustrates iterations as a function of sigma for both methods for fixed $c = 2$. In general, iterations grow as the degree of polynomial approximation increases. However, the rate of this growth is also a function of the variance and the truncation point for the random variable. In our specific example, the key aspect affecting iterations is coercivity. In particular, coercivity is lost with Gaussian random variables (i.e. Hermite polynomials) due to their unbounded nature. In the table, iteration counts are acceptable for low order polynomial approximations where the global matrix remains positive-definite. However, as the degree of polynomial approximation increases, the matrix becomes indefinite and convergence suffers. The tables illustrate how iterations are generally well-behaved when the Gaussian is truncated far from the point where coercivity is lost (and behaves similar to the Gaussian case if not sufficiently truncated to guarantee coercivity). In general, the number of iterations required for a single deterministic sub-problem is comparable to one half the number of iterations required for the Galerkin method though this becomes less true as the problem loses coercivity.

6. Model Validation and Comparison. In this section we compare the performance of the two methods using both the model developed above and the implementations in Trilinos. For our numerical examples, we consider

$$-\nabla \cdot [(\mu + \sigma \sum_{k=1}^{M} \lambda_k \xi_k f_k(x))\nabla u] = 1 \tag{6.1}$$

where

$$\mu = .2,\ \sigma = .1,\ M = 3:5, \tag{6.2}$$

$\{\lambda_k, f_k\}$ are the eigenpairs associated with the covariance kernel

$$C(\mathbf{x}_1, \mathbf{x}_2) = exp(-|x_1 - x_2| - |y_1 - y_2|). \tag{6.3}$$

The KL-expansion of this kernel is investigated extensively in [7]. The ξ_k are chosen to be identically independently distributed uniform random variables on $[-1, 1]$. Since no analytic expression is known to exist, to measure the error for the Galerkin method the 'true' solution is assumed to correspond with the solution produced by a high order ($p = 10$) Galerkin scheme. To measure the error for the collocation method we use the solution from a level 10 sparse grid approximation as the true solution. The error is measured by computing the mean and variance of the approximate solutions and comparing it to the mean and variance of the order 10 approximations. The linear solves for both methods stop when $\frac{||r||_2}{||r^{(0)}||_2} < 10^{-12}$. In measuring the time, setup costs are ignored. The times reported are non-dimensionalized by the time required to perform a single deterministic matrix vector product and compared with the model developed above.

Fig. 6.1 shows that the stochastic Galerkin method gives higher accuracy per stochastic degree of freedom. This can be seen in the close vertical alignment of the data points in Fig. 6.1. The Galerkin scheme requires substantially fewer degrees of freedom. Since the degrees of freedom in the Galerkin scheme are coupled, the cost per degree of freedom will be higher. Fig. 6.2 shows that Galerkin still performs significantly better than the collocation method, producing a more accurate solution in less time, and that the gap widens as the dimension of the random variable increases. Furthermore Fig. 6.2 shows that the cost model developed above is relatively accurate for the Galerkin method. The problem with the collocation model is that for these low dimensional problems the assumption that $2^p N_\xi \approx |\Theta|$ is not valid. We expect the collocation data to agree with the model for high dimensional problems.

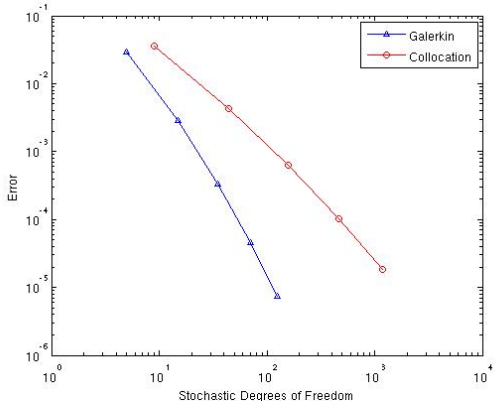


FIG. 6.1. *Errors vs Stochastic DOF for* $M = 4$

7. Conclusion. We have shown that when mean based preconditioning is used that the stochastic Galerkin method outperforms the sparse grid collocation method for solving the linear elliptic diffusion when the random diffusion coefficient is modeled by a KL-expansion. We have also developed a cost model for both methods which closely mirrors the complexity of the algorithms. We suspect that when a PC-expansion is used to model diffusion that the matrix vector product involved in solving the Galerkin system becomes prohibitively expensive. Currently, running numerical experiments using the PC-expansion for diffusion is beyond our computational abilities for large values of the stochastic discretization parameters. We plan to apply our model to these regimes in a future study.

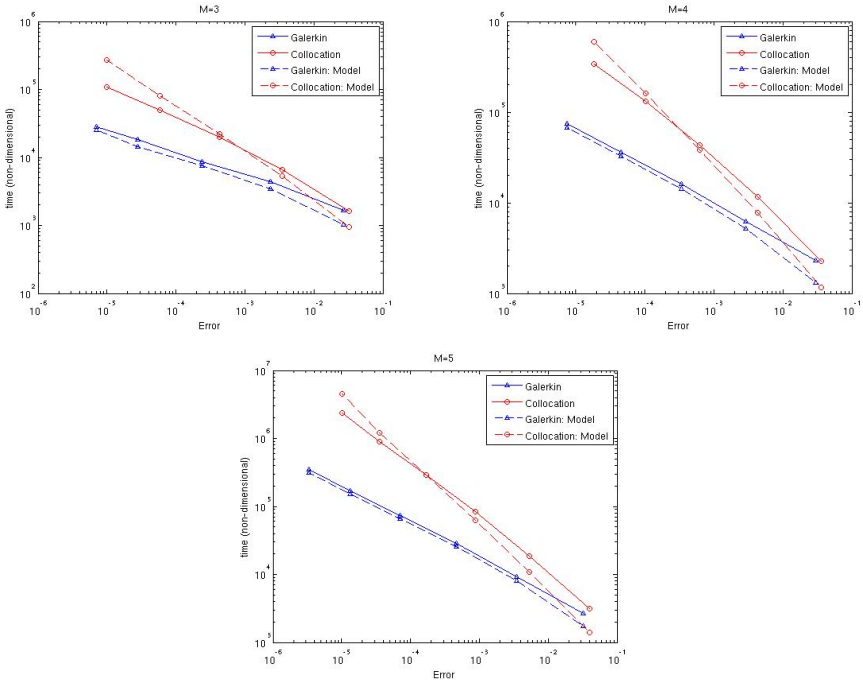


FIG. 6.2. *Solution Time Vs. Error for* $M = 3 : 5$.

REFERENCES

[1] I. BABUŠKA, F. NOBILE, AND R. TEMPONE, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Journal of Numerical Analysis, 45 (2007), pp. 1005–1034.

[2] I. BABUŠKA, R. TEMPONE, AND G. ZOURARIS, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM Journal of Numerical Analysis, 42 (2004), pp. 800–825.

[3] M. DEB, I. BABUŠKA, AND J. ODEN, *Solution of stochastic partial differential equations using galerkin finite element techniques*, Comput. Methods Appl. Mech. Engrg, 190 (2001), pp. 6359–6372.

[4] M. ELDRED ET AL., *Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 4.0 users manual*, Tech. Rep. SAND2006-6337, Sandia National Laboratories, October 2006.

[5] W. GAUTSCHI, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, Oxford, 2004.

[6] R. GHANEM AND M. PELLISSETTI, *Iterative solution of systems of linear equations arising in the context of stochastic finite elements*, Advances in Engineering Software, 31 (2000), pp. 607–616.

[7] R. GHANEM AND P. SPANOS, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1991.

[8] M. LOÈVE, *Probability Theory*, Springer-Verlag, 1978.

[9] C. MILLER, H. ELMAN, R. TUMINARO, AND E. PHIPPS, *A comparison of the stochastic galerkin method and sparse grid collocation method for stochastic linear partial differential equations*. In Preparation.

[10] F. NOBILE, R. TEMPONE, AND C. G. WEBSTER, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal of Numerical Analysis, 46 (2008), pp. 2309–2345.

[11] C. POWELL AND H. ELMAN, *Block-diagonal preconditioning for spectral stochastic finite element systems*, IMA Journal of Numerical Analysis, 29 (2009), pp. 350–375.

[12] R. SAGAR AND V. SMITH, *On the calculation of rys polynomials and quadratures*, International Journal of Quantum Chemestry, 43 (1992), pp. 827–836.

[13] S. SMOLYAK, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Dokl. Akad. Nauk SSSR, 148 (1963), pp. 1042–1043.

[14] N. WEINER, *The homogeneous chaos*, American Journal of Mathematics, 60 (1938), pp. 897–936.

[15] D. XIU AND J. HESTHAVEN, *High-order collocation methods for differential equations with random inputs*, SIAM Journal of Scientific Computing, 27 (2005), pp. 1118–1139.

VISUALIZING LARGE-SCALE MULTIVARIATE DATA USING PARALLEL COORDINATES PLOTS IN VTK AND PARAVIEW

DAVID FENG* AND ANDREW WILSON†

Abstract. As information visualization becomes more commonly used in conjunction with scientific visualization to display complex data sets, it is important that standard visualization libraries and tools such as the Visualization Toolkit (VTK) and ParaView begin to incorporate standard information visualization techniques. Parallel coordinates plots are a standard tool for visualizing trends and clusters in multivariate data sets. This report describes the design and implementation of VTK classes that generate parallel coordinates plots and their integration into two ParaView plugins that separately handle large and small-scale data sets.

1. Introduction. Parallel coordinates plots are a standard information visualization technique used to highlight trends and clusters in multivariate data sets. They are often the most powerful when linked to scientific/spatial visualizations of the same data, as samples that exhibit interesting patterns in the parallel coordinates plot can then be seen in spatial context. As interest in information visualization grows, it is important that standard visualization libraries and applications grow to incorporate such techniques. Parallel coordinates plots were first proposed by d'Ocagne[1] and independently re-discovered a century later by Inselberg[4]. Since then, much work has gone into improving the plot in terms of visual perception, rendering efficiency, and interaction.

The standard parallel coordinates plot takes a multivariate data set and represents each variable as an axis and each sample as a polyline that intersects each axis. For the sake of clarity, consider an example toy data set:

	height	age	score
Anne	60”	19	96
Betsy	51”	17	80
Frank	54”	16	78
Joe	61”	18	95
Sam	48”	15	81
Edward	45”	19	97
Vivian	55”	17	85

Each row of this table represents a sample, in this case a teenager. Each column describes a different property, or variable, of a single teenager. In this case, each teenager has three properties: their height, age, and score on a generic examination. Upon closer scrutiny, it appears that taller teenagers appear to score higher on this examination. Also, younger teenagers tend to be shorter as well. Viewed in the parallel coordinates plot shown in Figure 1.1, all of these trends pop out immediately. In this plot, each teenager is a line, and the intersection of that line with the different variable axes indicates their height, age, and exam score. Even more noticeable than the simple trends described above is the outlier: one of the line refers to a teenager that is relatively short while being older than his peers. Referring back to the table, it is clear that Edward is this outlier.

This report describes the design and implementation of parallel coordinates plot classes in the Visualization Toolkit (VTK), one of the most widely used visualization software libraries, and the integration of these classes into ParaView, a large-scale data visualization tool

*University of North Carolina at Chapel Hill, dfeng@cs.unc.edu
†Sandia National Laboratories,atwilso@sandia.gov

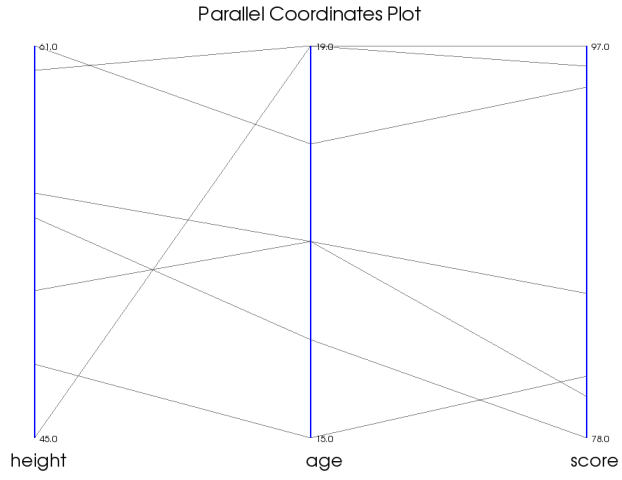


FIG. 1.1. *The parallel coordinates plot corresponding to the data shown in the example table of teenagers above. Shorter teenagers tend to be younger and score lower. One line that bucks this trend is the shortest teenager who also happens to be the oldest and has the highest score.*

based on VTK. The ParaView integration process presented interesting design challenges due to ParaView's focus on large, distributed data sets.

The VTK implementation of parallel coordinates plots uses the existing view and representation framework that drove the development of other information visualization techniques including graph and tree visualizations. The basic premise is that a representation is responsible for taking a data set and converting it into a renderable form (points, lines, polygons, etc.) and the view displays the representation in a render window and handles user interaction.

To integrate these classes into ParaView, special attention was paid to how to handle extremely large data sets. A parallel coordinates plot represents each sample in a multivariate data set as a line, which means that data sets with billions of samples require billions of lines to be plotted. With so many primitives, interactivity is difficult if not impossible to achieve. In addition to implementing the standard parallel coordinates plot as described above, both the VTK classes and ParaView plugin contain a render mode that uses histogram precomputation to draw a summary of the plots rather than every individual sample. Additionally, when operating in a distributed environment, the ParaView plugin performs all precomputations in parallel using VTK's existing MPI communication classes.

2. Relationship to Scatter Plots. It is important to point out that parallel coordinates plots are closely related to another information visualization technique: the scatter plot. Scatter plots represent two variable values of a sample as Cartesian (x, y) coordinates. Therefore, a single point in a scatter plot encodes the same amount of information as a line segment connecting two variable axes in a parallel coordinates plot. This more compact (and ubiquitous) representation does not easily translate to more than two or three variables. However, it is often easier to think about selection other interactions with a parallel coordinates plot in terms of its effect on the corresponding scatter plot. This will become particularly true in Section 3.1, which discusses the mechanisms for line selection.

3. VTK Parallel Coordinates View and Representation. The majority of the parallel coordinates plot functionality has been implemented in two classes: `vtkParallelCoordinatesView`, based on `vtkRenderView`, and `vtkParallelCoordinates-`

Representation, inheriting from vtkRenderedRepresentation. A custom interactor style, vtkParallelCoordinatesInteractorStyle, catches user interactions and translates them into parallel coordinates-specific actions that are sent to the vtkParallelCoordinatesView for processing. As described in the introduction, the view compiles all selection information (selection mode, selected points, etc) and sends it to the representation class. The representation handles translating the input into renderable representations, more specifically vtkProps. The view class displays those props in its vtkRenderWindow.

vtkParallelCoordinatesRepresentation accepts as input any vtkDataObject and treats each contained array as a separate variable. This implies that all input arrays must have the same length. If an array has multiple components, only the first component is used, as treating array components as different arrays added a considerable, and potentially unnecessary, amount of code complexity. Support for vtkArrayData objects is functional, albeit not ideal. Currently the representation uses the vtkArrayToTable filter to copy the first array from the vtkArrayData object into a new table. Because vtkArrays have a completely separate iteration mechanism from vtkAbstractArrays, this unfortunate copy greatly simplifies code internally.

The representation supports curved lines instead of straight lines. To replace the line segments connecting axes with curves, simply toggle the boolean UseCurves flag on the representation. Internally, these curves are cubic splines with zero-derivative constraints at the axes, as described by Moustafa and Wegman[5]. This representation preserves line intersections between plot axes, although those intersections occur in locations different from the straight line representation. Graham and Kennedy advocate more traditional cubic splines with constant derivative constraints to make axis intersections more distinct[2], but intersections between axes and line densities are not preserved in this representation.

The representation class also takes as optional input a vtkTable whose first column contains a vtkStringArray of axis title names. If this input is empty, axis titles are generated from the array titles of the input data. If there are still no title, an automatic set of title are generated. Axis title can also be set manually by calling SetAxisTitles(...) on the representation class.

To use the vtkParallelCoordinatesRepresentation, first instantiate a vtkParallelCoordinatesView and then use the AddRepresentation(...) method. An example of this simple functionality can be seen in the vtkSNL code repository under vtkSNL/Examples/Cxx/ParallelCoordinates. This small applet demonstrates all of the features of the parallel coordinates classes. To load an example data set, select File → Load Default Data.

The user has control over several useful properties of the parallel coordinates plot. Line opacity and plot colors are controlled by the vtkViewTheme applied to the view and representation. The map from view theme property to plot property is as follows:

Theme	Plot
cell color	line color axis label color
cell opacity	line opacity
background color	background color
edge label color	axis color

3.1. User Interaction. There are two interaction styles: axis manipulation and line selection. In both modes, hovering the mouse near an axis results in a highlight box that displays the axis value at the nearest point on the axis. In axis manipulation mode, the user has control

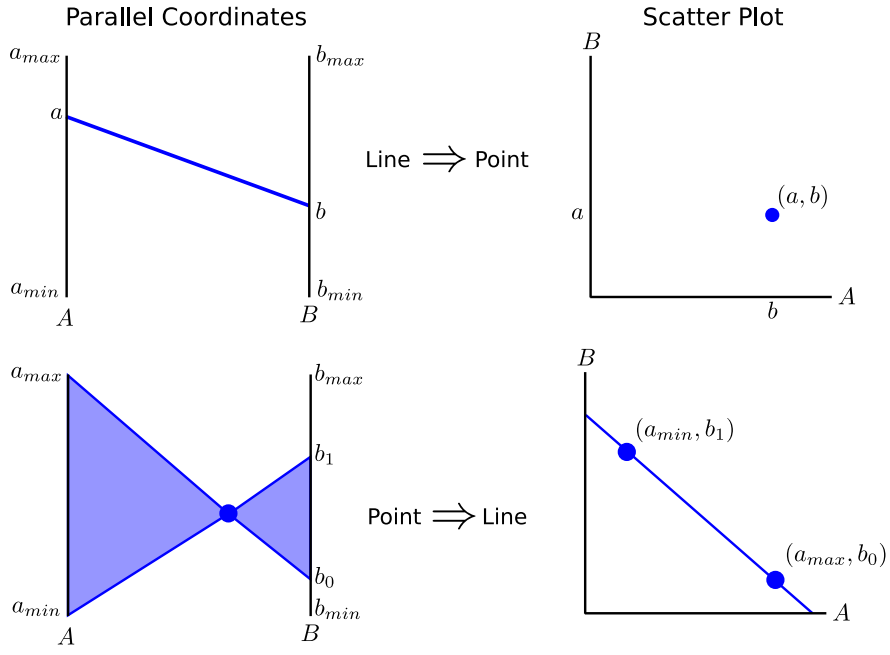


FIG. 3.1. *Top: a segment between two axes in a parallel coordinates plot represents the same information as a point in a Cartesian scatter plot. Bottom: all line segments that intersect at a point between two parallel coordinates plot axes represent a line in a Cartesian scatter plot.*

over axis position, axis order, and the value range of each axis. The user can click on an axis and drag it to a new position; when two axes get close to each other, they swap positions. Clicking and dragging on the ends of the axes allows the user to increase or decrease the minimum or maximum value on the axis. Line selection mode presents to the user three three selection techniques: lasso selection, angle selection, and linear function selection.

3.1.1. Lasso Selection. To use the lasso selection mode, the user simply clicks and drags over the plot and lines that pass near the dragged path are selected. While performing hardware picking is efficient, it only works when all lines are being drawn. The histogram representation mode described in Section 3.2 specifically seeks to avoid drawing all lines, so an alternative way to think about lasso selection as a filter on the raw input data would therefore be useful.

The connection between parallel coordinates and scatter plots can be used to produce such a filter. First consider a single line segment drawn between two parallel coordinates plot axes. This line segment represents a single (x,y) value pair, which in a Cartesian scatter plot is represented as a point. Next consider a point drawn between two axes in a parallel coordinates plot. The entire set of line segments that pass through that point make up a line in scatter plot space. This point-line duality between parallel coordinates and scatter plots is shown in Figure 3.1.

If the user wished to select all lines that pass near a point in the parallel coordinates plot, any (x,y) value from the input data that exists near its corresponding Cartesian line will be selected. Going a step further, if the user wished to select all lines that pass through a line segment drawn in parallel coordinates, we can break that segment down into an infinite set of points interpolating between the two endpoints. Those two endpoints have their corresponding lines in scatter plot space. Interpolating between those two Cartesian lines defines

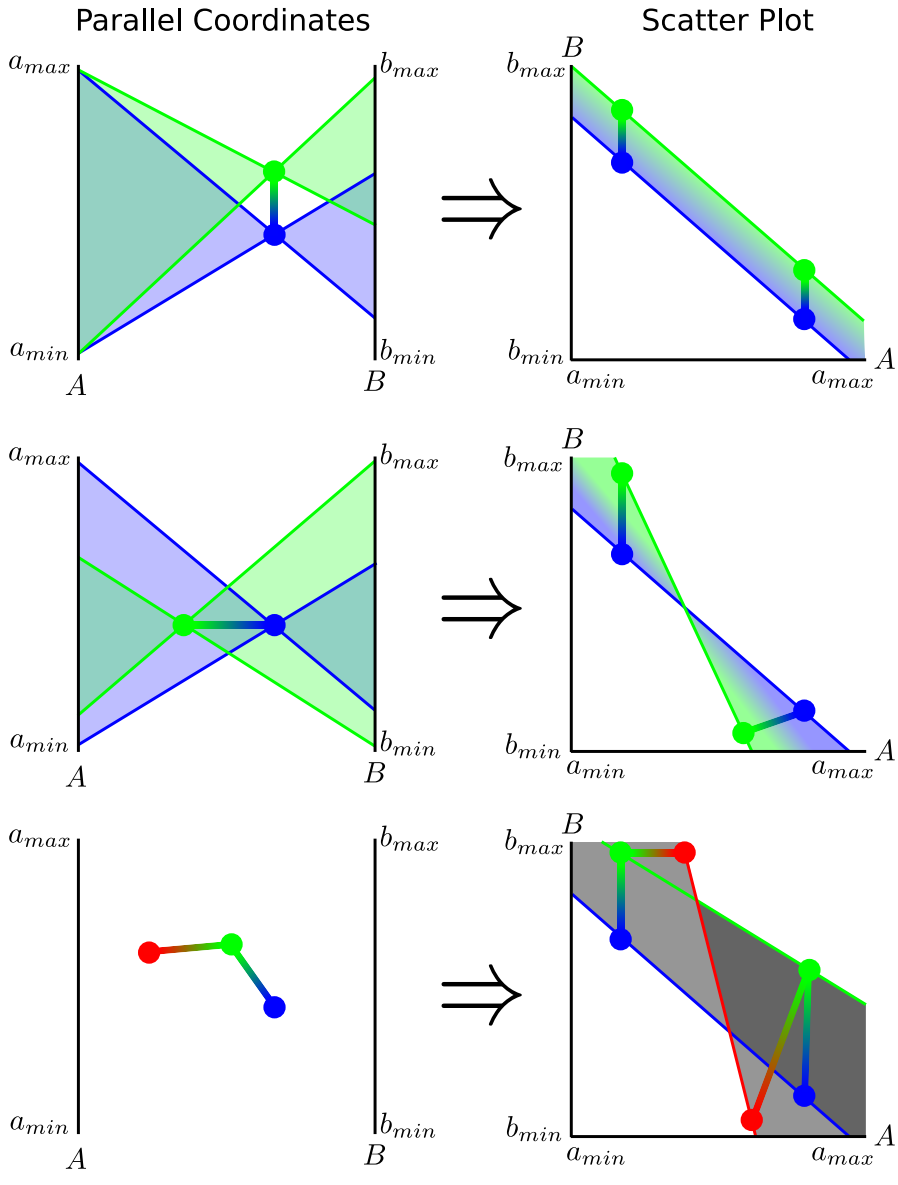
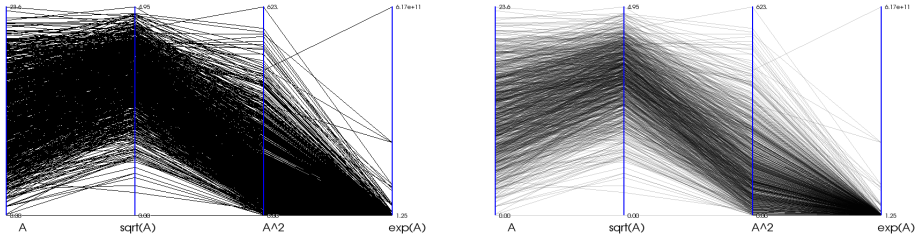


FIG. 3.2. *Top and middle: A line segment drawn in a parallel coordinates plot can be thought of as the region defined by the interpolation between the two Cartesian lines corresponding to the line segment's endpoints. Bottom: multiple line segments produce a more complex region. All points that fall within the union of these Cartesian regions are parallel coordinates lines that pass through the drawn line segments.*

a bounded region, and all (x,y) pairs that fall within that region will be selected, as shown in the top of Figure 3.2. To filter the input, one need only to check that a given (x,y) pair is below one of the boundary lines and above the other.

We can now extend this to a more natural brushing technique that allows the user to simply drag their mouse over the screen, selecting any values that fall under the mouse. When the user drags their mouse over the screen, they in effect draw a series of line segments. This set of segments produces a corresponding set of bounding regions; a value in the data is



(a) Opaque Lines (b) Transparent Lines

FIG. 3.3. *Left: a high density of opaque lines in a parallel coordinates plot can result in overplotting, when the lines blend into a large nondescript block. Right: By reducing the opacity of each line in an overplotted parallel coordinates plot, the viewer can more easily recognize regions of high density. This ability comes at the cost of identifying outliers.*

selected if it falls in the union of these regions, as shown in the bottom of Figure 3.2. The filter test is nearly the same: if a given (x,y) point in the input is above one of the boundary lines and below another, it gets selected. If the user's drawn line extends over an axis, it is necessary to create a different set of filters. Checking to see if a point is above a line is a trivial operation, so it turns out that this selection technique is fairly efficient.

This filter has been implemented in the `vtkBivariateLinearTableThreshold` class. To use this class, one specifies an input table, a pair of columns to filter in that table, and one or more line equations. When the filter function is set to `BETWEEN`, it checks to see if any points fall within the space bounded by all of the supplied linear functions.

3.1.2. Angle Selection. Angle selection is a useful interaction style proposed by Hauser et al. [3] that allows the user to select all lines that have a similar slope between two axes in the parallel coordinates plot. If the two axes have the same value range, this techniques selects all points for which a sample's value for one variable a constant value above (or below) its value for the other variable. In Cartesian space, this corresponds to a horizontal line ($y = x + constant$). If the two axes have different value ranges, the set of all parallel coordinates lines corresponds to a sloped line in Cartesian space($y = slope * x + constant$). The slope of the line is related to the ratio of the range of one axis to the other. All (x,y) pairs that fall near this line are therefore selected. Again, this is a fairly simple test that is implemented in the `vtkBivariateLinearTableThreshold`. To select table rows that fall near one or more linear functions, simply set the filter function to `NEAR`.

3.1.3. Function Selection. More complex trend analysis can be done in function selection mode. When a linear trend exists between two variables, the parallel coordinates lines all tend to intersect near a single point, which does not have to be between the two axes. This is because a point in parallel coordinates space corresponds to a line in Cartesian space. To select all lines that follow this trend, the user needs to draw two characteristic lines on the parallel coordinates plot. Examples of such characteristic lines can be seen in the bottom of Figure 3.1. Those two lines correspond correspond to two points in Cartesian space, which then can be used to extrapolate the linear trend of interest. All (x,y) pairs that fall near this line in Cartesian space are selected. Because of this, function selection is actually a generalization of angle selection, and therefore uses the `vtkBivariateLinearTableThreshold` class in exactly the same manner.

3.2. Histogram Representation. A major limitation of parallel coordinates plots is their ability to scale to large numbers of samples. First, plots with millions or more lines

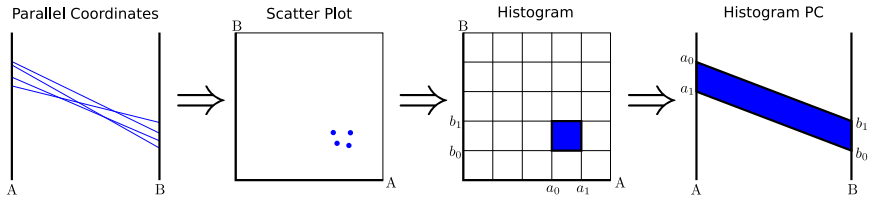


FIG. 3.4. *Four different ways of representing the same information. Left: a parallel coordinates plots. Left-middle: a scatter plot. Right-middle: a 2D histogram with all four data points in the same bin. Right: A parallel coordinates-based representation of the histogram.*

visible tend to become overplotted as lines overlap with each other, as shown in Figure 3.3(a). While decreasing the opacity of the lines (shown in Figure 3.3(b)) adequately addresses this issue, it does not address the more difficult problem of render performance. Rendering each line separately does not scale to large data sets.

Novotný and Hauser developed a conceptually simple, but powerful, approach to achieving parallel coordinates render scalability[6]. The simple insight is that decreasing line opacity in overplotted parallel coordinates visualizations emphasizes regions of high line density and de-emphasizes outliers. Rather than plotting every line repeatedly to produce this density map, one can easily precompute a density map at a more manageable resolution.

A common statistical tool for computing such densities is the histogram. Novotný and Hauser propose computing a 2D histogram between each adjacent pair of axes. Each bin in the histogram represents a unique range of values for both variables. This essentially clusters all lines within similar intersections on both axes into the same histogram bin. Each histogram bin then gets rendered into the parallel coordinates plot as a bar with width determined by the range of values represented by the histogram bin and opacity weighted by the bin count. The equivalence between all of these representations is shown in Figure 3.4. An example of such as plot is shown in Figure 3.5.

As stated previously, methods for dealing with overplotting such as decreasing line opacity and computing histograms directly display regions of high line density. However, an important use of parallel coordinates is to discover outliers, as demonstrated by the toy example in the Introduction. Novotný and Hauser show that it is possible to automatically identify outliers using one of a number of outlier detection algorithms. The algorithm chosen for this work uses a median filter to identify spikes in the histogram. Bins identified as spikes are accepted as outliers if they are below an internally set threshold. This threshold is automatically computed based upon the user's preferred number of outliers. Figure 3.5 contains a histogram-based parallel coordinates plot with outliers.

The `vtkParallelCoordinatesHistogramRepresentation` class computes all histograms and outliers, which inherits from `vtkParallelCoordinatesRepresentation`. The only significant changes are the addition of an internal histogram computation filter (`vtkPairwiseExractHistogram2D`), methods for mapping the histograms to quads, and an outlier computation filter (`vtkComputeHistogram2DOutliers`). The histogram computation filter takes as input a single `vtkTable` and computes a 2D histogram between each adjacent column. Internally it allocates a single instance of the `vtkExtractHistogram2D` filter for each column pair. The outlier computation filter takes as input a `vtkTable` and the extracted histograms come in the form of a `vtkMultiBlockDataSet` that contains multiple `vtkImageData` histogram objects. The histogram extraction filters have customizable output dimensions (accessible through the `NumberOfBins` variable) and histogram extent. To customize the extent of the histogram in either dimension, first enable `UseCustomHistogram-`

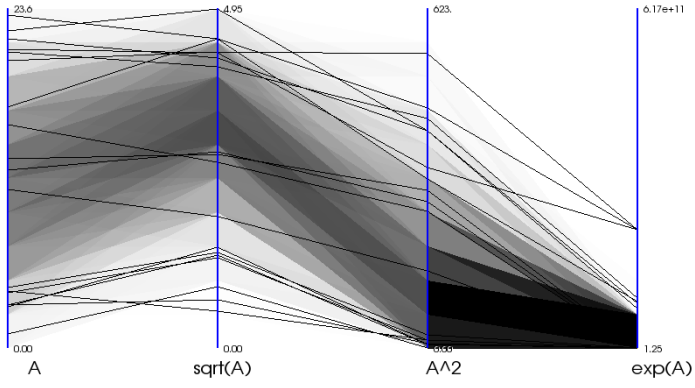


FIG. 3.5. *Drawing a large number of lines can lead to poor plot interactivity. Using a set of precomputed 2D histograms, approximations to density plots such as those seen in Figure 3.3(b) can be generated. The transparent line and histogram plots emphasize regions of high line density and demphasize regions of low line density. By identifying outlier bins in the precomputed histograms, outlier lines can be drawn explicitly.*

`Extents` and then set the `CustomHistogramExtents` array.

As stated previously, the user controls the output of the outlier computation filter by specifying how many outliers they wish to see (via the `PreferredNumberOfOutliers` internal variable). `vtkComputeHistogram2DOutliers` inherits from `vtkSelectionAlgorithm` and produces a `vtkSelection` with a single `vtkSelectionNode` containing an array of outlier table row indices. Additionally, it also produces a second output that consists of a `vtkTable` containing only the outliers rows from the input table.

4. ParaView Plugins. Integrating the previously described VTK classes into ParaView exposes the technique directly to a wide base of visualization users. A client-side ParaView plugin that passes all data to the client is the easiest way of integrating these classes. However, histogram computation is a process that lends itself well to distributed computation. Designing this server side plugin revealed that the two plugins would require slightly different implementation pipelines.

4.1. Client Plugin. The client-side parallel coordinates plugin is based on the `ClientGraphView` plugin found in Sandia's Titan toolkit. It makes no modifications to the underlying VTK classes. In brief, a class called `ClientParallelCoordinatesView` is a Qt class inheriting from `pqSingleInputView` that is a lightweight wrapper around `vtkParallelCoordinatesView`. As with Titan's other client-side plugins, instantiation of this view with a data object in ParaView prompts the view to create its own default representation of that data. `vtkParallelCoordinatesView` by default creates a `vtkParallelCoordinatesHistogramRepresentation`. Additionally, `ClientParallelCoordinatesDisplay.ui` describes a custom Qt display panel that contains controls for the parallel coordinates view. Following the example of other plugins, the `ClientParallelCoordinatesDisplay` class connects all of the widgets in this panel to the `ClientParallelCoordinatesView`.

4.2. Server Plugin. The histogram-based parallel coordinates plot described in Section 3.2 lends itself well to distributed computation. However, the `vtkParallelCoordinatesHistogramRepresentation` class required some modifications to do so. The original VTK class handles histogram and outlier computation internally, thus simplifying usage for the casual user. To migrate this class to the ParaView client/server architecture, all components that touch the original data must be extracted and placed on data server. Once computation is

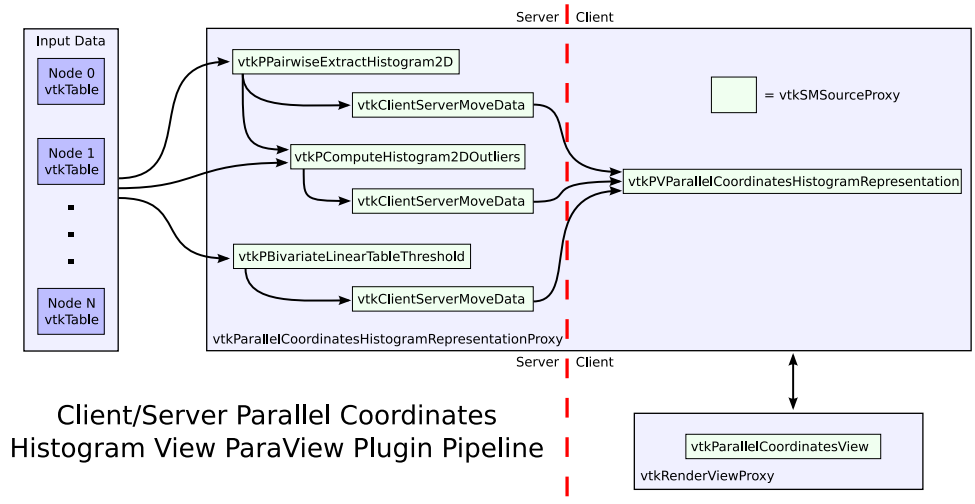


FIG. 4.1. *The pipeline for the client/server parallel coordinates histogram view plugin. Histograms and outliers are identifies on the server, then passed to the representation and view stored on the client.*

complete, the results of those operations can then be sent to the client. Figure 4.1 depicts the basic client-server pipeline used for the distributed representation.

The first step in this process is to parallelize all of the filters used in the client-side representation. The `vtkPExtractHistogram2D` filter wraps the serial `vtkExtractHistogram2D` by having all nodes compute local histograms and then performing a reduction to compute the total histogram. `vtkPPairwiseExtractHistogram2D` uses `vtkPExtractHistogram2D` filters internally rather that `vtkExtractHistogram2D` filters. Outlier computation requires the complete histogram, so the parallel histogram extraction class distributes the reduced histogram to all of the nodes.

Every node then identifies outlier bins individually (all reaching the same results) and extracts the input subset that falls within outlier bins. `vtkPComputeHistogram2DOutliers` combines all of the outlier data into a single array that resides on the root node. This data collection process also is required by the class responsible for filtering the input data during selection. `vtkPBivariateLinearTableThreshold` is responsible for accumulating all selected rows on the root node.

The parallel classes are contained in `vtkSMParallelCoordinatesHistogram-RepresentationProxy`. This class places each filter on the server and connects them to `vtkClientServerMoveData` proxies responsible for delivering their output from the root node to the client. The delivered results are piped from the root node into the client-side `vtkPVParallelCoordinatesHistogramRepresentation`. The "PV" version of the class differs from `vtkParallelCoordinatesHistogramRepresentation` in that it does not contain histogram or outlier filters and does not touch the source data.

5. Future Work. The histogram-based outlier detection filter currently does not always produce expected results. For example, changing the number of histogram bins by small increments causes some outliers to disappear and reappear seemingly at random. A more robust outlier filter would be useful. The median filter-based algorithm is based on one of the options proposed by Novotný and Hauser, however they also advocate the use of histogram clustering as a means of detecting both value clusters and outliers. Such complex techniques will likely lead to more robust outliers at the cost of user-interactivity.

It is often useful to link scatter plots to parallel coordinates plots during visual analysis. Scatter plot classes were recently added to the ParaView development trunk, so work into linking these two visualization techniques within ParaView would be useful. Additionally, scatter plots are currently only available within ParaView; writing VTK versions of these classes would be extremely beneficial to the visualization developer community.

The parallel coordinates plots discussed so far all assume certain data values. Uncertainty visualization has recently become an active research area, and some research has already begun into incorporating uncertainty into both parallel coordinates and scatter plots. The now existing implementations of such plots will serve as useful starting points for the development of new, uncertainty-aware visualization techniques.

6. Acknowledgments. I would like to thank all of the members of organization 1424 in Sandia and the staff at Kitware, Inc. for their support during the development of this software. Timothy Shead and Kenneth Moreland in particular provided invaluable guidance into the Titan and ParaView architectures. Finally, thanks go to Andrew Wilson, my mentor and officemate for the summer, and David Rogers for giving me the opportunity to work in such a positive, productive environment.

REFERENCES

[1] M. d'Ocagne, *Coordonnées Parallèles et Axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considration des coordonnées parallèles*, Paris: Gauthier-Villars, 1885.

[2] M. Graham and J. B. Kennedy, *Using curves to enhance parallel coordinate visualisations*, in IV, 2003, pp. 10–16.

[3] H. Hauser, F. Ledermann, and H. Doleisch, *Angular brushing of extended parallel coordinates*, in in: Proceedings of IEEE Symposium on Information Visualization, IEEE Computer Society Press, 2002, pp. 127–130.

[4] A. Inselberg, *The plane with parallel coordinates*, The Visual Computer, V1 (1985), pp. 69–91.

[5] R. Moustafa and E. Wegman, Springer New York, ch. Multivariate Continuous Data: Parallel Coordinates.

[6] M. Novotny and H. Hauser, *Outlier-preserving focus+context visualization in parallel coordinates*, IEEE Transactions on Visualization and Computer Graphics, 12 (2006), pp. 893–900.

GLOBAL SENSITIVITY ANALYSIS FOR STOCHASTIC COLLOCATION EXPANSION

GARY TANG*, M.S. ELDRED†, AND LAURA P. SWILER‡

Abstract. Non-intrusive stochastic expansion methods for uncertainty quantication (UQ) has received a great deal of attention the past decade because of their rigorous mathematical foundations and their ability to efficiently accurately characterize the probablilistic metrics of complex engineering systems. Furthermore, their formulations are especially amenable to variance-based global sensitivity analysis, and in this paper, we describe a novel method to obtain variance-based sensitivity information as a post-processing procedure to the construction of these stochastic expansion models.

1. Introduction. Modern analysis and design of engineering systems are typically performed with the use of large-scale computer simulations. As a part of understanding model behavior, the ability to characterize relationships between the inputs (whose values are often uncertain) and the output is important. Because of model complexity and implementation nuance this relationship, commonly known as output sensitivity or simply sensitivity, is often obscured. The goal of sensitivity analysis is precisely to identify the most significant factors or variables affecting the model predictions or results. This is not to be confused with uncertainty analysis, which is to quantify the uncertainty in model results due to the uncertainty in the inputs; for sensitivitiy analysis, inputs need not be uncertain. The greatest barrier to conducting either of these analyses is the computational cost to run the simulations. In fact, the popularity of stochastic expansion methods can be attributed almost entirely to their ability to accurately quantify system uncertainty with much fewer function evaluations than traditional sampling methods. Following variance-based approaches to global sensitivity analysis, we present an effective strategy for computing input sensitivities that does not require additional function evaluations beyond those needed to construct the stochastic expansion. In this paper, we focus on the stochastic collocation (SC) class of expansion models.

The remaining sections of this paper outline the basics of variance-based sensitivity analysis and stochastic collocation followed by the specific formulation of sensitivity indices for a stochastic collocation representation. In particular, Section 2 presents the notation and formulation used for variance-based decomposition and definitions for the sensitivity metrics. Section 3 discusses the fundamentals of SC for uncertainty quantification (UQ) and includes a detailed derivation for the sensitivity metrics within the machinery of SC. Finally, Section 4 provides results for a variety of response functions which compare this paper's formulation with analogous work in polynomial chaos expansion [1] and Latin Hypercube sampling. Section 5 concludes with some suggestions for future work that can further couple the relationship between SA and UQ.

2. Variance-Based Sensitivity Analysis. Traditional sensitivity analysis examines the local influence of a parameter x_j to a response $Y(\mathbf{x})$ by its partial derivative at a single point in the parameter space. Alternatively, global sensitivity analysis (GSA) captures the effect of a parameter by measuring some aggregate contribution over the entire space. Using variance as an indicator for importance is not new and can be shown to underlie the regression based methods[2, 3, 7] for sensitivity analysis. This section begins with some general concepts to variance-based GSA and concludes with the definitions for sensitivity as defined in the method of Sobol'. For details on other variance based methods, the authors refer readers to

*Aeronautics and Astronautics, Stanford University, garytang@stanford.edu
†Principal Member of Technical Staff, Sandia National Laboratories, mseldre@sandia.gov
‡Principal Member of Technical Staff, Sandia National Laboratories, lpswile@sandia.gov

the literature.[8]

2.1. Notation. Here, we introduce some notation that will be used in the rest of the paper. Let u be a multi-index such that $u \subseteq \mathcal{U}$, where $\mathcal{U} = \{1, 2, \ldots, d\}$. Let us also define $x_u = \{x_i \,|\, \forall i \in u\}$ as the set of variables whose indices lies in u, and $\hat{f}_u$ to be a basis function that depends only on x_u. Furthermore, let u' be the complement of u and defined such that $\{u \bigcup u'\} = \mathcal{U}$ and $\{u \bigcap u'\} = \emptyset$. Finally, define the collection of all u to be the power set $\mathcal{F} = \mathcal{P}(\mathcal{U})$.

2.2. ANOVA Decomposition. Consider a square-integrable function $f(\mathbf{x}) : \Omega \mapsto \mathbb{R}$, where $\mathbf{x} = (x_1, x_2, \ldots x_d)$ may include both non-probabilistic variables and probabilistic variables, is decomposed in the form:

$$f(\mathbf{x}) = \sum_{\mathcal{F}} \hat{f}_u(x_u) \tag{2.1}$$

The decomposition is of analysis-of-variance (ANOVA) type if it satisfies the following properties: *

$$\int \hat{f}_u(x_u) d\mu(x_u) = 0 \quad \text{for} \quad u \neq \emptyset \tag{2.2}$$

$$\int \hat{f}_u(x_u)\hat{f}_v(x_v) d\mu(\mathbf{x}) = 0 \quad \text{for} \quad u \neq v \tag{2.3}$$

$$\text{Var}[f] = \sum_{u \in \mathcal{F} \neq \emptyset} \text{Var}[\hat{f}_u] \tag{2.4}$$

It is simple to show that these properties are satisfied when the basis function $\hat{f}_u$ is defined as

$$\hat{f}_u = \begin{cases} \int f(\mathbf{x}) d\mu(x_{u'}) - \sum_{w \subset u} \hat{f}_w(x_w) & \text{for} \quad u \neq \emptyset \\ \int f(\mathbf{x}) d\mu(\mathbf{x}) & \text{for} \quad u = \emptyset \end{cases} \tag{2.5}$$

2.3. Global Sensitivity Analysis via Method of Sobol'. In GSA, the general approach to measuring the importance of a parameter x_i is to compare its variance of the conditional expectation, $\text{Var}_{X_i}[\text{E}(Y \,|\, x_i)]$, against the total variance, $\text{Var}[Y]$. Sobol' [12] generalized this idea to the ANOVA decomposition, which allows one to measure the importance of x_u via the variance of $\hat{f}_u$; namely, its contribution to the total variance of f. These sensitivity measures, collectively known as Sobol' indices, are called main effect indices S_u and total effect indices S_{T_u} respectively defined to be:

$$S_u = \frac{D_u}{D} \tag{2.6}$$

$$S_{T_u} = \sum_{\{v \in \mathcal{F} \,|\, u \subseteq v\}} \frac{D_v}{D} \tag{2.7}$$

*let $\mathrm{d}\mu(x)$ be a normalized Lebesgue measure for non-probabilistic variables

where

$$D = \text{Var}[f] \tag{2.8}$$

$$\begin{aligned} D_{u\neq\emptyset} &= \text{Var}[\hat{f}_u] \\ &= \int \hat{f}_u^2 d\mu(x_u) - \left(\int \hat{f}_u d\mu(x_u)\right)^2 \\ &= \int \hat{f}_u^2 d\mu(x_u) \end{aligned} \tag{2.9}$$

Using the orthogonality property of the ANOVA decomposition in Equation 2.3 we can show that the partial variance D_u simplifies to

$$D_u = \int \left(\int f(\mathbf{x}) d\mu(x_{u'})\right)^2 d\mu(x_u) - \sum_{w\subset u} \int \left(\hat{f}_w(x_w)\right)^2 d\mu(x_u) \tag{2.10}$$

where the details can be found in Appendix 7.1.

These indices have very intuitive interpretations. S_u measures the main effect of x_u by evaluating the variance contribution of the basis function $\hat{f}_u$ that depends *strictly* on the set of variables in x_u. Similarly, S_{T_u} measures the total effect of x_u by evaluating the variance contribution of *all* basis functions $\hat{f}_v$ whose dependencies include x_u. In practice, we are typically concerned only with the total effect of sets u which have a cardinality of one, and thus, henceforth we will denote the total effect to be S_{T_j} for $j = 1, 2, \ldots, d$.

3. Stochastic Collocation Expansion. The stochastic collocation (SC) method is an attractive technique for uncertainty quantification (UQ) due to its strong mathematical basis and ability to produce functional representations of stochastic variability. Moreover, because the probability space is merely an extension of the existing parameter space, SC has the flexibility to expand over non-probabilistic variables as well.[15] SC forms interpolation functions by evaluating the model at prescribed collocation point sets derived from tensor product or sparse grids. More specifically, the SC expansion is formed as a weighted tensor product of one-dimensional Lagrange polynomials l_j^i. Because l_j^i has the feature of being equal to 1 at its particular collocation point j and 0 at all other points, the coefficients, or weights, of the expansion are just the corresponding response values $f(x_j^i)$. For d variables and m_{i_k} collocation points in dimension i_k the multi-dimensional expansion can be written as:

$$f(\mathbf{x}) \approx \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d})(l_{j_1}^{i_1} \otimes \cdots \otimes l_{j_d}^{i_d}) \tag{3.1}$$

The key to maximizing performance with this approach is to use the appropriate abscissas defined by Gauss quadrature rules. Greater theoretical detail on this topic can be found in the literature [9, 10] so we only present the most salient result relevant to this paper: the weighted integral of a polynomial of up to order $2n-1$ can be computed exactly using an n term summation when the abscissas are appropriately chosen. More accurately, the abscissas are chosen to the n zeros $\{\xi_i\}_{i=1}^n$ of the n^{th} order polynomial ψ_n^p belonging to the family of polynomials $\{\psi_i^p\}_{i=0}^\infty$ that are optimal for a given weight function $p(x)$.

$$\int g(x)p(x)dx = \sum_{i=1}^{n} w_i g(\xi_i),\ g \in \mathbb{P}^m,\ m \leq 2n-1 \tag{3.2}$$

$$\psi_n^p(\xi_i) = 0, \quad i = 1, 2, \ldots n \tag{3.3}$$

Following these rules, it can be shown [11] that the stochastic expansion converges exponentially in L_2. Once the optimal polynomial basis is determined, the set of $(m_{i_k} - 1)^{th}$ order, one-dimensional Lagrange polynomials can be constructed by the expression

$$l^{i_k}_{j_k}(x^{i_k}) = \prod_{s=1 \neq j_k}^{m_{i_k}-1} \frac{x^{i_k} - \xi^{i_k}_s}{\xi^{i_k}_{j_k} - \xi^{i_k}_s}, \quad j = 1, 2, \ldots, m_{i_k} \tag{3.4}$$

where ξ^{i_k} represents the m_{i_k} zeros of the $\psi^{p_{i_k}}_{m_{i_k}}$. Finally, Equation 3.1 is obtained by taking the tensor product over each dimension $i_k = 1, 2, \ldots, d$. Note that the analytic moments (e.g. mean and variance) and the sensitivity indices can be computed without actually constructing these interpolants.

3.1. Computing the Sobol' Indices. We assume that a set of abscissa and related function evaluations have been performed. The most critical part of computing the Sobol' indices is evaluating the nested integral in Equation 2.10. Let's first evaluate the inner integral with our interpolatory representation of f.

$$\int f(\mathbf{x}) d\mu(x_{u'}) = \int \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})(l^{i_1}_{j_1} \otimes \cdots \otimes l^{i_d}_{j_d}) d\mu(x_{u'}) \tag{3.5}$$

For notational consistency, let's separate those interpolants that are dependent on x_u from those that are dependent on $x_{u'}$ and integrate over $x_{u'}$ using Gauss quadrature rules. Consider some $u \in \mathcal{F}$ with cardinality k

$$\begin{aligned} \int f(\mathbf{x}) d\mu(x_{u'}) &= \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d}) \int (\mathbf{l}^{u'} \otimes \mathbf{l}^{u}) d\mu(x_{u'}) \\ &= \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})(\mathbf{w}^{u'} \otimes \mathbf{l}^{u}) \end{aligned} \tag{3.6}$$

where

$$u = \{u_1, \ldots, u_k\} \tag{3.7}$$

$$\mathbf{w}^{u} = w^{i_{u_1}}_{j_{u_1}} \otimes \cdots \otimes w^{i_{u_k}}_{j_{u_k}} \tag{3.8}$$

$$\mathbf{l}^{u} = l^{i_{u_1}}_{j_{u_1}} \otimes \cdots \otimes l^{i_{u_k}}_{j_{u_k}} \tag{3.9}$$

We can simplify the form of Equation 3.6 by combining the tensor product of weights $\mathbf{w}^{u'}$ with the coefficients $f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})$ to form new weighted coefficients $h(x^{i_{u_1}}_{j_{u_1}}, \ldots, x^{i_{u_k}}_{j_{u_k}})$.

$$\begin{aligned} \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})(\mathbf{w}^{u'} \otimes \mathbf{l}^{u}) &= \sum_{j_{u_1}=1}^{m_{i_{u_1}}} \cdots \sum_{j_{u_k}=1}^{m_{i_{u_k}}} \left(\sum_{j_{u'_1}=1}^{m_{i_{u'_1}}} \cdots \sum_{j_{u'_{d-k}}=1}^{m_{i_{u'_{d-k}}}} f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d}) \mathbf{w}^{u'} \right) (\mathbf{l}^{u}) \\ &= \sum_{j_{u_1}=1}^{m_{i_{u_1}}} \cdots \sum_{j_{u_k}=1}^{m_{i_{u_k}}} h(x^{i_{u_1}}_{j_{u_1}}, \ldots, x^{i_{u_k}}_{j_{u_k}})(\mathbf{l}^{u}) \end{aligned} \tag{3.10}$$

Now, let us evaluate the outer integral. Following the aforementioned result from Gauss quadrature that states an m_{i_k} term summation can exactly integrate any univariate polynomial of degree less than $2m_{i_k}$, we can *exactly* integrate the square of Equation 3.10 since

its highest order univariate polynomial is of degree $2(m_{i_k} - 1)$. Furthermore, if we note the one-dimensional Lagrangian interpolants, l^i_j, to have the property

$$l^q_r(x^q_p) \cdot l^q_s(x^q_p) = \begin{cases} 1 & p = s = r \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

the outer integral can be simplified to

$$\int \left(\sum_{j_{u_1}=1}^{m_{i_{u_1}}} \cdots \sum_{j_{u_k}=1}^{m_{i_{u_k}}} h(x^{i_{u_1}}_{j_{u_1}}, \ldots, x^{i_{u_k}}_{j_{u_k}})(\mathbf{l}^u) \right)^2 d\mu(x_u) = \sum_{j_{u_1}=1}^{m_{i_{u_1}}} \cdots \sum_{j_{u_k}=1}^{m_{i_{u_k}}} h^2(x^{i_{u_1}}_{j_{u_1}}, \ldots, x^{i_{u_k}}_{j_{u_k}}) \otimes \mathbf{w}^u \tag{3.12}$$

and when substituted back into Equation 2.10 we obtain the final result

$$D_u = \sum_{j_{u_1}=1}^{m_{i_{u_1}}} \cdots \sum_{j_{u_k}=1}^{m_{i_{u_k}}} h^2(x^{i_{u_1}}_{j_{u_1}}, \ldots, x^{i_{u_k}}_{j_{u_k}}) \otimes \mathbf{w}^u - \sum_{w \subset u} D_w \tag{3.13}$$

Clearly, the evaluation of D_u is simply a recursive computation of Equation 3.12. To minimize computational effort, it is recommended to solve for the partial variances of u in order of increasing cardinality. The total effect indices S_{T_j} can be trivially computed by summing over the corresponding main effect indices S_u.

This result shows that D_u, and by virtue S_u, can be evaluated without any additional function evaluations or the need to solve for additional weights/abscissas. Moreover, this methodology extends to sparse grids, which are merely linear combinations of tensor product grids, provided that the abscissas of the tensor product grids follow Gauss quadrature rules.

4. Implementation and Results. The analytic expressions for sensitivity presented in this paper were implemented and verified in Sandia National Labs' software framework DAKOTA (Design and Analysis Toolkit for Terascale Applications)[13], and the source code can be obtained from the version of the day release.

A series of tests were conducted to compare the method in this paper with existing approaches for GSA. For the first four tests, the test functions were chosen to be simple polynomials whose sensitivities could be obtained analytically. The remaining three functions are common for testing sensitivity methods.[7] For each test, we compare the performance of our approach to traditional Latin Hypercube sampling (LHS) and to an analogous implementation for generalized polynomial chaos expansion[1] (PCE) on a variety of tensor (tr[order]) and sparse (sp[level]) grids.

The results show that GSA via stochastic expansion methods dramatically outperforms LHS and highlights the excellent convergence rates when the functions are of sufficient regularity. While this trend is unlikely to extend to higher dimensional problems where expansion methods suffer from the curse of dimensionality, this problem may be mitigated with the use of anisotropic grids.[15] For nearly all the cases, we see identical results for PCE and SC when the abscissas are chosen from tensor grids, which is consistent with their behavior in computing analytic moments.[14] The exception can be found in Table 7 but this is likely a consequence of computational roundoff as opposed to solution disagreement. A related consequence is the possibility of negative sensitivity indices; the specific formulation to the PCE approach does not require subtraction and eliminates the possibility of castastrophic cancellation. As mentioned in Section 3.1, the evaluation of the Sobol' Indices are exact with respect to the SC representation presented in this paper; therefore, the accuracy and convergence behavior of the stochastic expansion models are transferred, without loss, to the sensitivity

TABLE 4.1
Sobol' Indices for Rosenbrock function $f = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

Approach	Func Eval	Grid Type	S_1	S_2	S_{T_1}	S_{T_2}
LHS	400	-	4.06127e-01	3.72295e-01	5.83464e-01	8.63159e-01
LHS	2000	-	2.84753e-01	3.72392e-01	6.24346e-01	7.86900e-01
LHS	4000	-	1.37274e-01	3.12069e-01	7.15033e-01	8.58973e-01
LHS	20000	-	1.73756e-01	2.82603e-01	7.09529e-01	8.58973e-01
LHS	40000	-	1.53909e-01	3.03002e-01	7.02275e-01	8.34521e-01
LHS	200000	-	1.47978e-01	2.79160e-01	7.21528e-01	8.46659e-01
PCE	4	tr2	2.09627e-01	1.97593e-01	8.02407e-01	7.90373e-01
PCE	9	tr3	1.20599e-01	2.93134e-01	7.06866e-01	8.79401e-01
PCE	25	tr5	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
PCE	36	tr6	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
PCE	13	sp2	2.84129e-01	7.15871e-01	2.84129e-01	7.15871e-01
PCE	65	sp4	1.52832e-01	2.82389e-01	7.17611e-01	8.47168e-01
PCE	321	sp6	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
PCE	1537	sp8	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
SC	4	tr2	2.09627e-01	1.97593e-01	8.02407e-01	7.90373e-01
SC	9	tr3	1.20599e-01	2.93134e-01	7.06866e-01	8.79401e-01
SC	25	tr5	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
SC	36	tr6	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
SC	13	sp2	1.54653e-01	2.90070e-01	7.09930e-01	8.45347e-01
SC	65	sp4	1.53198e-01	2.82267e-01	7.17733e-01	8.47802e-01
SC	321	sp6	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
SC	1537	sp8	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01
TRUE	-	-	1.53198e-01	2.82267e-01	7.17733e-01	8.46802e-01

TABLE 4.2
Sobol' Indices for $f = (x_1 - 1)^4 + (x_2 - 1)^4$

Approach	Func Eval	Grid Type	S_1	S_2	S_{T_1}	S_{T_2}
LHS	400	-	4.64092e-01	5.41070e-01	5.41070e-01	4.95429e-01
LHS	2000	-	4.76199e-01	5.24702e-01	5.24412e-01	5.04126e-01
LHS	4000	-	4.87648e-01	5.12858e-01	4.91029e-01	5.12199e-01
LHS	20000	-	5.02428e-01	4.97679e-01	5.09053e-01	5.02952e-01
LHS	40000	-	4.89737e-01	5.10320e-01	4.93644e-01	4.91893e-01
LHS	200000	-	4.95960e-01	5.04051e-01	4.99884e-01	4.97408e-01
PCE	4	tr2	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
PCE	9	tr4	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
PCE	25	tr6	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
PCE	36	tr8	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
PCE	17	sp2	4.95299e-01	4.95299e-01	5.04701e-01	5.04701e-01
PCE	97	sp4	4.99964e-01	4.99964e-01	5.00036e-01	5.00036e-01
PCE	305	sp6	4.99964e-01	2.82267e-01	5.00036e-01	5.00036e-01
PCE	705	sp8	4.99985e-01	4.99985e-01	5.00015e-01	5.00015e-01
SC	4	tr2	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
SC	9	tr4	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
SC	25	tr6	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
SC	36	tr8	5.00000e-01	5.00000e-01	5.00000e-01	5.00000e-01
SC	17	sp2	5.28463e-01	5.28463e-01	4.81537e-01	4.81537e-01
SC	97	sp4	4.98911e-01	4.98911e-01	5.01089e-01	5.01089e-01
SC	305	sp6	5.00132e-01	5.00132e-01	4.99868e-01	4.99868e-01
SC	705	sp8	4.99987e-01	4.99987e-01	5.00013e-01	5.00013e-01
TRUE	-	-	5.00000e-01	5.00000e-01	5.00000e-01	5.0000e-01

analysis. It follows that performing GSA, like solving for mean or variance, will converge at a reduced rate for non-smooth functions in L_2. An example of reduced performance can be seen in Table 4.7 for the non-smooth Sobol' g-Function.

5. Conclusions. This paper presented a computationally effective approach to global sensitivity analysis on stochastic collocation models. Retaining the high levels of accuracy and exponential rates of convergence that have made stochastic expansion methods preferred over their counterparts, this approach allows sensitivity indices to be computed as a simple post-processing procedure to the construction of the expansion. Furthermore, inexpensive

TABLE 4.3
Sobol' Indices for $f = x_1^2 - \frac{x_2}{2}$

Approach	Func Eval	Grid Type	S_1	S_2	S_{T_1}	S_{T_2}
LHS	400	-	8.11961e-01	1.88291e-01	8.21556e-01	1.73615e-01
LHS	2000	-	8.21845e-01	1.78386e-01	8.24890e-01	1.84108e-01
LHS	4000	-	8.05006e-01	1.95046e-01	7.98831e-01	1.99964e-01
LHS	20000	-	8.16281e-01	1.83728e-01	8.01936e-01	1.90491e-01
LHS	40000	-	8.09067e-01	1.90941e-01	8.11791e-01	1.86326e-01
LHS	200000	-	8.10709e-01	1.89293e-01	8.09960e-01	1.87076e-01
LHS	400000	-	8.09877e-01	1.90123e-01	8.09727e-01	1.91407e-01
PCE	4	tr2	8.00000e-01	2.00000e-01	8.00000e-01	2.00000e-01
PCE	16	tr4	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
PCE	36	tr6	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
PCE	64	tr8	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
PCE	13	sp2	8.00000e-01	2.00000e-01	8.00000e-01	2.00000e-01
PCE	65	sp4	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
PCE	321	sp6	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
PCE	1537	sp8	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	4	tr2	8.00000e-01	2.00000e-01	8.00000e-01	2.00000e-01
SC	16	tr4	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	36	tr6	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	64	tr8	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	13	sp2	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	65	sp4	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	321	sp6	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
SC	1537	sp8	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01
TRUE	-	-	8.10127e-01	1.89873e-01	8.10127e-01	1.89873e-01

TABLE 4.4
Sobol' Indices for $f = x_2^2 - \frac{x_1}{2}$

Approach	Func Eval	Grid Type	S_1	S_2	S_{T_1}	S_{T_2}
LHS	400	-	1.67513e-01	8.33176e-01	1.61058e-01	8.13480e-01
LHS	2000	-	1.79156e-01	8.21072e-01	1.91097e-01	8.34315e-01
LHS	4000	-	1.77226e-01	8.22785e-01	1.82247e-01	8.31742e-01
LHS	20000	-	1.98623e-01	8.01388e-01	1.89664e-01	8.11840e-01
LHS	40000	-	1.82394e-01	8.17612e-01	1.82364e-01	8.14272e-01
LHS	200000	-	1.88671e-01	8.11330e-01	1.87260e-01	8.08251e-01
LHS	400000	-	1.91731e-01	8.08270e-01	1.91781e-01	8.11015e-01
PCE	4	tr2	2.00000e-01	8.00000e-01	2.00000e-01	8.00000e-01
PCE	16	tr4	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
PCE	36	tr6	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
PCE	64	tr8	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
PCE	13	sp2	2.00000e-01	8.00000e-01	2.00000e-01	8.00000e-01
PCE	65	sp4	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
PCE	321	sp6	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
PCE	1537	sp8	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	4	tr2	2.00000e-01	8.00000e-01	2.00000e-01	8.00000e-01
SC	16	tr4	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	36	tr6	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	64	tr8	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	13	sp2	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	65	sp4	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	321	sp6	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
SC	1537	sp8	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01
TRUE	-	-	1.89873e-01	8.10127e-01	1.89873e-01	8.10127e-01

availability of this information presents an attrative route to adaptive grid refinement, helping to ameliorate the curse of dimensionality and subequently improving the efficacy of stochastic expansions methods.

6. Acknowledgments. We would like to thank Sandia National Laboratories for their generous financial support.

7. Appendix.

TABLE 4.5
Sobol' Indices for $f = \frac{(x_2+0.5)^4}{(x_1+0.5)^2}$

Approach	Func Eval	Grid Type	S_1	S_2	S_{T_1}	S_{T_2}
LHS	400	-	6.22785e-01	6.56084e-01	2.07789e-01	9.07812e-01
LHS	2000	-	3.29550e-01	5.28888e-01	4.88533e-01	7.41134e-01
LHS	4000	-	2.45757e-01	5.12369e-01	5.00761e-01	8.04966e-01
LHS	20000	-	2.65514e-01	4.98655e-01	5.14580e-01	7.86911e-01
LHS	40000	-	2.78471e-01	5.25235e-01	4.61501e-01	7.41430e-01
LHS	200000	-	2.64119e-01	5.19057e-01	4.87495e-01	7.40763e-01
PCE	4	tr2	2.43001e-00	5.89552e-01	4.10448e-01	7.56999e-01
PCE	9	tr3	2.52312e-01	5.30029e-01	4.69971e-01	7.47688e-01
PCE	25	tr5	2.61703e-01	5.11375e-01	4.88625e-01	7.38296e-01
PCE	36	tr6	2.61888e-01	5.11029e-01	4.88971e-01	7.38111e-01
PCE	13	sp2	3.29508e-01	6.70492e-01	3.29508e-01	6.70492e-01
PCE	65	sp4	2.63513e-01	5.15699e-01	4.84301e-01	7.36487e-01
PCE	321	sp6	2.61916e-01	5.10990e-01	4.89010e-01	7.38084e-01
PCE	1537	sp8	2.61914e-01	5.10983e-01	4.89017e-01	7.38086e-01
SC	4	tr2	2.43001e-00	5.89552e-16	4.10448e-01	7.56999e-01
SC	9	tr3	2.52312e-01	5.30029e-01	4.69971e-01	7.47688e-01
SC	25	tr5	2.61703e-01	5.11375e-01	4.88625e-01	7.38296e-01
SC	36	tr6	2.61888e-01	5.11029e-01	4.88971e-01	7.38111e-01
SC	13	sp2	2.48662e-01	4.28701e-01	5.71299e-01	7.51338e-01
SC	65	sp4	2.61310e-01	5.10192e-01	4.89808e-01	7.38690e-01
SC	321	sp6	2.61914e-01	5.10983e-01	4.89017e-01	7.38086e-01
SC	1537	sp8	2.61914e-01	5.10983e-01	4.89017e-01	7.38086e-01

TABLE 4.6
Sobol' Indices for the Ishigami Function $f = \sin(2\pi x_1 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 0.1(2\pi x_3 - \pi)^4 \sin(2\pi x_1 - \pi)$

Approach	Func Eval	Grid Type	S_1	S_2	S_3	S_{T_1}	S_{T_2}	S_{T_3}
LHS	5000	-	2.88446e-01	4.42871e-01	5.52915e-03	5.93029e-01	4.00977e-01	2.32240e-01
LHS	25000	-	3.05843e-01	4.37050e-01	-1.87194e-02	5.66739e-01	4.30240e-01	2.48935e-01
LHS	50000	-	3.28016e-01	4.34040e-01	-8.27517e-03	5.61866e-01	4.45603e-01	2.41798e-01
LHS	250000	-	3.15380e-01	4.42334e-01	5.15642e-03	5.55366e-01	4.43332e-01	2.46431e-01
LHS	500000	-	3.14020e-01	4.42135e-01	-9.13024e-04	5.60501e-01	4.33634e-01	2.36443e-01
PCE	8	tr2	1.00000e-00	3.62082e-32	3.62082e-32	1.00000e-00	7.01534e-32	2.12723e-31
PCE	27	tr3	4.15063e-01	4.39941e-01	2.69333e-32	5.60059e-01	4.39941e-01	1.44995e-01
PCE	64	tr4	3.99679e-01	3.15380e-01	2.17138e-32	6.84620e-01	3.15380e-01	2.84941e-01
PCE	216	tr6	4.05169e-01	2.80300e-01	1.56108e-30	7.19700e-01	2.80300e-01	3.14531e-01
PCE	25	sp2	1.00000e-00	1.36887e-31	0.00000e-00	1.00000e-00	1.36887e-31	0.00000e-00
PCE	177	sp4	6.71159e-01	4.44492e-02	1.83743e-07	9.55551e-01	9.59348e-02	2.32906e-01
PCE	1073	sp6	3.69705e-01	3.54341e-01	3.14446e-19	6.45659e-01	3.54790e-01	2.75516e-01
PCE	6017	sp8	3.16646e-01	4.37727e-01	1.91006e-31	5.62273e-01	4.37727e-01	2.45627e-01
PCE	32001	sp10	3.13906e-01	4.42411e-01	2.04501e-30	5.57589e-01	4.42411e-01	2.43684e-01
SC	8	tr2	1.00000e-00	0.00000e-00	-1.73938e-15	1.00000e-00	0.00000e-00	1.73938e-15
SC	27	tr3	4.15063e-01	4.39941e-01	9.02345e-17	5.60059e-01	4.39941e-01	1.44995e-01
SC	64	tr4	3.99679e-01	3.15380e-01	-4.35361e-16	6.84620e-01	3.15380e-01	2.84941e-01
SC	216	tr6	4.05169e-01	2.80300e-01	1.65375e-16	7.19700e-01	2.80300e-01	3.14531e-01
SC	25	sp2	6.46098e-02	9.35390e-01	0.00000e-00	6.46098e-02	9.35390e-01	0.00000e-00
SC	177	sp4	3.38208e-01	3.53492e-01	-5.12588e-16	6.46508e-01	3.53492e-01	3.08299e-01
SC	1073	sp6	3.13909e-01	4.42416e-01	1.53970e-15	5.57584e-01	4.42416e-01	2.43675e-01
SC	6017	sp8	3.13905e-01	4.42411e-01	-1.41137e-15	5.57589e-01	4.42411e-01	2.43684e-01
SC	32001	sp10	3.13905e-01	4.42411e-01	1.59101e-14	5.57589e-01	4.42411e-01	2.43684e-01

7.1. Derivation for Result in Equation 2.10. Firstly, expand the square of the integrand

$$\begin{aligned}
\hat{f}_u^2 &= \left(\int f(\mathbf{x}) d\mu(x_{u'}) - \sum_{w \subset u} \hat{f}_w(x_w) \right)^2 \\
&= \left(\int \sum \hat{f}_u d\mu(x'_u) - \sum_{w \subset u} \hat{f}_w(x_w) \right)^2 \\
&= \left(\int \sum \hat{f}_u d\mu(x'_u) \right)^2 - 2 \int \sum \hat{f}_u d\mu(x'_u) \sum_{w \subset u} \hat{f}_w(x_w) + \left(\sum_{w \subset u} \hat{f}_w(x_w) \right)^2
\end{aligned}$$

TABLE 4.7
Sobol' Indices for the Sobol' g-Function $f = 2\prod_{j=1}^{5} \frac{|4x_j - 2| + a_j}{1 + a_j}$; $\mathbf{a} = [0, 1, 2, 4, 8]$

Approach	Func Eval	Grid Type	S_1	S_2	S_3	S_4	S_5
LHS	7000	-	5.91333e-01	1.47510e-01	6.75943e-02	3.35949e-02	-3.03280e-03
LHS	35000	-	6.12067e-01	1.59304e-01	6.93052e-02	3.49980e-02	5.59985e-03
LHS	70000	-	6.41981e-01	1.47865e-01	8.06888e-02	2.24580e-02	1.00360e-02
LHS	350000	-	6.35479e-01	1.60470e-01	6.89460e-02	2.34116e-02	7.40881e-03
LHS	700000	-	6.33740e-01	1.61268e-01	7.06319e-02	2.69095e-02	7.70250e-03
PCE	32	tr2	0.00000e-00	0.00000e-00	6.45161e-02	5.80645e-01	6.45161e-02
PCE	243	tr3	5.99272e-01	1.28220e-01	5.42446e-02	1.87888e-02	5.65402e-03
PCE	1024	tr4	6.46390e-01	1.68398e-01	7.58935e-02	2.76298e-02	8.59202e-03
PCE	3125	tr5	6.21874e-01	1.46776e-01	6.40180e-02	2.27051e-02	6.93877e-03
PCE	7776	tr6	6.40868e-01	1.63388e-01	7.30962e-02	2.64539e-02	8.19362e-03
PCE	16807	tr7	6.28497e-01	1.52461e-01	6.70901e-02	2.39619e-02	7.35684e-03
PCE	32678	tr8	6.38803e-01	1.61536e-01	7.20691e-02	2.60245e-02	8.04870e-03
PCE	61	sp2	1.00000e-00	0.00000e-00	0.00000e-00	0.00000e-00	0.00000e-00
PCE	801	sp4	8.09099e-01	1.47072e-01	3.03062e-02	3.10093e-06	1.35196e-02
PCE	6993	sp6	5.90683e-01	1.42680e-01	6.88667e-02	4.81179e-02	5.71842e-02
PCE	51713	sp8	6.33501e-01	1.56378e-01	6.71917e-02	2.25731e-02	7.56251e-03
PCE	135073	sp9	6.38484e-01	1.58953e-01	6.96320e-02	2.40071e-02	6.87157e-03
PCE	345665	sp10	6.35135e-01	1.58349e-01	6.99588e-02	2.49043e-02	7.84157e-03
SC	32	tr2	7.33333e-01	7.33333e-01	7.33333e-01	7.33333e-01	7.33333e-01
SC	243	tr3	5.99272e-01	1.28220e-01	5.42446e-02	1.87888e-02	5.65402e-03
SC	1024	tr4	6.46390e-01	1.68398e-01	7.58935e-02	2.76298e-02	8.59202e-03
SC	3125	tr5	6.21874e-01	1.46776e-01	6.40180e-02	2.27051e-02	6.93877e-03
SC	7776	tr6	6.40868e-01	1.63388e-01	7.30962e-02	2.64539e-02	8.19362e-03
SC	16807	tr7	6.28497e-01	1.52461e-01	6.70901e-02	2.39619e-02	7.35684e-03
SC	32768	tr8	6.38803e-01	1.61536e-01	7.20691e-02	2.60245e-02	8.04870e-03
SC	61	sp2	6.12891e-01	-9.96566e-02	-1.72524e-01	-1.90741e-01	-1.95295e-01
SC	801	sp4	4.90716e-01	-1.11982e-01	-2.18710e-01	-2.67265e-01	-2.83116e-01
SC	6993	sp6	5.67804e-01	1.24812e-02	-9.42399e-02	-1.50300e-01	-1.71939e-01
SC	51713	sp8	6.18682e-01	1.14612e-01	1.90377e-02	-3.11604e-02	-5.12477e-02
SC	135073	sp9	6.28501e-01	1.38023e-01	4.59513e-02	-1.93184e-03	-2.09570e-02
SC	345665	sp10	6.32966e-01	1.49798e-01	5.97042e-02	1.31964e-02	-5.12018e-03

Approach	Func Eval	Grid Type	S_{T_1}	S_{T_2}	S_{T_3}	S_{T_4}	S_{T_5}
LHS	7000	-	6.72206e-01	1.86171e-01	1.17109e-01	5.77715e-03	-9.02123e-03
LHS	35000	-	7.38654e-01	2.29849e-01	1.22343e-01	4.64609e-02	3.02973e-02
LHS	70000	-	7.40672e-01	2.40038e-01	1.11206e-01	5.98694e-02	2.47003e-02
LHS	350000	-	7.22398e-01	2.07503e-01	8.47251e-02	2.15933e-02	-4.94203e-03
LHS	700000	-	7.26557e-01	2.22900e-01	1.04010e-01	3.48873e-02	9.51685e-03
PCE	32	tr2	0.00000e-00	0.00000e-00	3.06452e-01	8.22581e-01	3.06452e-01
PCE	243	tr3	7.77373e-01	2.55631e-01	1.18106e-01	4.27975e-02	1.31030e-02
PCE	1024	tr4	7.11752e-01	2.14784e-01	9.98585e-02	3.69641e-02	1.15712e-02
PCE	3125	tr5	7.46823e-01	2.35794e-01	1.09272e-01	4.00032e-02	1.23773e-02
PCE	7776	tr6	7.19827e-01	2.19470e-01	1.01963e-01	3.76491e-02	1.17544e-02
PCE	16807	tr7	7.37548e-01	2.30066e-01	1.06711e-01	3.91829e-02	1.21614e-02
PCE	32768	tr8	7.22820e-01	2.21229e-01	1.02752e-01	3.79052e-02	1.18227e-02
PCE	61	sp2	1.00000e-00	0.00000e-00	0.00000e-00	0.00000e-00	0.00000e-00
PCE	801	sp4	8.09099e-01	1.47072e-01	3.03062e-02	3.10093e-06	1.35196e-02
PCE	6993	sp6	6.70159e-01	1.93377e-01	9.84211e-02	6.48668e-02	6.56442e-02
PCE	51713	sp8	7.30498e-01	2.21190e-01	1.02726e-01	4.15261e-02	1.91126e-02
PCE	135073	sp9	7.29110e-01	2.22165e-01	1.02688e-01	3.80657e-02	1.26439e-02
PCE	345665	sp10	7.25354e-01	2.22384e-01	1.03870e-01	4.02503e-02	1.53053e-02
SC	32	tr2	4.00000e-01	4.00000e-01	4.00000e-01	4.00000e-01	4.00000e-01
SC	243	tr3	7.77373e-01	2.55631e-01	1.18106e-01	4.27975e-02	1.31030e-02
SC	1024	tr4	7.11752e-01	2.14784e-01	9.98585e-02	3.69641e-02	1.15712e-02
SC	3125	tr5	7.46823e-01	2.35794e-01	1.09272e-01	4.00032e-02	1.23773e-02
SC	7776	tr6	7.19827e-01	2.19470e-01	1.01963e-01	3.76491e-02	1.17544e-02
SC	16807	tr7	7.37548e-01	2.30066e-01	1.06711e-01	3.91829e-02	1.21614e-02
SC	32768	tr8	7.22820e-01	2.21229e-01	1.02752e-01	3.79052e-02	1.18227e-02
SC	61	sp2	1.06778e-00	2.91470e-01	7.28675e-02	1.82169e-02	4.55422e-03
SC	801	sp4	9.89681e-01	3.25014e-01	1.54804e-01	5.61298e-02	1.61487e-02
SC	6993	sp6	8.44473e-01	2.60080e-01	1.23259e-01	4.68687e-02	1.49192e-02
SC	51713	sp8	7.58487e-01	2.31469e-01	1.08062e-01	4.02856e-02	1.27392e-02
SC	135073	sp9	7.41164e-01	2.26819e-01	1.05551e-01	3.90980e-02	1.22739e-02
SC	345665	sp10	7.32933e-01	2.24861e-01	1.04492e-01	3.85827e-02	1.20597e-02

$$\int \hat{f}_u^2 d\mu(x_u) = \int \left(\int \sum \hat{f}_u d\mu(x'_u) \right)^2 d\mu(x_u) - 2 \int \left(\int \sum \hat{f}_u d\mu(x'_u) \sum_{w \subset u} \hat{f}_w(x_w) \right) d\mu(x_u) + \int \left(\sum_{w \subset u} \hat{f}_w(x_w) \right)^2 d\mu(x_u)$$

Now applying the orthogonality property on the second term on the right hand side

$$\begin{aligned} 2 \int \left(\int \sum_{\mathcal{F}} \hat{f}_u d\mu(x'_u) \sum_{w \subset u} \hat{f}_w(x_w) \right) d\mu(x_u) &= 2 \int \int \sum_{\mathcal{F}} \hat{f}_u(x_u) \sum_{w \subset u} \hat{f}_w(x_w) d\mu(x'_u) d\mu(x_u) \\ &= 2 \int \int \sum_{\mathcal{F}} \sum_{w \subset u} \hat{f}_u(x_u) \hat{f}_w(x_w) d\mu(\mathbf{x}) \\ &= 2 \int \sum_{w \subset u} \left(\hat{f}_u(x_u) \right)^2 \end{aligned}$$

and similarly for the third term

$$\int \left(\sum_{w \subset u} \hat{f}_w(x_w) \right)^2 d\mu(x_u) = \sum_{w \subset u} \int \left(\hat{f}_w(x_w) \right)^2 d\mu(x_u)$$

Finally, substituting back into Equation 2.9

$$\begin{aligned} \int \hat{f}_u^2 d\mu(x_u) &= \int \left(\int \sum \hat{f}_u d\mu(x'_u) \right)^2 d\mu(x_u) - \int \sum_{w \subset u} \left(\hat{f}_w(x_w) \right)^2 d\mu(x_u) \\ &= \int \left(\int f(\mathbf{x}) d\mu(x'_u) \right)^2 d\mu(x_u) - \sum_{w \subset u} \int \left(\hat{f}_w(x_w) \right)^2 d\mu(x_u) \end{aligned}$$

REFERENCES

[1] Sudret, Bruno, "Global Sensitivity analysis using polynomial chaos expansion," *Reliability Engr. & Safety System*, 93, 2008, pp.964-979.

[2] Saltelli, A et al., *Sensitivity Analysis*, Wiley, New York, 2001, pp123-120.

[3] Helton, J.C. et al., "Survey of sampling-based methods for uncertainty and sensitivity analysis," *Reliability Engineering and System Safety*, 91, 2006, pp.1175-1209.

[4] Iman, R.L. and J.C. Helton, "An investigation of uncertainty and sensitivity analysis techniques for computer models," *Risk Analysis*, 8, 1998, pp.71-90.

[5] Storlie, C.B. and J.C. Helton. "Multiple predictor smoothing methods for sensitivity analysis: Description of techniques," *Reliability Engineering and System Safety*, 93, 2008, pp.28-54.

[6] McKay, M.D. et al., "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, 21, 1979, pp.239-245.

[7] Storlie, C.B. et al., "Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models," *Reliability Engineering & System Safety*, 94, 11, 2009, pp1735-1763.

[8] Saltelli, Andrea et al., *Global Sensitivity Analysis*, Wiley, New York, 2008, pp.155-183.

[9] Golub, G.H. and J. H. Welch, "Calculation of Gauss Quadrature Rules," *Mathematics of Computation*, 23, 106, 1969, pp.221-230.

[10] W. Gautschi, Algorithm 726: ORTHOPOL-A package of routines for generating orthogonal polynomials and Gauss-type quadrature rules, ACM Trans. Math. Software20, (1994) 2162.

[11] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic di?erential equations," *SIAM J. Sci. Comput.*, 24, 2002, pp.619644.

[12] Sobol', I.M., "Sensitivity estimates for nonlinear mathematical models," *Math. Modeling and Computational Experiments*, 1, 1993, pp.407-414.

[13] DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis, Software Framework, Version 4.2+ VOTD July 13th 2009, Sandia National Labs, Albuquerque, NM, 2009.

[14] Constantine, P. and G. Iaccarino, "Comparing spectral Galerkin and spectral collcoation methods for parameterized matrix equations," Technical Report UQ-08-02, Stanford University, Stanford, CA 2008.

[15] Eldred, M. S., "Recent Advances in Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Analysis and Design," 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Converence, Palm Springs, CA, 2009, AIAA-2009-2274.

Discrete Mathematics and Informatics

Discrete mathematics is the study of fundamentally discrete mathematical structures with application to problems arising in the computing sciences. Likewise, the field of informatics includes processing and reasoning about collected information or data — often to identify associations and to extract knowledge, with the objective to enable an informed decision–making process. Articles in this section encompass both of these disciplines with application to numerical linear algebra, data classification, and combinatorial optimization.

Z. Wen
S.S. Collis

January 11, 2010

SOLVING K-DETECTABILITY SENSOR PLACEMENT PROBLEM WITH MIXED INTEGER PROGRAMMING FORMULATIONS

T.K. FENG*, J.P. WATSON†, R. CARR‡, AND J.C. BECK§

Abstract. We present three Mixed Integer Programming (MIP) formulations for the k-detectability sensor placement optimization in public water distribution network. All three formulations found optimal solutions for small problem sizes. But, they do not scale well for large problem size. Further, we propose a class of problems called k-generalized p-median problem, which is a generalization of the well known p-median facility location problem. We show that the k-detectability sensor placement problem is mathematically equivalent to the k-generalized p-median problem.

1. Introduction. An Early Warning System (EWS) is an online and real-time strategy which has been implemented in Europe and U.S. to identify accidental and intentional contaminations in public water systems [1, 2]. The objective of an EWS is to identify and locate high risk contamination events in a timely fashion. This allows the appropriate response to be executed and protects the public from water contamination disasters.

A critical decision in all EWSs is the strategic placement of contamination-detecting sensors in the water distribution network. Poor placement of sensors results in a waste of the government's tax dollar and puts the public at risk. Therefore, it is crucial to find a robust placement of sensors.

There are a variety of technical approaches to solve the sensor placement problem in the context of EWS. The approaches can be categorized into two major classes: time-dependent formulations and time-independent formulations.

A Time-Dependent Formulation models the problem in a realistic, time-related manner. It often includes explicit physical constraints such as pipe network topology, flow direction, fluid dynamics, and travel time. For example, the "level of disaster" is time-dependent: as time passes after a contamination takes place, the "level of disaster" increases proportionally to time.

A Time-Independent Formulation abstracts time away from the model. Instead, it explores the structural protection from the placement or a combination of placements of sensors. For example, an area that has two sensors within a certain radius is considered to be safe.

Please refer to [5] for a full review of different sensor placement formulations.

We attempt to solve this problem using a Mixed Integer Programming (MIP) approach that belongs to the time-independent class. It is a direct extension of [1]. In that article, the disasters caused by all contaminants at each node are calculated using a water quality simulation software, EPANET, [7]. The disaster matrix is then fed to a MIP solver as the input of the problem. Although the simulation of the disaster matrix is both time and resource consuming, it is a preprocess that can performed beforehand. In [1], the objective is to minimize the overall disaster caused by all contaminants by placing a pre-specified number of sensors in the water distribution network. When a contaminant traverses a node that has a sensor, the EWS detects the contaminant and responds accordingly. The damage or disaster inflicted by this contaminant contributes to the objective.

In our formulation, we take one step further. We consider the detectability of the contaminants. A common feature of a sensor placement formulation is the idealization of the ability

*University of Toronto, tkfeng@mie.utoronto.ca
†Sandia National Laboratories, jwatson@sandia.gov
‡Sandia National Laboratories, rdcarr@sandia.gov
§University of Toronto, jcb@mie.utoronto.ca

of a sensor detecting a contaminant. In reality, no sensor detects contaminants correctly 100% of the time. We introduce the concept of k-detectability. A contaminant is k-detectable when the contaminant traverses at least k nodes that contain sensors before EWS detects the contaminant. When k is 1 for all contaminants, it is essentially the original sensor placement problem studied in [1].

2. Problem Definition. The k-Detectability Sensor Placement Problem requires the placement of p sensors in a closed system with the goal of detecting all the contaminants and hence reducing the level of subsequent disasters. The closed system is composed of a set of connected nodes, V. Each contaminant in the set of all contaminants, $a \in A$, initiates its attack at a node $j \in V$ and diffuses across the closed system. The disaster caused by contaminant a when node j is compromised by a is d_{aj}. When the system detects contaminant a, we assume that the appropriate actions are taken and the contaminant is removed from the system. The objective is to place p sensors in the closed system so the sum of the damages or the level of disasters caused by all contaminants is minimized.

We define the meaning of *contaminant detection* here. In a problem that was previously studied by Berry et al. [1], a contaminant is *detected* after the *first* sensor detects the contaminant. The MIP formulation of the problem is termed as DSP in [1]. In our paper, we expand the definition of *contaminant detection*. The system *detects* contaminant a after it has been detected by at least k_a different sensors. This is the k-Detectability Sensor Placement Problem (k-DSP). Interestingly, the problem studied in [1] is exactly 1-DSP.

3. Formulation. We present three formulations to model k-DSP. Each formulation has its distinct features. Nonetheless, they have some decisions in common. We describe the common decision variables here. s_j is a binary decision variable which equals 1 when a sensor is placed in node $j \in V$ and 0 otherwise. D_a is a continuous decision variable that represents the overall damage caused by contaminant a.

3.1. X Formulation. The X formulation models k-DSP by answering the following questions: where does each individual contamination detection take place and what contamination does it detect? The X formulation is based on binary decision variable, x_{aj}, $a \in A$ and $j \in V$. x_{aj} equals 1 when a sensor placed at node j detects contaminant a and 0 otherwise. Note that when a sensor detects a contaminant, it does not necessarily mean that the contaminant has been detected by the system. In order for the system to detect contaminant a, at least k sensor detections are required. The X formulation, k-DSP.X, is stated as follows:

$$\min_x \sum_{a \in A} D_a \qquad (OX)$$

$$\sum_{j \in V} s_j = p \qquad (X1)$$

$$x_{aj} \le s_j \quad \forall a \in A,\ j \in V \qquad (X2)$$

$$\sum_{j \in V} x_{aj} = k_a \quad \forall a \in A \qquad (X3)$$

$$D_a \ge d_{aj} x_{aj} \quad \forall a \in A,\ j \in V \qquad (X4)$$

Objective (OX) minimizes the sum of overall damage per contaminant. Constraint $(X1)$ ensures the system to have exactly p sensors. In order for a sensor to detect a contaminant at location j, there must be a sensor placed at node j. This condition is enforced by constraint $(X2)$. Furthermore, each contaminant, a, requires exactly k_a sensor detections–this is enforced by constraint $(X3)$. Finally, constraint $(X4)$ relates the damage incurred at each node to the overall damage.

3.2. Y Formulation. The *Y* formulation answers a different question: where does the detection of sensor k_a take place? The binary decision variable, y_{aj}, is 1 when the $k_a th$ sensor detects contaminant a at node j (or, when the system detects contaminant a). The following is the *Y* formulation, k-DSP.Y,

$$\min_{y} \sum_{a \in A} D_a \tag{OY}$$

$$\sum_{j \in V} s_j = p \tag{Y1}$$

$$y_{aj} \leq s_j \quad \forall a \in A,\ j \in V \tag{Y2}$$

$$\sum_{j \in V} y_{aj} = 1 \quad \forall a \in A \tag{Y3}$$

$$D_a = \sum_{j \in V} d_{aj} y_{aj} \quad \forall a \in A \tag{Y4}$$

$$k_a y_{aj} \leq \left(\sum_{i \in V, d_{ai} < d_{aj}} s_i \right) + s_j \quad \forall a \in A,\ j \in V \tag{Y5}$$

Objective (OY) is the same as (OX) with one exception. It minimizes over y. Similarly, constraints $(Y1)$, $(Y2)$, and $(Y3)$ are equivalent to $(X1)$, $(X2)$, and $(X3)$; the y decision variables replace the x variables. Constraint $(Y3)$ sums y to 1 because the system detects a given contaminant, a, once, whereas the sensor detects contaminant a k_a times. Constraint $(Y4)$ relates the overall damage to the damage incurred at each node. When the system detects contaminant a at node j (i.e., $y_{aj} = 1$), there must exist at least k_a sensors in places where contaminant a travelled. The set of nodes where contaminant a travelled when it is detected at node j is defined by $\{i \in V, d_{ai} < daj\}$. This is captured in constraint $(Y5)$.

3.3. XY Formulation. The *XY* formulation utilizes concepts from both k-DSP.X and k-DSP.Y. Binary decision variable x_{aj} indicates whether a sensor placed at node j detects contaminant a or not; while y_{aj} indicates whether the system detects contaminant a at node j or not. Here is the *XY* model, k-DSP.XY,

$$\min_{x,y} \sum_{a \in A} D_a \tag{OXY}$$

$$\sum_{j \in V} s_j = p \tag{XY1}$$

$$y_{aj} \leq x_{aj} \leq s_j \quad \forall a \in A,\ j \in V \tag{XY2}$$

$$\sum_{j \in V} x_{aj} = k_a \quad \forall a \in A \tag{X3}$$

$$\sum_{j \in V} y_{aj} = 1 \quad \forall a \in A \tag{Y3}$$

$$D_a \geq d_{aj} x_{aj} \quad \forall a \in A,\ j \in V \tag{X4}$$

$$D_a = \sum_{j \in V} d_{aj} y_{aj} \quad \forall a \in A \tag{Y4}$$

$$k_a y_{aj} \leq \left(\sum_{i \in V, d_{ai} < d_{aj}} x_{ai} \right) + x_{aj} \quad \forall a \in A,\ j \in V \tag{XY5}$$

Objective (OXY) minimizes over decision variables x and y. Constraints ($XY1$), ($X3$), ($Y3$), ($X4$), and ($Y4$) are taken from either k-DSP.X or k-DSP.Y. Constraint ($XY2$) expands on ($X2$) and ($Y2$) by relating x and y. Constraint ($XY5$) is a variation of constraint ($Y5$). It replaces s_j by x_{aj}. ($XY5$) is a stronger constraint. In order for the system to detect contaminant a at node j ($y_{aj} = 1$), not only does it need at least k_a sensors ($\sum s$), it requires k_a sensor detections ($\sum x$). Furthermore, constraint ($XY5$) implies constraint ($Y5$) in k-DSP.Y.

3.4. Auxiliary Constraints. An auxiliary constraint is a constraint that is not required by the definition of a model. Frequently, auxiliary constraint adds more insight to a model. As a result, the added constraints may significantly reduce the solution space and increase the speed of a MIP solver. The following are two auxiliary constraints:

$$D_a \geq \frac{\sum_{j \in V} d_{aj} x_{aj}}{k_a} \quad \forall a \in A \tag{A1}$$

$$k_a \left(\sum_{i \in V, d_{ai} < d_{aj}} y_{ai} \right) + k_a y_{aj} \leq \left(\sum_{i \in V, d_{ai} < d_{aj}} x_{ai} \right) + x_{aj} \quad \forall a \in A,\ j \in V \tag{A2}$$

($A1$) is an extension of the ($X4$) constraint. ($X4$) is a maximum constraint. D_a is a maximum of all the $d_{aj}x_{aj}$ terms. The right hand side of ($A1$) is an average of the $d_{aj}x_{aj}$ terms. Auxiliary constraint ($A1$) states that the maximum is greater or equal to the average. Similarly, auxiliary constraint ($A2$) is an extension of constraint ($XY5$). ($A1$) is added to the X formulation and the XY formulation. ($A2$) is added to the XY formulation.

4. Methodology and Test Problems. In this section, we describe the basic methodology used to create test problem instances for k-DSP. In [1], the authors used the EPANET toolkit to simulate the effect of water contamination spread on a real system. The results from the simulation are recorded as d_{aj}, the disaster coefficients. They are the inputs of DSP. The ideal data for k-DSP is to use the same data generated by EPANET. However, due to security issues, the data is not publicly available.

The test problem instances in this paper are generated randomly, following the Euclidean distance rule. We pick random points on a two-dimensional space. The Euclidean distance between point a and another point, j, is recorded as d_{aj}. In our problem instances, we set the number of nodes to be equal to the number of contamination events. We create four problem instances, which we denote as $N25$, $N30$, $N35$, and $N40$. These networks contain exactly 25, 30, 35, and 40 nodes, respectively. In all problem instances, there are exactly 10 sensors available, $p = 10$. k_a, number of detections required for contaminant a, is taken from a uniform distribution between 2 and 4.

In another set of test problem instances, we set all k_a's equal to 1. This is exactly 1-DSP, which is studied extensively in [1]. We compare how our models behave when k-DSP is simplified to 1-DSP. The test problem instances are summarized in Table 4.1.

Our problem instances are considerably smaller in size compared to SNL-1, SNL-2, and SNL-3 in [1]. This is because we are solving a more difficult problem, where $k > 1$. We choose 25, 30, 35, and 40 as the number of nodes, because it is comparable in size with the "Anytown U.S.A." network in [8], which has 16 nodes.

5. Results. We solved all test problem instances using ILOG's AMPL/CPLEX 9.1 MIP solver, which is a state-of-the-art MIP solver. The run time and the number of nodes traversed by the MIP solver are recorded in Table 5.1 and 5.2 respectively. Each test instance is given a maximum of 1440 CPU minutes. If a instance finishes before the time limit, the run time and the number of nodes are recorded in Table 5.1 and 5.2, respectively. If a instance does not finish, it is recorded as *Timed Out* and the optimality gap is recorded in Table 5.1.

TABLE 4.1
Test problem instances definition. The first four instances are k-DSP. The last four are 1*-DSP.*

Instance Name	Number of Nodes, $\|V\|$	Number of Sensors, p	k-Detectability
N25	25	10	$U[2,4]$
N30	30	10	$U[2,4]$
N35	35	10	$U[2,4]$
N40	40	10	$U[2,4]$
N25-1	25	10	1
N30-1	30	10	1
N35-1	35	10	1
N40-1	40	10	1

TABLE 5.1
Run times for X, Y, and XY formulations. Run time is recorded in number of CPU minutes. '∗' means there is no optimal solution found after 1440 CPU minutes, an optimality gap is recorded instead. '–' indicates that the experiment has yet to finish.

Instance Name	X Formulation	Y Formulation	XY Formulation
N25	19	1	1
N30	*3.94%	16	8
N35	–	412	130
N40	–	*10.43%	*3.24%
N25-1	0	0	0
N30-1	0	0	0
N35-1	0	0	0
N40-1	0	0	0

The run time and the number of nodes are two key measures of the performance of a MIP formulation. They are used to compare the performances of different formulations. Both Y and XY solved $N25$ and $N30$ within 20 CPU minutes. X had difficulty solving $N30$. It was not able to find an optimal solution after 1440 CPU minutes. All three formulations could not solve $N40$ to optimality within the time limit. However, Y has a larger optimality gap compared to XY. In general, there exists a trend amongst X, Y, and XY formulations. In terms of run time, XY has better performance than Y; Y has better performance than X.

We compare the relationship between run-time (number of nodes) versus problem size. As size of the problem increases, the run time is increased, while the number of nodes is also increased. When the number of nodes reaches 40, the MIP solver was not able to find an optimal solution within 1440 CPU minutes. As observed from these results, we conclude that the MIP formulations do not scale well to problem instances of large size.

We look at a special case of the k-detectability sensor placement problem: when $k = 1$ (N25-1, N30-1, N35-1 and N40-1), i.e. the 1-DSP. The run-time (Table 5.1) of all 1-detectability instances are less than 1 CPU minute, significantly less when compared to their counter parts, $k > 1$. In Table 5.2, we observe that the 1-detectability problem requires zero branch-and-bound nodes. In essence, the root node is the solution. [1] reported the same observations. The difference between 1-DSP and k-DSP is in the number of branch-and-bound nodes. 1-DSP requires no branch-and-bound nodes, but k-DSP needs a significant number of branch-and-bound nodes to find the optimal solution.

TABLE 5.2

Number of nodes traversed in a MIP search tree from running X, Y, and XY formulations on the problem instances. '$*$' is recorded where no optimal solutions are found after 1440 CPU minutes, the number of nodes traversed up to the time limit is recorded. '–' indicates that the experiment has yet to finish.

Instance Name	X Formulation	Y Formulation	XY Formulation
N25	495300	400	700
N30	*29450000	26200	1900
N35	–	339600	41300
N40	–	*367000	*10900
N25-1	0	0	0
N30-1	0	0	0
N35-1	0	0	0
N40-1	0	0	0

6. Discussion. The DSP formulation found in [1] is equivalent to a p-median problem, a facility location problem described in [4]. The difference is only in the terminology. The contaminant is the customer and the sensor is the facility. The demand of the customer is satisfied by visiting the closest existing facility. The objective of the p-median problem is to minimize the total distance travelled by all customers. The structure and the solution methods of the p-median problem have been studied extensively [6]. Unfortunately, k-DSP does not have a direct equivalent in the theory of facility location. The p-center problem and the capacitated p-median problem have similar features to k-DSP, yet they are different. To the best of our knowledge, problems of the same types as k-DSP have not yet been studied in the literature. Therefore, this is a first attempt to solve a difficult problem, which is a generalization of the p-median problem.

We define the generalization of the p-median problem formally. There are exactly p facilities to be located to serve $|A|$ customers, each customer, $a \in A$, is satisfied when they visit at least k_a facilities. The objective is to minimize the total distance travelled by all customers while keeping all customers satisfied. We call this generalization the k-generalized p-median problem. When $k = 1$, it is the p-median problem.

The structures of k-DSP.X and k-DSP.Y are similar to the structure of the p-median problem. In k-DSP.Y, constraint $(Y5)$ is the additional constraint that differs from the p-median problem. However, this additional constraint causes a significant increase in computational time. The same happens to k-DSP.X when constraint $(X4)$ is added. The observation that the additional constraint resulted in higher run times leads us to the implementation of Lagrangian relaxation methods. In [3], a Lagrangian based local search heuristic is applied to the capacitated p-median problem because of the additional capacity constraint. We developed a similar approach based on a Lagrangian relaxation method for k-DSP.X. The result was disappointing. The largest lower bound found using the Lagrangian relaxation was the same as the Linear relaxation lower bound.

The combination of X and Y formulations gave an unexpected result. By combining the decision variables, x and y, the solution space essentially doubled. Intuitively, XY should perform worse than both X and Y individually because of the difference in the size of the solution space. In fact, XY spends more time on search at each branch-and-bound node (Readers can verify that by simple division). However, the interplay between the x and the y variables in the XY model reduces the overall run time in some problem instances. This is because the number of nodes traversed by XY is usually the lowest amongst all three formulations. This is more evident when the problem size is large.

The contribution of this paper is two-fold. First, the proposed MIP formulations are a first attempt to solve the k-detectability sensor placement problem. Second, the relationship between DSP and p-median creates a new class of problems of k-generalized p-median problem. DSP corresponds to *the* p-median problem; k-DSP corresponds to the k-generalized p-median problem.

For future experiments, we would like to test the proposed formulations on real-world data, such as SNL-1, SNL-2, and SNL-3. Currently, the tests are only performed on random data. Although, the size of the real-world data is much larger, we want to observe if the structure of the real-world data can be exploited to our advantage.

7. Conclusions. We proposed three Mixed Integer Programming formulations, X, Y, and XY formulations. They model the k-detectability sensor placement problem. X and Y were formulated naturally based on the problem definition. XY combined both X and Y. The combined formulation showed marked improvements in performance in terms of run time. However, none of the formulations fared well when the problem size was large. Furthermore, our approaches to k-DSP was a first attempt to solve the class of problems we called the k-generalized p-median problem.

REFERENCES

[1] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber, and J.-P. Watson, *Sensor placement in municipal water networks with temporal integer programming models*, Journal of Water Resources Planning and Management, 132 (2006), pp. 218–224.

[2] B. Drage, J. Upton, and M. Purvis, *On-line monitoring of micropollutants in the river trent (uk) with respect of drinking water abstraction*, Water Science and Technology, 38 (1998), pp. 123–130.

[3] L. A. Lorena and E. L. Senne, *Local search heuristics for capacitated p-median problems*, Networks and Spatial Economics, 3 (2003), pp. 407–419.

[4] P. Mirchandani and R. Francis, *Discrete Location Theory*, John Wiley and Sons, 1990.

[5] A. Ostfeld and A. Kessler, *Protecting urban water distribution systems against accidental hazards intrusions*, in IWA Second Conference, 2001.

[6] M. Resende and R. Werneck, *A hybrid heuristic for the p-median problem*, Journal of Heuristics, 10 (2004), pp. 59–88.

[7] L. Rossman, *The epanet programmer's toolkit for analysis of water distribution systems*, in Annual Water Resources Planning and Management Conference, 1999.

[8] T. Walski, J. E. Downey Brill, J. Gessler, I. C. Goulter, R. M. Jeppson, K. Lansey, H.-L. Lee, J. C. Liebman, L. Mays, D. R. Morgan, and L. Ornsbee, *Battle of the network models: Epilogue*, Journal of Water Resources Planning and Management, 113 (1987), pp. 191–203.

SEMISUPERVISED NAMED ENTITY RECOGNITION

TAYLOR P. TURPEN* AND DANIEL M. DUNLAVY†

1. Introduction. With the ever increasing number of documents readily available to the casual reader, careful observer or scientific analyst it is becoming more important to quickly recognize names of people places, locations, etc., within those documents. Although it may be a relatively easy task to extract *named entities* through manual inspection, parsing the million of blogs created daily[1] is untractable with such an approach.

In this paper we focus on statistical methods for solving the Named Entity Recognition (NER) problem [4], i.e., the identification of words and phrases that are associated with a given set of entity categories. Specifically, we will focus on sequence-based machine learning methods, with an emphasis on semisupervised learning [1, 8], an iterative approach to predictive modeling that uses both labeled and unlabeled data in generating models.

The definition of the NER problem used in the paper is as follows. Given a sequence of words, $\tilde{\mathbf{x}} = \{\tilde{x}_1, \ldots, \tilde{x}_n\}$, and associated named entity labels $\tilde{\mathbf{y}} = \{\tilde{y}_1, \ldots, \tilde{y}_n\}$, the goal is to generate a model for predicting the labels of new word sequences. Spefically, we seek to find a model, f, such that

$$\hat{y} = f(\tilde{x}) \,, \tag{1.1}$$

"best agrees" with $\tilde{\mathbf{y}}$. Throughout the remainder of this paper, we will refer to $\tilde{\mathbf{x}}$ as the *labeled data*, $\tilde{\mathbf{y}}$ as the *true labels*, and $\hat{\mathbf{y}}$ as the *predicted labels*.

A set of features associated with a word is used in determining the most probable label for that word. Examples of features explored in this paper include the position of the sequence (e.g., sentence beginning, end, or other) and capitalization; the statistics on the use of such features as they relate to the labeled data and true labels are used to set parameters in the NER model, f.

Within the class of sequence-based NER are also rule-based methods, i.e., sets of prioritized logical conjunctions that are tested for each sequence being labeled. One benefit of statistical methods, over rule-based approaches, is that all that is required is a sequence of labeled data and the associated true labels; there is no need to know grammatical constructs specific to the particular subject domain or genre in which the text is written in order to generate models. However, there is evidence that incorporation of such information into statistical models may be useful and we plan to explore which method may work best for a particular data set or type of data in future work.

A major challenge in solving the NER problem is handling word sense ambiguity, the fact that words often have more than one meaning and thus entity type. For example, the word "Washington" may be of type *person* (Geroge Washington), *place* (Washington, D.C. or the state of Washington), or *organization* (Washington University). One benefit in using statistical-based methods is that most models compute likelihoods or probabilities of all entity types specified for each word. For example, consider the following sentence.

`John attends George Washington University with George Washington.`

Typical computation performed by a statistical model may be as follows for this example (assuming only four possible entity types).

*San Diego University, tturpen-10@sandiego.edu
†Sandia National Laboratories, dmdunla@sandia.gov
[1] http://technorati.com/blogging/state-of-the-blogosphere/

TABLE 1.1
Different ways of combining these probabilities lead to different algorithms.

	Probability			
Word	*Person*	*Place*	*Organization*	*Other*
John	0.95	0.01	0.01	0.03
attend	0.01	0.01	0.01	0.97
George	0.07	0.04	0.85	0.04
Washington	0.07	0.04	0.85	0.04
University	0.01	0.10	0.75	0.05
with	0.01	0.01	0.01	0.97
George	0.75	0.21	0.01	0.03
Washington	0.79	0.19	0.01	0.01

Machine learning methods build predictive models using statistical inferences derived from training data (e.g., the labeled data described above). There are three main machine learning approaches: unsupervised, supervised and semisupervised. Unsupervised techniques are given no structural or implicit information about the data they are processing. For the NER problem, unsupervised methods (see, e.g., [3]) do not use any information about the entity labels associated with the training data. Supervised learning uses both the labeled data and the true labels to build predictive models.

A sequence-based semisupervised NER approach trains a model on an initial set of labeled data and true labels, makes predictions on a separate set of unlabeled data, and then iteratively attempts to create improved models using predictions of previously generated models (plus the original labeled data and true labels). The main advantage of semisupervised approaches is that it is possible to create more accurate NER models from less training data. Since annotating training data is a task which requires time and money resources, and the data may be related to a niche technical area, proprietary, and/or classified, requiring subject matter experts or limited personnel, this benefit of semisupervised learning can become crucial.

The Stanford Natural Language Processing Group has created software for solving the NER problem [2], known as the Stanford Named Entity Recognizer (SNER). SNER uses a supervised learning approach called conditional random fields [3, 6], where a model maximizing the conditional probabilities of the sequence of true labels, $\tilde{\mathbf{y}}$, given the labeled data, $\tilde{\mathbf{x}}$, is generated. Our work has focused on creating a semisupervised approach by wrapping an outer loop of iteration around SNER, where at each iteration the training data used by SNER changes. Our wrapper implementation is called the Semisupervised Utility for Named Entity Recognition (SUNER) and is discussed in more detail in Section 3. SUNER accomplishes two tasks: 1) performance of NER modeling for different types highly depends on the various features used in creating the NER models and 2) performance using semisupervised learning can be comparable to that of supervised learning using only a fraction of the size of training data used by supervised learning.

2. Named Entity Recognition. Named Entity Recognition is a difficult task. Polysemy and synonomy don't make it any easier. Polysemy is defined as "the coexistence of many possible meanings for a word or phrase."[5]. Whereas, synonomy is "the state of being synonomous. . . having the character of a synonym. . . one of two or more words or expressions of the same language that have the same or nearly the same meaning in some or all senses."[7] And that's just the beginning. How should NER deal with slang terms? Do words used in professional applications have the same meaning as those used in casual emails or blogs?

NER can also be used to identify proteins, acids, gene sequences etc.

2.1. Statistical Inference. As we discussed earlier, there are two main approaches to sequence based methods: Rules and Statistics. We also mentioned something known as a machine learning feature set, which describes words in numerical or boolean values. A NER semisupervised algorithm trains itself by building a statistical model based on what it knows to be significant. This is broken into two stages: initial training and iterative training. Initial and iterative training constitue the entirety of semisupervised learning to be discussed in section 3. Initial training, which we will touch on briefly here is how the program remembers which words correspond to which names by recording which features were true for those words. In essence it builds a percentage confidence associated with different features being true. For example the sentence:

```
I saw Washington<person> walking down Central<street>
with Boogie<monster>.
```

Evaluate the word "Washington". After reading the sentence, it can be assumed that a word that does not begin the sentence nor end the sentence and has a capital letter is a person. The same goes for street. However, as far as statistical modeling is concerned, any word that has a capital letter and ends the sentence is a monster. When this concept is extrapolated to thousands of documents it becomes more easy to generalize each feature set and the values associated with each entity label.

2.2. Applications. While we have tested SUNER on the University of Pennsylvania Biomedical Database and Newswire Documents there are many other applications and ways to think about named entities such as:

- Variable cross-language labels can be useful because their evaluation is consistant regardless of data subject matter.
  - PERS, ORG, LOC, etc.
- Static part of speech tagging is useful for many translation applications as well as being used in conjunction with the general cross-language labels.
  - Noun, Verb, Predicate, Adjective, Adverb etc.
- Variable improvisational tags are flexible and imaginative where their use comes from the importance of their application.
  - Terrorist, first baseman, gene-protein, car salesman, radioactive isotope etc.

3. Semisupervised Learning. Sequence based semisupervised learning is a machine learning technique that reads through an already annotated training document word by word. It builds the statistical inferences from that training data, initial training, and then uses them to make predictions on an unannotated document. The highest of those predictions are then assumed to be part of the annotated corpus and are used to train a new statistical model, also known as iterative training. Usually semisupervised learning will continue to iteratively train statistical models until some stopping criteria is met. This is known as self-training because in a way the model is training itself without absolute certainty as to which labels are true, only an educated guess as to which labels are statistically likely.

3.1. Initial training. Initial training can be viewed as describe in equation 1.1. Where $\hat{y}$ is the sequence of predicted labels created after the training process f on the training data $\tilde{x}$. Take this one step further and include what is known as a weighting scheme. Because it may be more important that a word includes a capital letter than it is that the word is at the beginning of the sentence, the function becomes:

$$\hat{y} = f(\tilde{x}, w_d) \,, \tag{3.1}$$

Where w_d is the default weighting scheme. SNER uses a Quasi-Newtonian Minimizing scheme in order to achieve the most ideal weight for each feature depending on $\tilde{x}$.

3.2. Iterative Training. Iterative training takes place on unannotated data. Basically, semisupervision takes the $\hat{y}$ achieved after initial training. If the prediction values, or percentage confidences of $\hat{y}$, are above a certain threshold, say 95%, then the program will train a new model considering those names to be true names. That gives us:

$$\hat{y}_0 = f(\tilde{x}, w_0, \hat{x}, t_d) \,, \tag{3.2}$$

Where $\hat{y}_0$ is the sequence of new predictions from $\tilde{x}$ original training data, w_0 weighting scheme as minimized for that data, and $\hat{x}$ the set of words and names from $\hat{y}_0$ with predictions higher than that of, t_d the prediction threshold. Note that once $\hat{x}$ is constructed via statistical predictions it does not become part of $\tilde{x}$. A prediction that falls above the acceptable threshold is only considered true for that iteration of iterative training. Predictions that persist through training iterations are classified under indelibility and will not be discussed in this paper.

3.3. Parameters. In order to control the iterative training portion of semisupervised learning there are several parameters that need to be evaluated.

3.3.1. Stopping Criteria.

- Performance threshold: Once the statistical model begins naming words with names that are incorrect, the model will stop iterative training.
- Numeric threshold: The model will stop iterative training once it has repeated a certain number of times. The performance metrics created as a result of numeric threshold implementation can then be used to maximize or minimize the number of iterations.
- Improvement threshold: Once the predicted names of words are the same predictions as the last iteration, because of a lack of idelibility, and no new predictions are being made, stop iterative training. This highlites a model's inability to improve its predictive power despite the lowering of t_d across iterations.
- Flexible Improvement threshold: This threshold is the same as the Improvement Threshold except that no new predictions are being made within an acceptable margin
- Floor threshold: Once the t_d drops below a certain percentage when using stepping(sec 3.3.2), the model will stop iterative training.
- t_d: This is not actually a stopping criteria but instead the threshold below which no predictions are accepted for potential word names

Often several of these stopping criteria will be used in conjunction with one another. When one of the parameters is exceeded the iterative training will stop regardless of the status of the others. Of course, some sort of marginal slush-factor can be used if the other parameters are nowhere close to being exceeded.

3.3.2. Stepping. Often to improve the quantity of accepted predictions, with little sacrifice to their quality, a stepping threshold is used. It is structured so that each iteration of iterative training results in a lower t_d. For example if the step size is 1% the first t_d may be 99%, then after the first iteration it will be 98% then 97% and so on.

Depending on the strength of the model being used the step size should be adjusted in order to reflect the model's ability to improve predictions. i.e. A very "strong" model would reflect poorly upon a large stepping size (greater than 1%). This is because a strong model will have little statistical variance (change in prediction values across $\hat{x}$) in comparable predictions. However, a weak model might perform well with one because the statistical

variance in predictions is usually larger with weaker models. The variability itself is part of the nature of f, and the size of $\tilde{x}$ when compared to $\hat{x}$. Stepping is also best used in conjunction with the improvement threshold. In that way, t_d will not be able to drop below acceptable levels if no literal predictive improvement is being made.

3.3.3. Named Entity Balancing. The goal of balancing is to moderate the preference of the statistical model. If the training data contains an unequal distribution of named entities (annotated words) then balancing is any sort of numeric "leash" put on the maximum number of accepted predicted named entities for any one set of names.

This helps to avoid what is known as "oversampling" or "undersampling" and prevents model preference toward a certain name that may not be a true representation of the global type of data being processed. Imagine wanting to create a model for scientific documents. If the model was trained on 500 newswire documents in the field of bioengineering, that model would more easily name words like "acid" or "rna" than it would "torque" or "pounds", even though the latter might be mentioned in the training set, they are not prevalent enough to make a statistical impression.

In SUNER we implement a simple linear balancing algorithm. The results for which we are still working on and will be added shortly.

4. Numerical Experiments. First, it will be shown that with all else held constant, semisupervised named entity recognition depends largely upon which features are being used. By providing evidence for the previous point it will also be shown that some features result in higher performance results than others. In order to demonstrate this a simple SUNER statistical model

using no balancing was implemented using k-fold cross validation. This data analysis technique involves dividing up a data set into k sections. Each section will be swapped out between being considered annotated(part of $\tilde{x}$), unannotated or "left-out" by the semisupervised program. When a document partition is considered unannotated, the program trains itself on the $k-2$ remaining documents considered to be annotated. It is important to train f on different annotated data partitions and then iteratively train it on other data partitions in order to avoid any sort of bias present within the data itself. Each partition, in turn, will also be left out of both training and iterative training in order to remove its bias on the model entirely.

The self-training also includes a number of "key features" which alternate on and off while all other features stay on. When a feature is "on" it will be read as true by the program if it applies to the word in question. If the performance evaluation metrics vary when the features are alternated, then the assumption that features affect the performance of semisupervised named entity recognition is true.

Second, it will be shown that with all else held constant, semisupervised named entity recognition can produce the same or better results than supervised named entity recognition with only a fraction of the size of training data. This will be done using leave-one-out k-fold cross validation and a balancing implemented self-training algorithm.

In order to minimize under or over sampling, a somewhat normal but randomized distribution algorithm was used to parse the data sets. Each trial includes the corresponding name/fold distribution table.

4.1. Results From 10-fold Cross Validation Self-Training With Key Features: 0,5,9. The first test consisted of a ten-fold cross validation on 261 Medline documents. When parsed into the Stanford Named Entity Recognition format, the document folds had the name distribution indicated by 4.1. The names, indixed by $L_0 ... L_5$ are as follows: Null-label(no name, or "other"), gene-protein, malignancy-histology, gene-rna, malignancy-sites,

malignancy-clinical-stages.

TABLE 4.1
10-fold Cross Validation Name Distribution

Fold	L_0	L_1	L_2	L_3	L_4	L_5
0	566	147	128	101	38	16
1	140	51	52	41	17	7
2	181	68	65	43	21	14
3	238	89	95	63	26	14
4	296	112	111	79	38	18
5	341	133	114	84	42	23
6	382	162	137	105	42	19
7	437	183	153	117	62	23
8	472	206	183	130	68	27
9	516	235	195	151	71	37

TABLE 4.2
10-fold Cross Validation, Fold 8, With Key Features: 0,5,9. Testing(and iterative training) File Length(lines): 4584 Training File Length:27794(lines), Number of Labels(6)

Fold	F-Measure	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
8	0.60545	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1
8	0.60401	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1
8	0.59907	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
8	0.59981	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
8	0.60545	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
8	0.60401	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
8	0.59907	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0.59981	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The F-Measures for one of the folds is in table (4.1,4.2). A 0 indicates that that particular feature was not used for that iteration. A 1 indicates the opposite. The F-measure is the weighted harmonic mean of both precision and recall values. Where TP(true positives) the presence of a named entity was correctly predicted, TN(true negatives) the absence of a named entity was correctly predictued, FP(false positives) the presence of a named entity was incorrectly predicted and FN(false negatives) the presence of a named entity was not predicted when it should have been, are elements of performance evaluation in the following equations:

$$\text{precision} = \frac{TP}{TP + FN}\,, \tag{4.1}$$

$$\text{recall} = \frac{TP}{TP + FP}\,, \tag{4.2}$$

$$\text{F-Measure} = \frac{\text{Precision} \cdot \text{Recall}}{(1-\alpha)P + \alpha\text{Recall}}\,, \tag{4.3}$$

Where α is the weighting measure which usually equals 0.5 weighting *recall* twice as much as *precision*.

REFERENCES

[1] O. Chapelle, B. Sch"olkopf, and A. Zien, eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[2] T. G. Jenny Rose Finkel and C. Manning, *Incorporating non-local information into information extraction systems by gibbs sampling*, in Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics, 2005, pp. 363–370.

[3] J. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, in MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, 2001, pp. 282–289.

[4] D. Nadeau and S. Sekine, *A survey of named entity recognition and classification*. 2006.

[5] *Oxford English Dictionary*, Oxford University Press, 2009.

[6] H. Wallach, *Conditional random fields: An introduction*, Rapport technique MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 50 (2004).

[7] *Webster's Ninth New Collegiate Dictionary*, Merriam-Webster, Spingfield, MA 1984.

[8] X. Zhu, *Semi-supervised learning literature survey*, Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

A STUDY OF DIVERSITY IN ENSEMBLE MODELS FOR CLASSIFICATION PROBLEMS

SEAN A. GILPIN[†] AND DANIEL M. DUNLAVY[‡]

Abstract. The relationship between ensemble classifier performance and the diversity of the predictions made by ensemble base classifiers, is of importance because it may help explain why some ensemble methods perform better than others. We explored how using ensembles with pairs of base classifier models (i.e. decision trees paired with SVM) can effect the diversity of the ensemble. We found that heterogeneous ensemble models are capable of increasing the accuracy of predictions as well as increasing the diversity of the ensembles.

1. Introduction. The problem of data classification, or data labeling, arises in a wide variety of applications. Examples include detecting spam e-mail messages based on the content of the messages (document classification), labeling cells and tumors as malignant or benign based on the context of MRI scan data (image classification), and identification of individuals based on fingerprints, facial features, and iris patterns (biometric identification). In all of these examples, the goal is to predict a discrete label (e.g., "spam" versus "not spam") for a particular data instance (e.g., a particular e-mail message) based on the attributes of that instance.

More formally, classification is the task of learning a function, f, that maps a set of data instance attributes, $\mathbf{x} = \langle a_1(\mathbf{x}), \ldots, a_m(\mathbf{x})\rangle$, to one of several predefined class labels from $\mathcal{Y} = \{c_1, \ldots, c_k\}$. The function f is often called a classifier or classifier model, but in this paper we will use the term classifier to designate classification functions and the term classifier model will only be used to describe the structure of such functions. The set of data instances used to learn, or train, a classifier is called the training set and is denoted $\mathcal{D}_{tr} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where n is the number of instances, $\mathbf{x}_i \in \mathbb{R}^m$ is a vector of attributes, or features, for data instance i, and $y_i \in \mathcal{Y}$ is the label for data instance i. In order to validate the models learned, it is common practice to select some of the training data to be used in testing the resulting classifier models. This testing, or validation data, is denoted $\mathcal{D}_{te}$ and is not used in training the classifier model.

Recent results in solving classification problems indicate that the use of ensembles, or sets of classifier models, often leads to improved performance over using single classifier models [1, 2, 3, 12]. Much of the previous work on ensembles of classifier models (see e.g., [5]) has focused on *homogeneous ensemble classifiers*—i.e., collections of classifier models of a single type. In this work, we focus on *heterogeneous ensemble classifiers*, where the collection of classifiers are not of the same type. Note that such classifier models are also referred to as hybrid ensemble classifiers. Our goal is to find when and how the use of heterogeneous ensembles can be advantageous.

Our work with heterogeneous ensembles focuses on using diversity measurements as a tool to help explain when and how heterogeneous ensembles can outperform homogeneous ensembles. Specifically we looked to find the relationship between diversity and accuracy. We wanted to find if heterogeneous performance boosts corresponded to increases in diversity measurements. As part of our previous work, we created a software framework called HEMLOCK (Heterogeneous Ensemble Machine Learning Open Classification Kit) for creating and evaluating heterogeneous ensemble classifiers. We extended HEMLOCK to allow evaluation of diversity measurements, and used it to produce all of the findings for the ensemble models we experimented with for our current work.

[†]Computer Science, University of California at Davis, sagilpin@ucdavis.edu
[‡]Computer Science and Informatics, Sandia National Laboratories, dmdunla@sandia.gov

2. Diversity. The success of ensemble classification models over non-ensemble models is partially dependent on the diversity of the predictions made by its base classifiers. To see this, consider the case when all of the base classifiers make the same predictions. In that case the ensemble would perform no better than any of the base classifiers taken individually and there would be no benefit to using an ensemble. In order to objectively study the relationship between diversity and ensemble performance, we first need to define some objective measurements capable of measuring the diversity of predictions made by a set of classifiers. Many studies of this relationship already do exist and below we detail the diversity measurements that have been used in most of those studies.

There are two types of diversity measurements that we have listed below: pairwise and non-pairwise. Pairwise measurements are designed to compare the differences in predictions of two classifiers. Their interpretation in that setting is clear, but once averaged over all possible pairs in a base classifier set, the interpretation may become less clear. In this paper we only explore the mean of pairwise diversity measurements but it may be useful to study other statistics such as the standard deviation as well, to get a better understanding of what these measurements are trying to tell us. Non-pairwise diversity measurements are designed to measure differences in predictions of sets of more than two classifiers. Although their definitions are typically more complex than the pairwise diversity measurements, their interpretations are not muddled by the details of working with ensembles of size greater than two.

2.1. Pairwise Diversity Measurements. The following diversity measure are used for comparing the predictions of two classifiers, f and g. In a set of base classifiers $\mathcal{B}$ of size L, the i^{th} base classifiers is referred to as f_i. The oracle function is a binary function whose domain consists of pairs of classifiers and instances. The value of the oracle function is one when a classifier correctly predicts the true label of the instance it is paired with, and the value is zero when the prediction is incorrect. In practice we can only evaluate this function for instances whose true label we already know, such as those found in the training set. The oracle function can be defined specific to classifier f such that its domain is the set of instances in the training set. In that case the following is the definition of the oracle function.

$$O_f(\mathbf{x}_j) = \begin{cases} 1 & : f(\mathbf{x}_j) = y_j \\ 0 & : otherwise \end{cases}$$

Using this definition, we can then define the variables a,b,c,d which characterize the proportions of correct or incorrect predictions made by two base classifiers f and g. These definitions, found in Table 2.1 will be useful in defining many of the pairwise diversity measurements.

Any pairwise diversity measurement pd, can be used as a measurement of total ensemble diversity by calculating the average $pd_{i,j}$, the diversity measurement comparing f_i and f_j, over every pair of base classifiers:

$$\frac{2\left(\sum_{i=1}^{|\mathcal{B}-1|}\sum_{j=i+1}^{|\mathcal{B}|} pd_{i,j}\right)}{(n)(n-1)}$$

The following are definitions for the pairwise diversity measurement that we explored. All of these definitions are listed in Kuncheva's book[10], and original references are provided as well.

Correlation. [10] Correlation measures the linear dependence between the predictions of two base classifiers. Specifically, it measures the dependence of the oracle values for a

TABLE 2.1
Statistics corresponding to oracle values of a pair of classifiers. These are defined here to simplify the formulations of many of the pairwise diversity measurements.

$a = \frac{1}{n}\sum_{i=1}^{n} O_f(\mathbf{x}_i)O_g(\mathbf{x}_i)$	Proportion of instances correctly predicted by f and g
$b = \frac{1}{n}\sum_{i=1}^{n} O_f(\mathbf{x}_i)(1 - O_g(\mathbf{x}_i))$	Proportion of instances predicted correctly by f and incorrectly by g
$c = \frac{1}{n}\sum_{i=1}^{n}(1 - O_f(\mathbf{x}_i))O_g(\mathbf{x}_i)$	Proportion of instances predicted incorrectly by f and correctly by g
$d = \frac{1}{n}\sum_{i=1}^{n}(1 - O_f(\mathbf{x}_i))(1 - O_g(\mathbf{x}_i))$	Proportion of instances incorrectly predicted by f and g

pair of base classifiers. The range of this measurement is between -1 and 1 where values close to zero imply that there is no correlation between between the oracle outputs. For a two class classification problem, a correlation value of -1 implies that the predictions made by the classifiers are opposite of each other. For a multiclass classification problem with greater than two classes, a correlation of -1 only implies that neither of the classifiers were both right or both wrong for any of the test instances.

$$\rho = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}$$

The Q Statistic. [14] Yule's Q statistic has similar interpretation as correlation. The range of this measurement is between -1 and 1.

$$Q = \frac{ad - bc}{ad + bc}$$

Disagreement. [11] Disagreement between a pair of classifiers is the proportion of instances for which they predict different class labels. The range of this measurement is between 0 (they never disagree), and 1 (they disagree on every instance in the test set).

$$D = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{\mathbf{z}|f(\mathbf{z})=g(\mathbf{z})\}}(\mathbf{x}_i)$$

Where $\mathbf{1}_{\{\mathbf{z}|f(\mathbf{z})=g(\mathbf{z})\}}(\mathbf{x}_i)$ is the indicator function whose value is 1 for instances where the predicted target labels are the same for both classifiers.

Double Fault Measurement. [7] The double fault measurement between a pair of classifiers is the proportion of instances for which they both predict the wrong class. This corresponds to the value d found in Table 2.1. The value of this measurement is one when both of the classifiers are always wrong and zero when the classifiers are never simultaneously wrong about the same instance. A low double fault measurement is desired for an ensemble, because otherwise many of the base classifiers will be making mispredictions for the same instances which will increase the chance of that instance being misclassified by the ensemble.

$$DF = d$$

2.2. Non-Pairwise Diversity.

Entropy. [6] This formulation of entropy measures how close to a tie the base classifier set comes to, when voting for the prediction of an instance. If the base classifiers come to an exact tie for each instance, then the entropy will be one, and if the base classifiers unanimously give the same class label predictions for each instance, then the entropy will be zero.

$$E = \frac{1}{N}\frac{2}{|\mathcal{B}|-1}\sum_{i=1}^{n} min\left\{\left(\sum_{j=1}^{|\mathcal{B}|} O_{f_j}(\mathbf{x}_i)\right), \left(|\mathcal{B}| - \sum_{j=1}^{|\mathcal{B}|} O_{f_j}(\mathbf{x}_i)\right)\right\}$$

Measure of Difficulty. [8] Define a random variable $\mathbf{X}$ as the proportion of base classifiers that will correctly classify a randomly chosen instance. Then $\mathbf{X}$ is a discrete random variable with possible values between $(0/L, 1/L, ..., L/L)$. The measure of difficulty is defined as the sample variance of $\mathbf{X}$.

General Diversity. [9] Let p_i denote the probability that exactly i of the L base classifiers predict the wrong class label for a randomly chosen instance. Let $p(i)$ be the probability that i randomly chosen base classifiers will all incorrectly predict the class label for a randomly chosen instance.

Then $p(1)$ and $p(2)$ can be defined in terms of p_i as follows:

$$p(1) = \sum_{i=1}^{L} \frac{i}{L} p_i$$

$$p(2) = \sum_{i=1}^{L} \frac{i}{L}\frac{(i-1)}{(L-1)} p_i$$

General diversity is defined below using $p(1)$ and $p(2)$. The value of this measure is one when the probability that two randomly chosen classifiers both misclassify a random instance is equal to the probability that one randomly chosen classifier misclassifies a random instance. A low general diversity measurement is desirable because it implies that misclassifications by base classifiers are more likely to be unique.

$$GD = 1 - \frac{p(2)}{p(1)}$$

Coincident Failure Diversity. [9] Coincident failure diversity is a modification of general diversity whose value is highest (one), when misclassifications are unique to one base classifier and lowest (zero) when base classifiers always make the same class label predictions. The following definition uses the same value for p_i as was defined for general diversity.

$$CFD = \begin{cases} 0 & : p_0 = 1 \\ \frac{1}{1-p_0}\sum_{i=1}^{L}\frac{L-i}{L-1} p_i & : p_0 < 1 \end{cases}$$

3. Numerical Results. In this section, we present numerical results illustrating the relationship between diversity and accuracy in ensemble classifiers as well as comparisons of the performance of heterogeneous versus homogeneous ensembles.

Each of the ensemble classifiers created consisted of 100 base classifiers which were trained using bagging [4]. The use of bagging ensured that each base classifier was trained with a subsample of the training set, and is a common method for increasing the diversity of

ensemble classifiers. Homogeneous ensembles consisted of base classifiers of support vector machines (SVMs), random trees [5], and naive Bayes classifiers as base classifier models. The base classifiers are implemented in WEKA [13] using the default model parameters in each case. The heterogeneous ensembles consisted of combinations of the three classifiers types in pairs and in varying proportions. Each ensemble model was used in conjunction with 5-fold stratified cross validation to create a total of 5 ensemble classifiers. Using 5-fold cross validation allowed us to evaluate the diversity and accuracy measurements using a test set not used in the training of the classifiers. We used 5-fold cross validation rather than 10-fold cross validation, because it lead to larger testing sets which we believe may be important when measuring diversity measurements. In order to make up for the smaller number of training instances used in 5-fold cross validation, we repeated every experiment 10 times using different random seeds for either the fold creation or bagging or both (see the specific experiments later in this section for more details). Each heterogeneous ensemble was created using a percentage of base classifiers from each of two homogeneous ensemble classifiers. The proportion of base classifiers (in 1% increments) led to different heterogeneous ensembles. This lead to a total of 297 different ensemble models for each experiment, 3 of which were homogeneous and 294 of which were heterogeneous.

Eight data sets representing a wide range of classification problems were used in the experiments. Table 3.1 presents the characteristics of these data sets; note the varying numbers of instances, classes, and features across the set. These data sets are a subset of the data used in previous work on analyzing performance of ensemble classifier models [1]. The performance measure computed for all experiments was *accuracy* (proportion of instances with correctly predicted labels) as it easily generalizes to multi-class classification problems. Majority voting was used as the fusion method for combining the base classifiers to create the ensemble classifiers, as all of the diversity measures used in this work output class labels only (versus class rankings or conditional class probability distributions). Although we computed all diversity measures presented in this paper in each of the experiments, results are presented for only three of the diversity measures: *CFD*, *Disagreement*, and *Double Fault*. These measures are representatives from the three different groups of equivalent diversity measures presented by Bian and Wang [3].

TABLE 3.1
Characteristics of the experimental data sets.

#	*Name*	*Instances*	*Classes*	*Continuous Attributes*	*Nominal Attributes*
1	abalone	4177	29	7	1
2	bupa	345	2	6	0
3	dna	3186	3	0	180
4	glass	214	6	9	0
5	ion	351	2	34	0
6	promoters	106	2	0	57
7	sonar	208	2	60	0
8	yeast	1484	10	8	0

Table 3.2 shows the accuracy of ensemble models averaged over a total of 50 ensemble classifiers (corresponding to the 10 independent runs of 5-fold cross validation). Accuracy measures for the homogeneous ensembles of SVM, naive Bayes, and random tree base classifiers and the best heterogeneous ensemble are presented. Note that across all data sets, at least one of the heterogeneous ensembles outperforms all homogeneous ensembles. Table 3.3 presents results of the average CFD measurements corresponding to the accuracy results

in Table 3.2. In 6 out of the 8 data sets, a heterogeneous model generated on average more diversity (as measured with CFD) than any of the homogeneous models. Table 3.4 shows the average disagreement measurements for the different ensemble models. The random tree ensembles dominate the other homogeneous models but the heterogeneous models still do better for 5 of the 8 data sets. Table 3.5 differs from the previous tables, because the double fault measurement corresponds to more diverse ensembles when the value is low. So for the heterogeneous models we found the model that had the minimum average double fault measurement.

The results do not show a clear relationship between diversity and accuracy. They do suggest that the most successful heterogeneous ensembles for a particular data set generally involve a large proportion of random tree base classifiers. To explain the success of the random trees, it is helpful to note that while the disagreement measures are usually high (adding to the diversity) the double fault measurement is usually relatively low. So the diversity that is created by random trees, is created in a way that doesn't reduce total ensemble accuracy.

The plots in Figures 3.1- 3.3 were chosen to demonstrate some of the trends and interesting results found in the plots of the experiment results. These plots are taken from three different data sets and each of them is a plot of a different heterogeneous ensemble pairing. Both of the heterogeneous ensembles that involve random trees show that the best performance is usually achieved through including a larger number of random tree base classifiers than SVM or naive Bayes base classifiers. The plots also show that there tends to be a wide range in the diversity values and the plots of the diversity measures are usually arching towards more diversity. The plot in Figure 3.2 is representative of heterogeneous ensembles that include naive Bayes and SVM base classifiers, in that they often show large improvement, in terms of accuracy, over what SVM or naive Bayes homogeneous ensembles demonstrate. However the pointed shape of the accuracy curve is unusual and demonstrates that the performance of the heterogeneous ensemble can be very sensitive to the exact composition of its base classifiers.

In the case of accuracy, disagreement diversity, and CFD our results show that these measurements exhibit an upward arcing trend while the double fault measurement often shows a downward arcing trend. Each of the heterogeneous ensemble experiments showed how two homogoneous ensembles of different types would perform when mixed together with different levels of composition. The question we had when we saw the arcs in the heterogeneous ensemble plots was: do the arcs represent extra performance or extra diversity that is caused by combining different types of base classifiers into one ensemble, or are the arcs just a byproduct of combining base classifiers from ensembles with large differences in per-

TABLE 3.2
Average accuracy over ten experiments repeated with different random seeds using 5-fold cross validation and bagging. At least one heterogeneous ensemble outperformed all of the homogeneous ensembles across the data sets.

Average Accuracy				
Dataset	*Naive Bayes Ensemble*	*SVM Ensemble*	*Random Tree Ensemble*	*Best Heterogeneous Ensemble*
abalone	0.2371	0.2532	0.2419	0.2555
bupa	0.5594	0.5809	0.7272	0.7333
dna	0.9396	0.9452	0.9501	0.9582
glass	0.5039	0.5820	0.7741	0.7791
ion	0.8253	0.8824	0.9367	0.9398
promoters	0.8874	0.9223	0.8994	0.9324
sonar	0.6899	0.8289	0.82886	0.82889
yeast	0.5792	0.5703	0.6080	0.6193

TABLE 3.3
Average coincident failure over ten experiments repeated with different random seeds using five fold cross validation and bagging. A heterogeneous ensemble on average outperformed all of the homogeneous ensembles.

Average Coincident Failure Diversity				
Dataset	*Naive Bayes Ensemble*	*SVM Ensemble*	*Random Tree Ensemble*	*Best Heterogeneous Ensemble*
abalone	0.1745	0.2238	0.1971	0.2475
bupa	0.5375	0.4897	0.6256	0.6253
dna	0.4748	0.8225	0.7337	0.9125
glass	0.4367	0.4849	0.6328	0.6324
ion	0.3880	0.6193	0.8546	0.8583
promoters	0.7709	0.8004	0.6433	0.8665
sonar	0.5125	0.6588	0.6921	0.7417
yeast	0.4015	0.3081	0.4917	0.5425

TABLE 3.4
Average disagreement diversity over ten experiments repeated with different random seeds using five fold cross validation and bagging.

Average Disagreement Diversity				
Dataset	*Naive Bayes Ensemble*	*SVM Ensemble*	*Random Tree Ensemble*	*Best Heterogeneous Ensemble*
abalone	0.2346	0.2677	0.7589	0.7595
bupa	0.2647	0.1328	0.3873	0.4056
dna	0.0213	0.0828	0.3797	0.3777
glass	0.3134	0.3239	0.3930	0.4463
ion	0.0558	0.0661	0.1558	0.1656
promoters	0.1376	0.1230	0.4437	0.4423
sonar	0.1268	0.1818	0.3735	0.3724
yeast	0.1668	0.1229	0.5027	0.5019

TABLE 3.5
Average double fault diversity over ten experiments repeated with different random seeds using five fold cross validation and bagging. Lower measurements correspond to more diverse models.

Average Double Fault Diversity				
Dataset	*Naive Bayes Ensemble*	*SVM Ensemble*	*Random Tree Ensemble*	*Best Heterogeneous Ensemble*
abalone	0.5787	0.5475	0.1809	0.1799
bupa	0.2438	0.3469	0.18702	0.18697
dna	0.0509	0.0396	0.0612	0.0359
glass	0.3240	0.2754	0.1228	0.1229
ion	0.1454	0.0947	0.0505	0.0506
promoters	0.0725	0.0543	0.1413	0.0544
sonar	0.2555	0.1528	0.1280	0.1248
yeast	0.3310	0.3626	0.1844	0.1843

formance and diversity? One might assume that if two homogeneous ensembles of the same type with different levels of accuracy were were blended together, that the plots of those measurements would result in a straight line between the end points (or more of a straight line than exhibited by the plots of the heterogeneous ensemble mixtures). To test that, we used five fold cross validation ten times to create ensembles of bagged random trees. We used

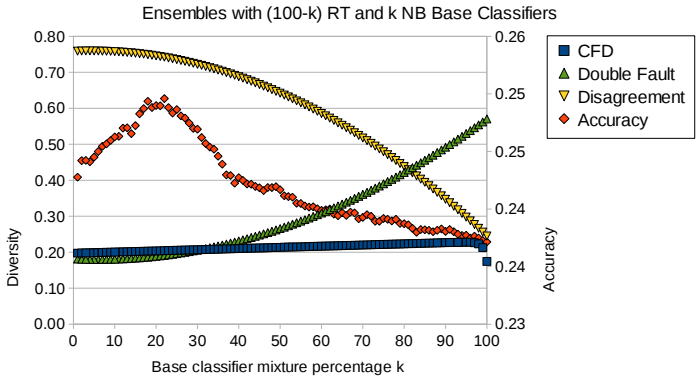


FIG. 3.1. *Comparison of homogeneous and heterogeneous ensembles using the* abalone *data set.*

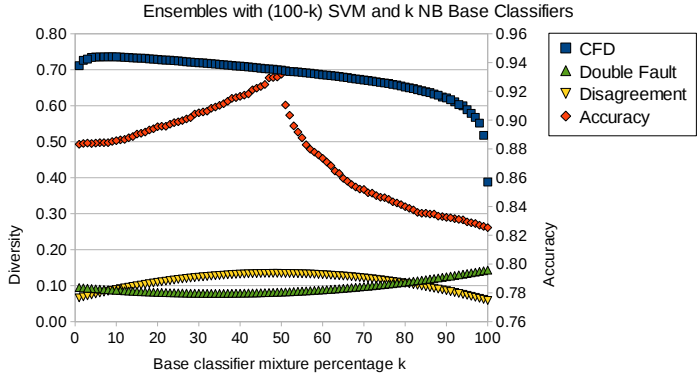


FIG. 3.2. *Comparison of homogeneous and heterogeneous ensembles using the* ion *data set.*

the same random seed for the cross validation each time, but used ten different seeds for the bagging procedure. For each of the five folds we selected the ensembles that had the best and worse performance based on one of the measurements: accuracy, disagreement, double fault, CFD. We then gradually blended the base classifiers from the two ensembles together, and plotted how the performance changed as the composition of base classifiers changed. The plots showed that for most of the measurements, the composition of two homogeneous ensembles of the same type lead to values closer the line than the values measured for the heterogeneous ensembles.

For each of the data sets, we calculated the average distance above the line, $a_{model,measurement}$, and the maximum distance above the line, $m_{model,measurement}$, for the composition of two random tree ensembles, and for the different heterogeneous ensembles. We then calculated $a_{heter,measurement} - a_{homo,measurement}$ for each of the heterogeneous models and each of the measurements. The average difference is important because it may not be practical to search for the best composition in which case an arbitrary heterogeneous composition will be chosen. In that case the differences in the average distances will represent the expected advantage of using a heterogeneous ensemble. We used the Wilcoxon signed rank test to calculate p-values

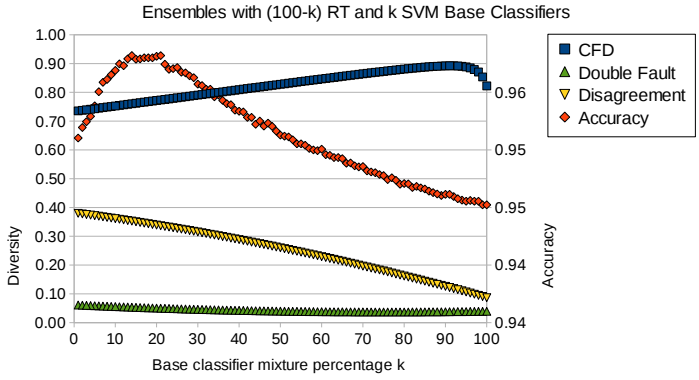


FIG. 3.3. *Comparison of homogeneous and heterogeneous ensembles using the* dna *data set.*

for the hypothesis:

$$H_a : Median(a_{heter,measurement} - a_{homo,measurement}) > 0$$
$$H_0 : Median(a_{heter,measurement} - a_{homo,measurement}) \leq 0$$

Where the median refers to the median value of the difference over all data sets. For this test to be relevant we must assume that the eight data sets used in this study are representative of the greater population of data sets. If the tests ends up lending support to the alternative hypothesis, then more times than not, a data set will be expected to have a larger measurement value above the line for the heterogeneous model when the composition is randomly chosen, as compared to a homogeneous ensemble that is a random mixture of the best and worst ensemble.

We also ran the same tests using only the compositions that had the maximum values for the measurements, rather than the average over all compositions. This was done to test the potential for heterogeneous ensembles to create an advantage over homogeneous ensembles. The following were the hypothesis used in our Wilcoxon signed rank tests:

$$H_a : Median(m_{heter,measurement} - m_{homo,measurement}) > 0$$
$$H_0 : Median(m_{heter,measurement} - m_{homo,measurement}) \leq 0$$

Test that lend support to the alternative hypothesis indicate that when a random data set is chosen, more often than not, the best composition for the heterogeneous will be further from the line than the best composition for the homogeneous mixture for a data set

These results lend support to the theory that the extra performance and diversity are caused by the interactions in the heterogeneous ensemble, for the tests where the p-values are small. These results only hold when the number of base classifiers for the ensembles is arbitrarily fixed (in this case 100).

One thing that was very clear from the plots from the above experiments was that heterogeneous ensemble models can have a much larger range of diversity than homogeneous ensemble models. The plots in Figures 3.4 and 3.5 can be compared to show the differences between heterogeneous mixtures and homogeneous mixtures, respectively, of base classifier sets. These plots are specifically taken from the experiments with the *yeast* data set but the trends found in these two graphs are representative of what is found in other data sets. What we see is that the range of accuracy in the heterogeneous and homogeneous mixtures are about

TABLE 3.6

P-values for significance tests using Wilcoxon signed rank tests. Tests with low p-values imply that the interactions of base classifiers of different models, found in heterogeneous ensembles, lead to higher values of the corresponding measurement than would be expected with homogeneous ensembles.

		P-value	
Measurement	Model	Average	Max
accuracy	NB_RT	1.000	0.926
	NB_SVM	0.027	0.875
	SVM_RT	0.629	0.727
disagreement	NB_RT	0.004	0.004
	NB_SVM	0.004	0.004
	SVM_RT	0.004	0.004
double fault	NB_RT	0.012	1.000
	NB_SVM	0.004	1.000
	SVM_RT	0.055	1.000
CFD	NB_RT	0.004	0.004
	NB_SVM	0.004	0.004
	SVM_RT	0.004	0.004

the same. However the range in diversity measurements for the heterogeneous mixtures are much larger than in the homogeneous mixtures. So even though the homogeneous ensembles show a wide range of accuracy measurements, they show relatively little difference in the amount diversity they display. This may explain why the heterogeneous models are often able have better accuracy than homogeneous models. During training, homogeneous models search through a smaller space in terms of diversity.

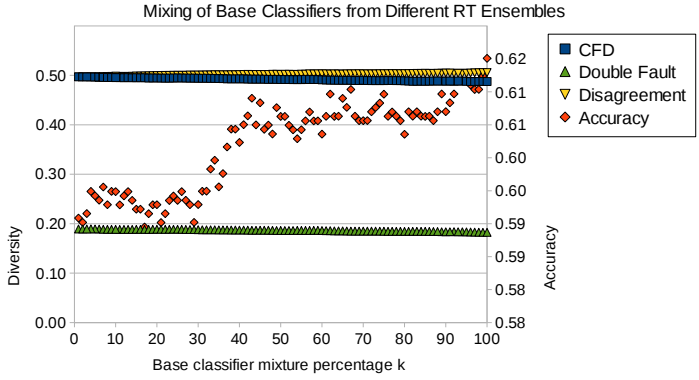


FIG. 3.4. *Heterogeneous ensembles using the* yeast *data set. Ensembles are composed of SVM and random tree base classifiers. The left most value on the x-axis represents homogeneous random tree ensembles, the right most value represents homogeneous SVM ensembles, and everything in between is representative of compositions of both SVM and random tree base classifiers, and are therefore heterogeneous ensembles.*

4. Conclusions and Future Work. Our experiments showed that heterogeneous ensemble models are capable of being more accurate and more diverse than homogeneous ensembles. The method we used performed a search over all possible heterogeneous model compositions involving only two base classifier models. In practice this may not be possible, but also may not be necessary using other fusion functions that re-weight the predictions

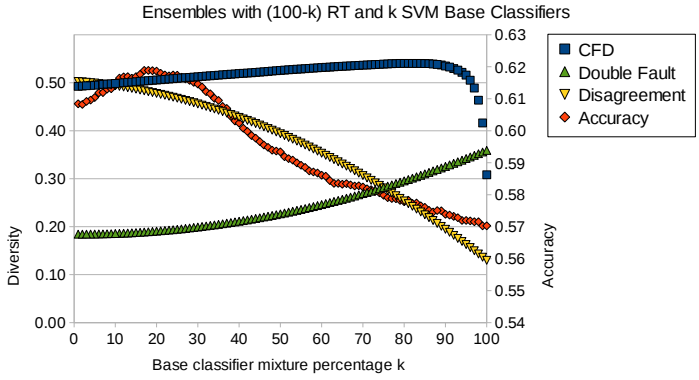


FIG. 3.5. *Homogeneous ensembles using the* yeast *data set. The ensembles are created using a mixture of of the best and worst performing ensembles for the given measurement. The left most value on the x-axis represents the worst ensemble for a measurement, the right most value represents the best ensemble, and everything in between is some combination of the two.*

made by base classifiers. So in the future we would like to run these same experiments use different fusion functions and see how that changes our results.

Another question that arose during our experimentation was whether or not the diversity measurements we used were adequate for for multiclass diversity problems, and whether or not they were appropriate for use with fusion functions that use prediction outputs other than just target class labels. If the fusion function is re-weighting the the outputs of the base classifier predictions, it seems logical that the diversity measurements should re-weight the amount that each base classifier affects the diversity measurement of the ensemble.

We would also like to run these test on more data sets to add significance to our theory that heterogeneous ensembles can increase performance and diversity. We also need to extend these results to the case where the number of base classifiers is not fixed. Ideally, out of bag (OOB) error could be used as a stopping criteria for the set of base classifiers for each ensemble. We would then still be interested if performance and diversity could increase in heterogeneous ensembles, but also if heterogeneous ensembles of significantly smaller sizes could achieve the same performance and diversity as the best homogeneous ensembles.

5. Acknowledgments. We would like to thank Philip Kegelmeyer for providing the data sets for our testing and for helpful suggestions throughout the project. We also thank the developers of the WEKA and JAMA libraries used in HEMLOCK. This project was made possible from funding by the LDRD Program at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

REFERENCES

[1] R. BANFIELD, L. HALL, K. BOWYER, AND W. P. KEGELMEYER, *A comparison of decision tree ensemble creation techniques*, IEEE Trans. Pat. Recog. Mach. Int., 29 (2007), pp. 173–180.

[2] J. BASILICO, D. DUNLAVY, S. VERZI, T. BAUER, AND W. SHANEYFELT, *Yucca mountain LSN archive assistant*, Tech. Rep. SAND2008-1622, Sandia National Laboratories, 2008.

[3] S. BIAN AND W. WANG, *On diversity and accuracy of homogeneous and heterogeneous ensembles*, Intl. J. Hybrid Intel. Sys., 4 (2007), pp. 103–128.

[4] L. BREIMAN, *Bagging predictors*, Machine Learning, 24 (1996), pp. 123–140.

[5] L. Breiman, *Random forests*, Machine Learning, 45 (2001), pp. 5–32.
[6] P. Cunningham and J. Carney, *Diversity versus quality in classification ensembles based on feature selection*, in ECML '00: Proceedings of the 11th European Conference on Machine Learning, London, UK, 2000, Springer-Verlag, pp. 109–116.
[7] G. Giancinto, F. Roli, and P. F. Roli, *Design of effective neural network ensembles for image classification purposes*, Image Vision and Computing Journal, 19 (2001), pp. 699–707.
[8] L. Hansen and P. Salamon, *Neural network ensembles*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 12 (1990), pp. 993–1001.
[9] W. Krzanowski and D. Partridge, *Software diversity: Practical statistics for its measurement and exploitation*, Information & Software Technology, 39 (1996), pp. 39–707.
[10] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.
[11] D. B. Skalak, *The sources of increased accuracy for two proposed boosting algorithms*, in Proc. American Association for Artificial Intelligence (AAAI), Integrating Multiple Learned Models Workshop, 1996, pp. 120–125.
[12] W. Wang, D. Partridge, and J. Etherington, *Hybrid ensembles and coincident failure diversity*, in Proc. International Joint Conference on Neural Networks, 2001.
[13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*, Morgan Kaufmann, June 2005.
[14] G. U. Yule, *On the association of attributes in statistics, with examples from the material of the childhood society*, Proceedings of the Royal Society of London, 66 (1899), pp. 22–23.

PARALLEL MONTE CARLO SIMULATION OF 3D SINTERING

CRISTINA GARCIA*, VEENA TIKARE†, AND STEVEN J. PLIMPTON‡

Abstract. A three-dimensional parallel implementation of a Monte Carlo model for the microstructure evolution during sintering is presented. The model accounts for the main phenomena occurring during sintering, including grain growth, pore migration and vacancy annihilation. The parallel implementation is based on the SPPARKS code and enables the simulation of systems with large number of particles. Several examples are shown and results are compared with a serial implementation as well as experimental data available.

1. Introduction. Sintering is a fabrication process where the fluid-like behavior of powders is exploited to build arbitrarily complex shapes. The consolidated piece is subjected to a firing process leading to strengthening and densification but is also accompanied by volume shrinkage. At the mesoscale level, sintering is the result of thermally activated adhesion processes which produce the bonding between particles and their coalescence [4]. The driving force for sintering is the reduction in surface free energy achieved by diffusional transport of material from the centers of the particles to the particle-particle neck [1].

One effective way of gaining insight into the process is to simulate the physical phenomena occurring during sintering. Different numerical simulation methods have been used, including finite element methods [8], finite difference methods [6], discrete element methods [2] and kinetic Monte Carlo models [3]. As the physico-chemical and mechanical properties of the final piece are determined by the structure of the sintered body there is an interest in studying the microstructural evolution of the material. Furthermore, the microstructural evolution provides the driving force for the deformation observed at the macroscopic level [1]. In this sense, the simplicity and versatility of kinetic Monte Carlo method makes it a sensible choice to investigate the microstructural evolution during sintering [11].

The kinetic Monte Carlo simulation of sintering has been developed in recent years and analysis for simple configurations in 2D [11] have been extended to estimation of macroscale parameters [3] as well as simulations of 3D complex configurations [9, 10]. This stochastic approach uses the Potts model to account for the local kinetics of the process, including grain growth, pore migration and vacancy diffusion and annihilation at grain boundaries. To obtain meaningful results a considerable number of particles must be considered. Nevertheless, simulation of microstructural evolution during sintering on the mesoscale with imaging of many particles sintering is challenging [10]. Therefore it is necessary to build techniques that enable a greater simulation space with capacity to include more particles.

This paper describes a parallel implementation of the kinetic Monte Carlo model for solid-state sintering of a three-dimensional powder compact. The implementation is built as an extension module for SPPARKS. SPPARKS is a kinetic Monte Carlo code designed to run efficiently on parallel computers using both rejection-free kinetic Monte Carlo and Metropolis Monte Carlo algorithms [5]. To verify the performance of the new parallel implementation, sintering for different initial configurations is simulated and results are compared with the serial version as well as some experimental data available. Results are encouraging in terms of the dynamics and the size of the problems that can be attempted.

The document is organized as follows. Section 2 describes the kinetic Monte Carlo model of sintering and the implementation under SPPARKS, and contrasts the differences between the serial and the parallel version. Next, Section 3 compares the results obtained with the

*San Diego State University, cgarcia@sciences.sdsu.edu
†Sandia National Laboratories, vtikare@sandia.gov
‡Sandia National Laboratories, sjplimp@sandia.gov

serial and parallel versions for different initial configurations as well as different problem sizes. Finally, Section 4 draws some conclusions and suggests directions of future work.

2. Simulation. A powder compact is a porous medium composed of powder particles and a phase of voids (pores) that percolate between the particles. The substance is formed by grains that are loosely in contact with each other. During sintering, the surface tension of the powder compact forces a mass movement that redistributes the substance and eventually leads to a reduction in the total porosity, i.e. densification with a corresponding reduction in the dimensions of the porous body. Briefly, the main mechanisms operating during sintering are [10]:

- Curvature driven grain growth in the presence of evolving porosity that inhibits grain growth by pinning.
- Pore migration by surface diffusion leading to pore shape evolution and coarsening.
- Formation of vacancies, grain boundary diffusion of vacancies and vacancy annihilation leading to densification.

To simulate sintering of a three-dimensional powder compact using a Metropolis Monte Carlo model the controlling mechanisms of material flow under sintering must be emulated. Essentially three aspects are defined: (1) a representation of the porous body, (2) a set of events that can transform it in a path-dependent kinetic manner simulated by the Metropolis algorithm and (3) an energy function to drive the Metropolis algorithm used for the dynamic evolution of sintering which is the reduction in interfacial energy.

To represent the material, a 3D cubic lattice structure (a grid) is overlaid in the simulation space and a neighborhood topology is prescribed. For each point in the grid a discrete state is assigned. Grain sites populating the lattice assume one of Q possible distinct states, the individual state is symbolized with q and the total number of states in the system is Q, thus: $q_{grain} \in \{1, 2, \ldots, Q\}$. The pore sites can assume only one state $q_{pore} = 0$. The neighborhood topology used is the 26 first neighbors in the cubic grid. In this model a vacancy is defined as a single, isolated pore site that does not have any other pore site in its neighborhood.

The events that transform the porous body correspond to the mechanisms operating during sintering. Consequently, grain growth is simulated by converting a grain site into the state of a neighboring grain chosen at random. Pore migration is simulated by exchanging a pore site with a neighboring grain site; after the exchange, the grain site assumes a new state corresponding to the state of the neighboring grain that results in the minimum energy (for the grain site, the joint energy for the new pore-grain configuration is not necessarily minimum). Densification is simulated by producing vacancies in the grain boundaries and annihilating them. An annihilation is simulated by moving a vacancy to the surface of the material; such movement proceeds through a path that conserves mass globally and brings the centers of mass of the grains adjacent to the site being annihilated closer together [1]. In addition, note than since this densification algorithm requires the definition of the surface of the material, periodic boundary conditions cannot be used.

The driving force for sintering is the reduction of the interfacial free energy [10]. To cast this condition in terms of the lattice configuration described, the energy of the system is given by the sum of all neighbor interaction energies of all sites:

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{n} \Big(1 - \delta(q_i, q_j)\Big) \tag{2.1}$$

where: N is the total number of sites, n is the number of neighbors (26 nearest neighbors in a cubic grid), q_i is the state of the current site, q_j is the state of the j-th neighbor site and δ is the Kronecker delta with: $\delta(q_i = q_j) = 1$ and $\delta(q_i \neq q_j) = 0$. According to this

energy definition, only unlike neighbors contribute to energy, i.e. only interfacial energy of the system is defined.

The dynamic evolution of the kinetic Monte Carlo model is driven by a reduction in (2.1), by means of a standard Metropolis algorithm. Therefore, every time an event is to be updated, a random number $R \in (0, 1)$ is generated and is compared with the probability P of accepting that change, if $R \leq P$ the change is accepted. The probability P is calculated as:

$$P = \begin{cases} \exp\left(\frac{-\Delta E}{k_B T}\right) & \text{for} \quad \Delta E > 0 \\ 1 & \text{for} \quad \Delta E \leq 0 \end{cases} \tag{2.2}$$

where: ΔE corresponds to the energy change, k_B is the Boltzmann constant and T is the simulation temperature, a measure of the thermal fluctuation in the system [11].

The parameterization of the model enables the consideration of different grain-pore mobility ratios, basically by varying the frequency in which the different events are attempted or using different values for T in (2.2). Similarly, the frequency of annihilation events f_a is adjusted inversely proportional to the average area of grain boundaries:

$$f_a \propto \frac{A_0}{A_{gb}} \tag{2.3}$$

In this equation A_0 stands for the average grain boundary area at the beginning of sintering and A_{gb} for the average grain boundary area at the current time. Time in the simulation is measured in terms of kinetic Monte Carlo steps, where one step corresponds to N attempted changes, being N the number of sites in the system. This simulation time is related linearly with the real sintering time.

In summary, the model described incorporates most of the characteristic phenomena in sintering: interfacial energy related to surface evolution, annihilation to introduce densification, sintering rate proportional to the pore surface area, sintering force inversely proportional to the grain size and change of dimension of the compact. Further details can be found in other references [11, 1, 3].

2.1. SPPARKS. SPPARKS is an acronym for Stochastic Parallel PARticle Kinetic Simulator. SPPARKS is a kinetic Monte Carlo (KMC) code that has algorithms for both rejection-free KMC and rejection KMC which is sometimes called Metropolis Monte Carlo [5].

SPPARKS is distributed as an open source code. It is highly versatile, supports a number of different applications, can be extended to add new functionalities and is able to run in serial or in parallel. The parallel version uses the message passing interface (MPI) to perform concurrent computations over all processors while minimizing communication overhead between processors. To accomplish this, the space of 3D particles (domain) is partitioned between the processors and Monte Carlo dynamics is calculated concurrently in each processor (Figure 2.1(a)). Information about border sites whose neighborhood could partially belong to other processors is kept locally in ghost sites (Figure 2.1(b)). Communications between processors are used to update the state of the ghost sites. They are also required to collect some of the statistical information that can be computed while simulating the evolution of the system at hand.

SPPARKS provides enormous advantages for the parallel implementation of the three-dimensional sintering simulation: it is a standardized framework for Monte Carlo simulation, supporting a versatile interface already tested for serial as well as parallel execution, with efficient diagnostic tools built-in and at the same time, is relatively easy to extend. All these reasons motivate the selection of SPPARKS as the base infrastructure for the parallel implementation of the sintering model.

(a) 3D Domain Partition

(b) Ghost Sites

FIG. 2.1. *Left: 3D Domain partition between processors. Right: Caricature of local site structure, central (owned) sites in white are surrounded by a layer of ghost sites owned by neighboring processors in gray.*

2.2. Serial vs. Parallel Implementation. During the Metropolis Monte Carlo used to simulate sintering, sites are chosen randomly while performing a batch of site-event rejections. These site-events correspond to the mechanisms operating in the sintering process that were described previously. Accordingly, once a local site is selected:

- If it corresponds to a grain site: a grain growth step is attempted. The state of the grain site is converted into the state of a neighboring grain site chosen at random.
- If it corresponds to a pore site: a pore migration step or a vacancy annihilation step is attempted. The pore migration step is simulated by exchanging a pore site with a neighboring grain site. The vacancy annihilation step is computed by moving the vacancy to the surface of the material, while displacing the centers of mass of the grains adjacent to the site being annihilated closer together.

All these events are accepted or rejected according to (2.2), i.e. using a local criterion. Consequently, almost all the events can proceed independently in every processor [7]. The only exception is the annihilation step, because it requires the coordinated modification of sites along a path that, in general, is distributed along several processors. Hence, the main difference between the serial and the parallel implementation is the handling of the annihilation step.

In the serial version, if the vacancy annihilation is to be performed the center of mass of the adjacent grain is calculated, as well as the line along which all sites collapse toward the annihilation site and the annihilation is performed immediately. The path of the annihilation starts in the vacancy current position, goes through the center of mass of the adjacent grain and continues in the same direction until arriving to the surface of the specimen (See Figure 2.2). Thus, the vacancy new position corresponds to the position of the last grain site in the annihilation path. At the same time, all the intermediate sites in the path are shifted one position in the direction toward the vacancy current position. In this way, mass is globally conserved, the centers of mass of the adjacent grains are moved closer together and the compact shrinks.

In the parallel version, since the lattice is distributed spatially over several processors, long range communications are required to perform a vacancy annihilation step. Thus, instead of computing the annihilation immediately it is registered in a list of pending annihilations to be updated at the end of the current batch of site-event rejections. At the end of the batch, the pending list of annihilations is processed normally: calculating the center of mass of the adjacent grain, the vacancy new position and the path of the annihilation. The annihilations in the list are performed one after the other. While performing the batch of site-event rejections, a flag is used to signal if the vacancy is pending annihilation and no further operations are allowed for the site. Nevertheless, it is possible that the actual processing of the annihilations

in the list cause the shifting in vacancies pending annihilation. In those cases, as the vacancy is no longer there no annihilation is performed. The following description summarizes the processing of the list of pending annihilations.

1. Each processor verifies that the local vacancies pending for annihilation are still vacancies and determines the adjacent grain.
2. The list of states of the adjacent grains for all the annihilations pending is gathered in all processors.
3. Each processor calculates the local center of mass of each one of the adjacent grains.
4. The center of mass of all the adjacent grains is reduced (calculated) in all processors.
5. Each processor determines the new position of the local vacancies pending.
6. If the vacancy current position and the new position are in the same processor, the path of the annihilation is calculated and the sites traversed are updated. If current and new position are in different processors, the initial position, the direction of the annihilation path and the number of discrete steps to the final position are stored in a local list.
7. A buffer is created and the annihilation paths that cross multiple processors are gathered in all processors.
8. All the processors follow each annihilation path in the list, consequently every processor knows which sites in all the domain are being modified. When a processor owns part of the sites to be updated, it updates them. It also sends the state of the first local element to the previous processor in the path and waits to receive the update for the last local element from the next processor in the path.

Figures 2.3(a) and 2.3(b) demonstrate some of the conflicting cases occurring while updating the list of annihilations in the parallel implementation. These schematics, drawn for simplicity in 2D and without ghost cells, show a domain divided between four processors and two annihilation paths to be updated: path $\overrightarrow{AB}$ starting at the vacancy site marked A and ending at the border site (surface) marked B and path $\overrightarrow{CD}$ starting at the vacancy site marked C and ending at border site marked D. Annihilation paths are contained in one processor in Figure 2.3(a), while in Figure 2.3(b) they cross multiple processors. If these annihilations are ordered in the pending list in such a way that annihilation $\overrightarrow{AB}$ is first, then both can be performed. On the contrary, if annihilation $\overrightarrow{CD}$ is first in the list, then it is unlikely that annihilation $\overrightarrow{AB}$ takes place in the one-processor case and definitively it is not taking place in the multi-processors case. In the one-processor case, if after updating annihilation $\overrightarrow{CD}$ A is still a pore site, then updating for $\overrightarrow{AB}$ proceeds. However, if A is no longer a pore site then there is no point in processing it as a vacancy being annihilated. In the multi-processors case, even when all processors are following the path of annihilations (to know how and when to exchange border information) only the bottom-right processor knows the new state of site A, only it could verify if it is still a pore site. Thus, rather than incurring in the overhead of communicating the new state for site A, there is a local book-keeping that allows each processor to detect that the site has been changed and avoid computing an annihilation for a vacancy that could no longer be there.

3. Results. To verify the performance of the new parallel implementation, sintering simulations have been conducted for three different initial configurations. The first case studied corresponds to a random initialization and a lattice of: $200 \times 200 \times 50$. The second case is a close packing of spheres in a lattice of: $200 \times 200 \times 200$. The third case is based on a microstructure obtained from microtomographic imaging of a real powder compact of copper particles [10] and a lattice of: $403 \times 403 \times 95$. This section is dedicated to present the results computed with the serial as well as the parallel version. All images corresponding to 3D

views and 2D slices are generated with ParaView, an open-source visualization application (http://www.paraview.org).

The parameters used to run the simulations are included in Table 3.1. There, RFreq stands for the relative frequency for attempting the event, T^* for the temperature factor k_BT to be applied in (2.2) and MCS Start for the Monte Carlo step to start the annihilation of vacancies. The parameters are kept almost equal, the only difference is in the first case where the microstructure has to be evolved, i.e grains should be available, before performing annihilation events. Thus, smaller factors of temperature for grain growth and pore migration are used to decrease the thermal fluctuation in the system while evolving the grain structure.

TABLE 3.1
Simulation Parameters

	Grain Growth		Pore Migration		Annihilation		
Lattice Size	RFreq	T^*	RFreq	T^*	RFreq	T^*	MCS Start
$200 \times 200 \times 50$	0.15	0.1	0.08	0.7	0.77	15.0	616
$200 \times 200 \times 200$	0.15	1.0	0.08	1.0	0.77	15.0	0
$403 \times 403 \times 95$	0.15	1.0	0.08	1.0	0.77	15.0	0

Figure 3.1(a) displays the initial configuration for the first case studied and Figure 3.1(b) contains a 2D slice of the corresponding microstructure. For the initialization, each site inside the simulation space has been assigned with a positive number (grain site) or a zero (pore site) at random, to obtain a starting density of ~70%. All the grain sites start with a different state, thus rather than a real grain structure the starting configuration is a collection of isolated one-site grains. Consequently, just grain growth and pore migration events are performed for a number of Monte Carlo steps until a rough grain structure is obtained. The microstructure built with this procedure constitutes a valid state to start the sintering simulation. Figure 3.2 shows 2D slices of the resulting microstructure obtained for serial and parallel versions. Figure 3.3(a) compares the densification curves vs. Monte Carlo steps for both cases. Different initializations with density of 70% are generated and averaged results are reported. It can be noticed that the evolution is almost identical. Figure 3.3(b) plots the evolution of the grain size (radius) vs. Monte Carlo steps for both cases and again, there is a complete agreement between the two.

Figure 3.4 displays the initial configuration used for the close packing of spheres. Starting from an initial density of 73.1%, both versions reach a density of 94% around 924 Monte Carlo steps. The corresponding microstructures are included in Figure 3.5. Curves of evolution of densification and grain size (radius) vs. Monte Carlo steps for both cases can be encountered in Figure 3.6(a) and Figure 3.6(b), respectively. All the results obtained are similar.

Next, simulation results are compared to experimental data. An input image obtained from micrographic imaging of real powder compact of copper particles is pre-processed to extrapolate a grain structure into the original 3D structure as explained in [10]. The outcome, shown in Figure 3.7, is used as the starting microstructure for the simulation. Results for the parallel implementation can be found in Figure 3.8. Results of a serial implementation of the model originally published in [10] are included in Figure 3.9 to facilitate comparisons. Additionally, Figure 3.10(a) and Figure 3.10(b) compare densification curves and grain size (radius) distributions for serial and parallel versions as well as real data calculated from the microtomographic images of Cu sintering (curves for serial and Cu sintering have been adapted from [10]). The distribution for the parallel version is computed at 82.9 % density, while distributions for serial and Cu sintering are calculated at 83.8% density. The

TABLE 3.2
Simulation Times in [sec]

	Number of Processors						
Lattice Size	1	2	4	8	16	32	64
$200 \times 200 \times 50$	1731	2211	1138	570	349	262	314
$200 \times 200 \times 200$	*	*	15751	8260	4971	2519	1913
$403 \times 403 \times 95$	*	*	*	20660	10236	5674	3287

parallel version seems to have a rate of grain growth similar to the real data, although it has a broader distribution with more fine grains and slightly higher component of coarser grains. It could be that parameters chosen for the parallel version result in a slower sintering, however, investigating the causes of this difference remains a topic for future work. Overall the microstructure evolution observed in the parallel version is comparable to that occurring in the real system.

Finally, Table 3.2 displays average times for execution in a cluster of 32 dual Intel Xeon processors, with 2.8 GHz and 4 GB RAM. The * indicates that the memory requirements have exceeded the available resources in one or several nodes and that the computation could not be done. While time savings are realized with additional processors, the scaling is not ideal, ways to improve performance should be explored.

4. Conclusions. A parallel implementation of a Monte Carlo algorithm for 3D sintering simulation has been described. To verify its performance, different initial configurations were simulated and results were compared with the serial version, and when available, with experimental data. There is an agreement, in the statistical sense, between the expected (serial) and obtained (parallel) microstructure, as well as in the average grain size evolution and the densification curves. Although the scaling in time is not ideal, due primarily to the non-local character of the annihilation algorithm, an important advantage of the parallel version is its ability to process a larger simulation space with capacity to include more particles and resolve microstructures with greater local detail. Nevertheless, future efforts must be directed toward the reduction of the overhead caused by the processing of the list of pending annihilations.

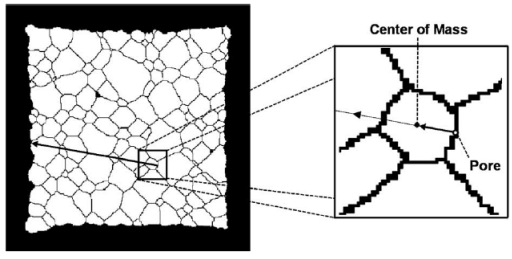
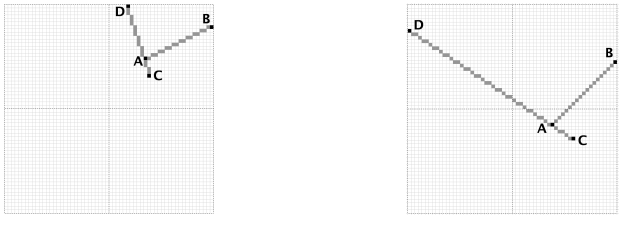
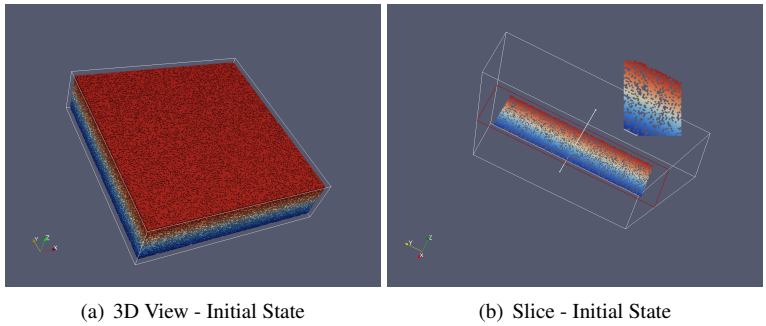


FIG. 2.2. *2D Schematic of vacancy annihilation. Black color denotes pores and grain boundaries; white denotes grains. Image taken from [1].*



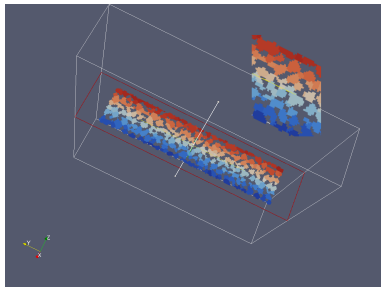
(a) One-processor (b) Multi-processors

FIG. 2.3. *Conflicting cases when updating the list of pending annihilations. Left: required updates contained in one processor. Right: required updates crossing multiple processors.*

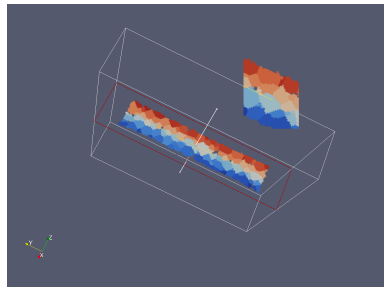


(a) 3D View - Initial State (b) Slice - Initial State

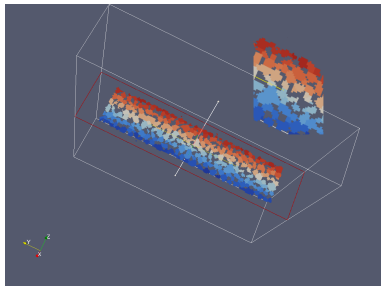
FIG. 3.1. *Initial configuration - Random Initialization. Density: 70.0%. Each grain in microstructure is depicted with a different color. System size:* $200 \times 200 \times 50$.



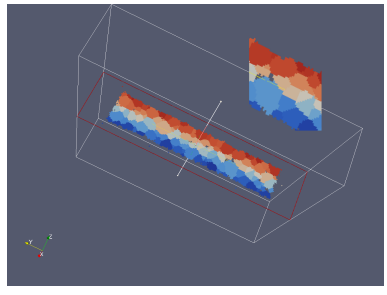
(a) Serial - Slice - 616 MCS



(b) Serial - Slice - 1231 MCS

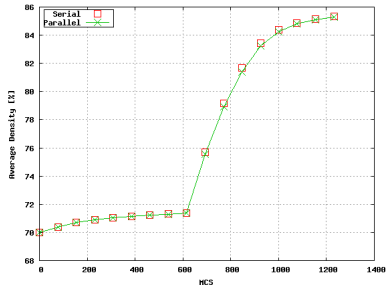


(c) Parallel - Slice - 616 MCS

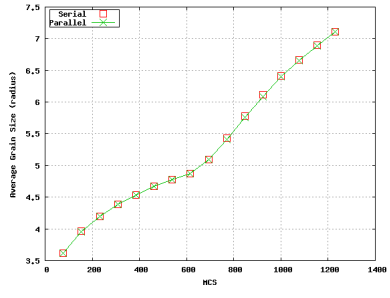


(d) Parallel - Slice - 1231 MCS

FIG. 3.2. *Random Initialization - Initial Density: 70%. Microstructures at 616 and 1231 Monte Carlo Steps. Top: serial version, density: 71.3% and 85.3% respectively. Bottom: parallel version, density: 71.1% and 84.9% respecively*

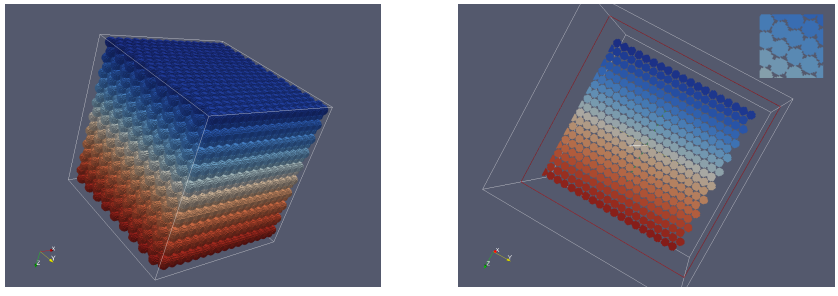


(a) Relative Density



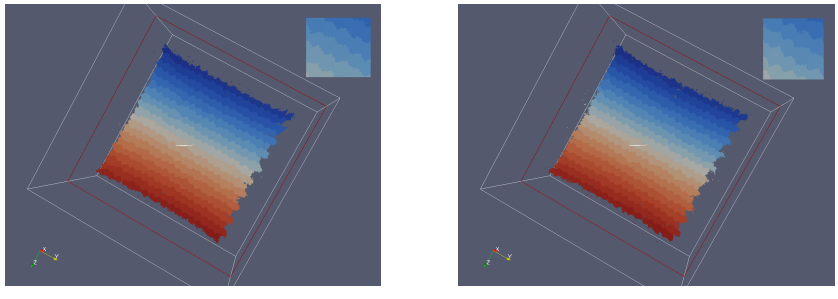
(b) Grain Size (radius)

FIG. 3.3. *Comparison of evolution of densification and grain size for random initialization modeled with serial (squares) and parallel (x's) implementations of the kMC sintering algorithm; lines are added as a guide.*



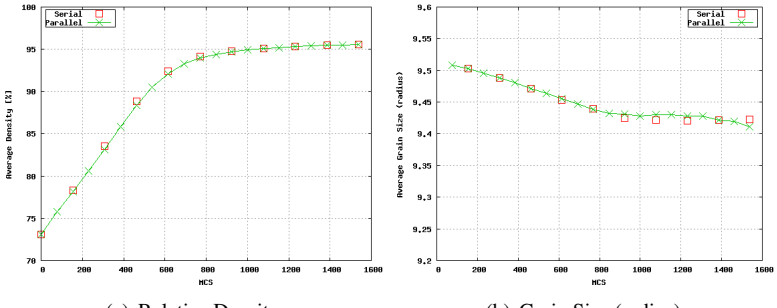
(a) 3D View (b) Slice

FIG. 3.4. *Initial configuration - Close Packing of Spheres. Density: 73.1%. System size:* $200 \times 200 \times 200$.



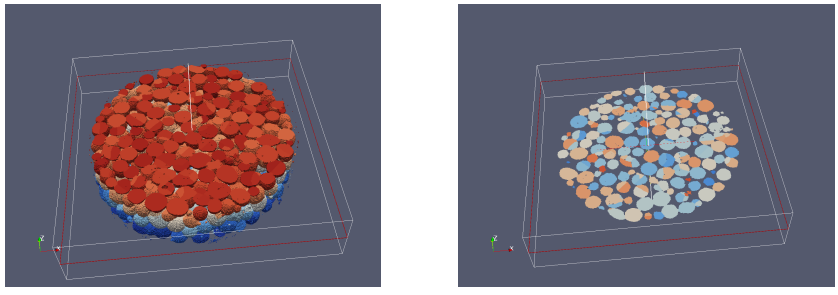
(a) Serial - Slice (b) Parallel - Slice

FIG. 3.5. *Sintering - Close Packing of Spheres: 924 Monte Carlo steps. Left: serial version, Density: 94.8%. Right: parallel version, density 94.7 %.*



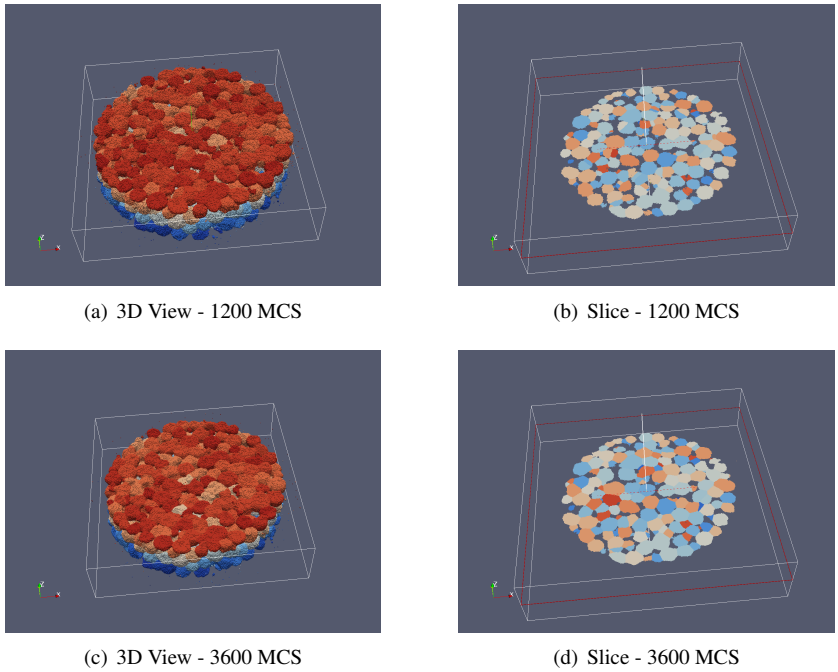
(a) Relative Density (b) Grain Size (radius)

FIG. 3.6. *Comparison of evolution of densification and grain size for close packing of spheres modeled with serial (squares) and parallel (x's) implementations of the kMC sintering algorithm; lines are added as a guide.*



(a) 3D View (b) Slice

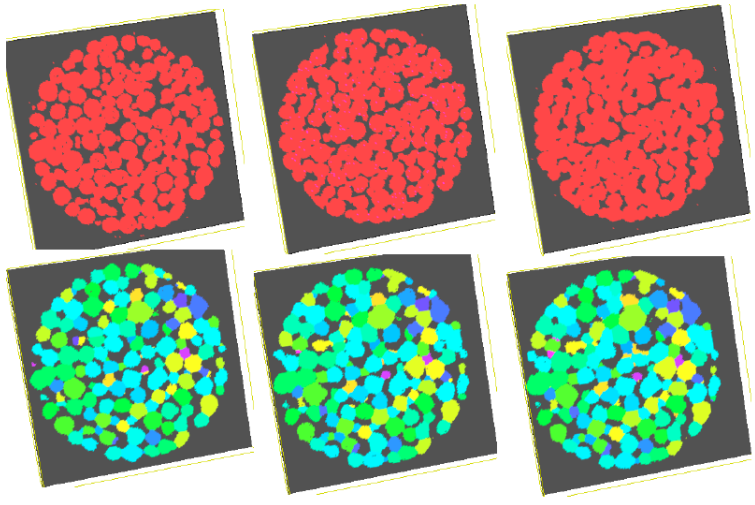
FIG. 3.7. *Microtomographic image with a grain structure extrapolated into the 3D structure. Density: 69.1%. System size:* $403 \times 403 \times 95$.



(a) 3D View - 1200 MCS (b) Slice - 1200 MCS

(c) 3D View - 3600 MCS (d) Slice - 3600 MCS

FIG. 3.8. *Sintering - Parallel Implementation - From Microtomographic Image. Top: 1200 Monte Carlo steps. Density: 78.7%. Bottom: 3600 Monte Carlo steps. Density: 82.9%*

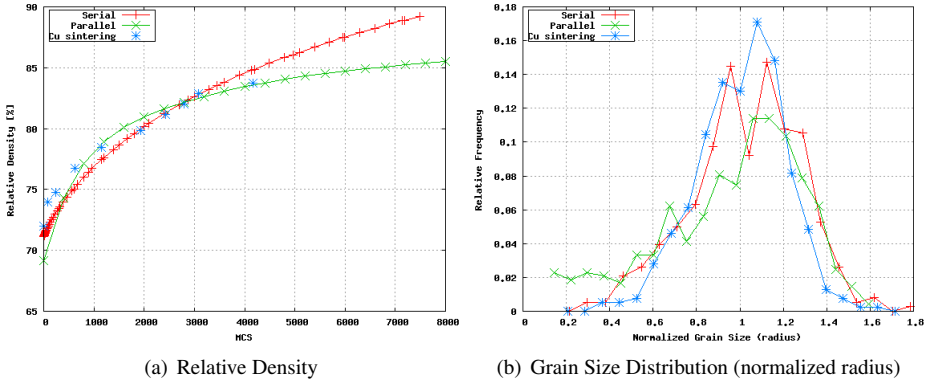


(a) time = 960 MCS
density = 78%

(b) time = 2030 MCS
density = 81%

(c) time = 3482 MCS
density = 84%

FIG. 3.9. *Sintering - From Microtomographic Image. Top: Slices through the 3D microtomographic images. Bottom: Slices through the simulation - Serial Implementation. Image taken from [10].*



(a) Relative Density

(b) Grain Size Distribution (normalized radius)

FIG. 3.10. *Comparison of evolution of densification and grain size for microstructure interpolated in microtomographic image modeled with serial (+'s) and parallel (x's) implementations of the kMC sintering algorithm vs. real data for Cu sintering (*); lines are added as a guide. In both figures, serial and Cu sintering information has been adapted from [10].*

REFERENCES

[1] M. Braginsky, V. Tikare, and E. Olevsky, *Numerical simulation of solid state sintering*, International Journal of Solids and Structures, 42 (2005), pp. 621–636.

[2] A. Jagota and G. Scherer, *Viscosities and sintering rates of a two-dimensional granular composite*, J. Am. Ceram. Soc., 12 (1993), pp. 3123–3135.

[3] E. Olevsky, V. Tikare, and T. Garino, *Multi-scale study of sintering: A review*, J. Am. Ceram. Soc., 89 (2006), pp. 1914–1922.

[4] E. A. Olevsky, *Theory of sintering: from discrete to continuum*, Material Science and Engineering, R23 (1998), pp. 41–100.

[5] S. Plimpton, A. Thompson, and A. Slepoy, *Spparks kinetic monte carlo simulator*. http://www.cs.sandia.gov/~sjplimp/spparks.html.

[6] P. Raj, A. Odulena, and W. R. Cannon, *Anisotropic shrinkage in particle-oriented systems – numerical simulation and experimental studies*, Acta Mater., 50 (2002), pp. 2559–2570.

[7] Y. Shim and J. G. Amar, *Hybrid asynchronous algorithm for parallel kinetic monte carlo simulations of thin film growth*, Journal of Computational Physics, 212 (2006), pp. 305–317.

[8] K. Shinagawa, *Finite element simulation of sintering process*, JSME Int. J. Ser. A, 39 (1996), pp. 565–572.

[9] V. Tikare, *3d numerical simulation of solid state sintering*, (2008). submitted.

[10] V. Tikare, M. Braginsky, D. Bouvard, and A. Vagnon, *An experimental validation of a 3d kinetic, monte carlo model for microstructural evolution during sintering*, CIMTEC, (2009), pp. 1–8.

[11] V. Tikare, M. Braginsky, and E. Olevsky, *Numerical simulation of solid-state sintering: I, sintering of three particles*, J. Am. Ceram. Soc., 86 (2003), pp. 49–53.

L_1-MOR: AN AUTOMATED MODEL ORDER REDUCTION FRAMEWORK BASED ON L_1 NORM AND MOMENT MATCHING CONSTRAINTS

PRATEEK BHANSALI * AND KEITH R. SANTARELLI †

Abstract. In this paper, we propose an automated model order reduction framework for Linear Time Invariant (LTI) systems (including infinite dimensional systems like time delays) via nonlinear optimization techniques. This model order reduction technique dubbed as L_1-MOR is based on L_1 norm minimization subject to moment matching constraints. We use this model order reduction methodology to obtain compact models for the one dimensional heat equation, an RLC filter with ideal transmission line and the Telegraph's equation for RC transmission line. The finite dimensional reduced order model generated by L_1-MOR offers an excellent match for the full simulation results of the original systems.

1. Introduction. Linear Time Invariant (LTI) systems are widely used for modeling physical phenomena. For example, the heat flow in a one dimensional rod, or interconnects in high speed VLSI circuits or power grid can be modeled as LTI dynamical systems. The simulation time and storage requirement becomes unmanageable as size or order of these systems increases. Hence, an emerging practice in the design community is to reduce the order of these high dimensional system using Model Order Reduction (MOR) techniques. MOR results in faster and simpler models suitable for efficient simulation. A reduced order system is an abstract mathematical or look up table based description of the original system. It replicates the behavior of the original system accurately while maintaining the same input and output ports as the original system. In addition to accuracy and speed, the reduced order model should fit well into current simulation flow. This is of particular importance when the original system is ∞-dimensional and can not fit in the current simulation flow directly.
There are several methodologies to generate reduced order models for LTI systems using MOR. Many of these methods can be broadly classified into two major categories [1]

1. Singular Value Decomposition (SVD) based methods, and
2. Moment matching based methods (also known as Krylov subspace methods).

SVD methods computes the reduced order model based on singular values of controllability/observability gramians of the LTI system. Examples of SVD based methods include the Balanced Truncation first introduced in [4]. SVD methods provide a global error bound between the original and reduced order model. Additionally, SVD based methods also preserve properties of the original model such as stability and passivity, [5]. On the other hand, moment matching based methods, like [3], obtain the reduced order model by matching moments of original and reduced order model at specified set of frequencies. While moment matching based reduced order model replicates the behavior of original system well at matching frequencies, it provide no error bounds between the original and reduced order model at other frequencies. Hence, an efficient method is desired which can combine the features of SVD and moment matching based methods. It is in this context, a new framework for producing reduced order model of LTI systems was proposed in [6]. This method produced reduced order model by casting the model order reduction problem as linear programming programming problem subject to moment matching constraints. The implementation in [6] involved a free parameter, α (as discussed in Section 2). For a particular value of the free parameter, chosen by the user, the problem of obtaining a reduced order model such that the peak error between the output of reduced order model and the original system is minimized in L_1 sense, can be cast as a linear programming (LP) problem subject to moment matching constraints. Thus, it attempts to combine the benefits of SVD and moment matching techniques. Moreover, it

*Department of EECS, University of California at Berkeley, bhansali@eecs.berkeley.edu
†Electrical and Microsystems Modeling, Sandia National Laboratories, krsanta@sandia.gov

is possible to obtain rational approximation of ∞-dimensional system using this method directly unlike the SVD based methods or moment based methods. However, this method is inconvenient for software implementation as it requires the free parameter $\alpha \in \mathbb{C}$ to be set by the user. In this work, we cast the problem of L_1 norm minimization based MOR subject to moment matching constraints as a nonlinear optimization problem. This automates the process of L_1 norm minimization based MOR as the parameter α is assumed unknown and is solved along with the minimization of L_1 norm.

This paper is organized as follows: In Section 2, we review the linear programming based framework and its shortcoming. Next, in Section 3, we reformulate the model order reduction methodology as nonlinear optimization problem which enables automation of this novel method of producing reduced order models. In Section 4, we generate the L_1-MOR based reduced order model for the one dimensional heat equation, an RLC filter with ideal transmission line, and the Telegraph's equation for the RC line. We then compare the results with the results of the original system simulations.

2. Background. In this section, we briefly review the linear programming based framework for model order reduction of the LTI systems developed in [6] based on the minimization the L_1 norm. The methodology deals with finding a real number $M > 0$ (preferably small) such that

$$\|y - y_r\|_\infty \leq M\|u\|_\infty \tag{2.1}$$

for every bounded input $u \in L^\infty(\mathbb{R}^+)$ for

$$L^\infty(\mathbb{R}^+) = \left\{ u : [0, \infty) \to \mathbb{R} : \sup_{t \geq 0} |u(t)| < \infty \right\} \tag{2.2}$$

where $y(t)$ is the original system's response and $y_r(t)$ is the response of a reduced order system. The smallest value of real number M for which Eqn. (2.1) holds is given by L_1 norm of the error system $\|h(t) - h_r(t)\|_1$.

$$\|h - h_r\|_1 = \int_0^\infty |h(t) - h_r(t)| dt \tag{2.3}$$

where $h(t)$ and $h_r(t)$ are the impulse response of the original and reduced order system, respectively.

In [6], $h_r(t)$ was constrained to be a linear combination of a fixed set of basis functions. Hence,

$$h_r(t) = \sum_{k=1}^{N} a_k g_k(t) \tag{2.4}$$

where $g_k(t)$,$k = 1, 2, \ldots, N$, represents the elements of the *Ritz basis* and a_k's are scalars. The $g_k(t)$ are given by

$$g_k(t) = \alpha \frac{(\alpha t)^{k-1}}{(k-1)!} e^{-\alpha t} \quad k = 1, 2, \ldots \tag{2.5}$$

with corresponding Laplace transform as

$$G_k(s) = \left(\frac{\alpha}{s + \alpha} \right)^k \quad k = 1, 2, \ldots \tag{2.6}$$

Also, in frequency domain $h(t)$ and $h_r(t)$ are described as

$$\begin{aligned} H(s) &= \mathcal{L}(h(t)) \\ H_r(s) &= \mathcal{L}(h_r(t)) \end{aligned} \tag{2.7}$$

where, $\mathcal{L}$ is the Laplace transform operator. Hence, for this notation, the reduced order model in the frequency domain is given as

$$H_r(s) = \sum_{k=1}^{N} a_k G_k(s) \tag{2.8}$$

With such an $h_r(t)$, the smallest value of M satisfying (2.1) was obtained by minimizing the objective function

$$J = \int_0^\infty |h(t) - \sum_{k=1}^{N} a_k g_k(t)| dt \tag{2.9}$$

subject to moment matching constraints of the form

$$\frac{1}{m!} H^{(m)}(s_l) = \frac{1}{m!} \sum_{k=1}^{N} a_k G_k^{(m)}(s_l) \quad m = 0, 1, \ldots \tag{2.10}$$

at frequency s_l. In Eqn. (2.10), left hand side represents m^{th} moment of the original system and right hand side represents the m^{th} moment of the reduced order system at frequency s_l. It can be easily derived that

$$\begin{aligned} G_k^{(m)}(s) &= \left(\frac{\alpha}{s+\alpha}\right)^k \qquad m = 0 \\ &= \frac{(-1)^m i(i+1)\ldots(i+m-1)\alpha^k}{m!} \left(\frac{1}{s+\alpha}\right)^{k+m} \qquad m = 1, 2, \ldots \end{aligned} \tag{2.11}$$

Now, in [6] Eqn.(2.9) was minimized for several α's. Note, for a constant α Eqn. (2.9) is a linear programming problem with linear constraints given by Eqn. (2.10). This process involved choosing a range $\mathbb{A}$ for α. This range $\mathbb{A}$ was subsequently gridded and an LP problem was solved for every grid point. This issue of gridding the range and solving LP's makes the implementation inefficient. Next, in the Section 3, we formulate the L_1 norm minimization problem of the error system $h(t) - h_r(t)$ as a nonlinear optimization problem which overcomes this shortcoming.

3. Model Order Reduction as Nonlinear Optimization Problem. In this section, the nonlinear formulation is sketched where model reduction problem is cast as a nonlinear optimization problem subject to nonlinear moment matching constraints. Based on Section 2 the model order reduction problem is to minimize the $\|h - h_r\|_1$, where $h(t)$ is original model and $h_r(t)$ is the reduced order model impulse response of the LTI system. That is,

$$\min J = \int_0^\infty |h(t) - h_r(t)| dt \tag{3.1}$$

subject to moment matching constraints

$$\frac{1}{m!} H^{(m)}(s_l) = \frac{1}{m!} \sum_{k=1}^{N} a_k G_k^{(m)}(s_l) \quad m = 0, 1, \ldots \tag{3.2}$$

at a frequency s_l. However, Eqn.(3.1) represents a infinite time horizon optimization problem with non-differentiable cost function over continuous time. To use the standard software packages for optimization we modify the cost function J sequentially as follows. First to resolve the issue of infinite horizon, we proceed as in [6].

3.1. Finite Time Horizon Approximation. Consider $T > 0$ such that we can rewrite the objection function, J, as

$$\begin{aligned} \min \quad & J = \int_0^\infty |h(t) - h_r(t)|dt \\ \min \quad & J = \int_0^T |h(t) - h_r(t)|dt + \int_T^\infty |h(t) - h_r(t)|dt \end{aligned} \tag{3.3}$$

Now, applying the triangle inequality

$$|\sum_{i=1}^{n} a_i| \leq \sum_{i=1}^{n} |a_i| \tag{3.4}$$

on the cost function we have

$$\begin{aligned} \min \quad J &= \int_0^T |h(t) - h_r(t)|dt + \int_T^\infty |h(t) - h_r(t)|dt \\ &\leq \int_0^T |h(t) - h_r(t)|dt + \int_T^\infty |h(t)|dt + \int_T^\infty |h_r(t)|dt \\ &= \int_0^T |h(t) - h_r(t)|dt + \bar{h} + \sum_{k=1}^{N} |a_k| \int_T^\infty |g_k(t,\alpha)|dt \\ &= \int_0^T |h(t) - h_r(t)|dt + \bar{h} + \sum_{k=1}^{N} |a_k \beta_k(\alpha)| \end{aligned} \tag{3.5}$$

where,

$$\begin{aligned} \bar{h} &= \int_T^\infty |h(t)|dt \\ \beta_k(\alpha) &= \int_T^\infty |g_k(t,\alpha)|dt \end{aligned} \tag{3.6}$$

In practice, T can be chosen based on $h(t)$ such that $\bar{h}$ is small. The minimum value of this cost function provides an upper bound on the optimum value of the original cost function.

3.2. Smoothing of the Cost Function. The cost function in Eqn. (3.5) is not differentiable because of absolute value function, $|\cdot|$. Hence, standard gradient-based optimization techniques which computes gradients and Hessian of the objective function can not be used directly for minimization. Thus, we make cost function infinitely differentiable by smoothing technique commonly used in semiconductor device models. In particular, we use $\tanh(\cdot)$ function to smooth the non-differentiable function, $|\cdot|$, which makes the L_1 norm differentiable. The $\tanh(\cdot)$ smoothing for absolute value function is given as

$$|x| \approx x \tanh(Kx) \tag{3.7}$$

where K is a large number. It is possible to obtain bounds on smoothed form as function of K proving that the difference between the true absolute value function and the smoothed function is small. Thus, the smoothed form of the cost function is given as

$$\begin{aligned} \min \quad J &= \int_0^T |h(t) - h_r(t)|dt + \bar{h} + \sum_{k=1}^{N} |a_k \beta_k(\alpha)| \\ &\approx \int_0^T (h(t) - h_r(t)) \tanh(K(h(t) - h_r(t)))dt + \bar{h} + \sum_{k=1}^{N} a_k \beta_k(\alpha) \tanh(K a_k \beta_k(\alpha)) \end{aligned} \tag{3.8}$$

3.3. Discretization of the Cost function. Now, we discretize the continuous time axis and turn the problem into an optimization problem solvable on a computer. There can be many ways to discretize the real time axis, here we choose the most obvious one with uniform step size over the interval $[0, T]$. If Δ is taken as the time step size such that $T = M\Delta$ then the cost function in (3.8) can be approximated via a Riemann sum as

$$\min \quad J = \Delta \sum_{i=1}^{M} (h(i\Delta) - h_r(i\Delta)) \tanh(K(h(i\Delta) - h_r(i\Delta))) + \bar{h} + \sum_{k=1}^{N} a_k \beta_k(\alpha) \tanh(K a_k \beta_k(\alpha)) \tag{3.9}$$

where $M = \frac{T}{\Delta}$. Thus, the optimization problem to be solved is

$$\begin{aligned} \min \quad J &= \Delta \sum_{i=1}^{M} (h(i\Delta) - h_r(i\Delta)) \tanh(K(h(i\Delta) - h_r(i\Delta))) \\ &\quad + \bar{h} + \sum_{k=1}^{N} a_k \beta_k(\alpha) \tanh(K a_k \beta_k(\alpha)) \\ \text{subject to} \quad & \frac{1}{m!} H^{(m)}(s_l) = \frac{1}{m!} \sum_{k=1}^{N} a_k G_k^{(m)}(s_l) \end{aligned} \tag{3.10}$$

where

$$h_r(i\Delta) = \sum_{k=1}^{N} a_k g_k(i\Delta, \alpha) \tag{3.11}$$

for $i = 1, 2, \ldots, M$ and our decision variables are a_k's and α. It is worthwhile to mention that these moment matching constraints are multiplied to Lagrange multipliers in the optimization procedure.

3.4. Stability of the Reduced Order Model. For $\Re(\alpha) > 0$, the reduced order model produced by the L_1-MOR is guaranteed to be stable. To ensure this while optimization problem is solved by the optimizer we add a linear constraint

$$\alpha > 0 \tag{3.12}$$

Now, we use this nonlinear optimization framework to produce guaranteed stable reduced order models.

4. Validation of Model Order Reduction Framework. In this section, we perform an evaluation of the model order reduction framework using three examples. For each example, a optimization problem is set up and its reduced model is generated. Details based on framework developed in Section 3 are provided for the model generation and reduced order model size. The results of the first two examples are compared with the results of LP based formulation in [6]. All simulations were performed on an Intel(R) Xeon(R) 2.67GHz workstation running Linux kernel 2.6.18 using a MATLAB simulation environment. For all the examples to follow, the tanh(·) smoothing parameter K is chosen as 10000.

4.1. One Dimensional Heat Equation. The first example is of a semi-infinite rod described by the half-line $x \geq 0$. Let $u(x,t)$ represent the temperature at the point x along the rod at time t. The partial differential Eqn. (4.1)

$$u_t(x,t) = u_{xx}(x,t) \tag{4.1}$$

is used to model one-dimensional temperature evolution along the rod. Now, a problem of practical interest can be the temperature $u(x = L, t)$ at position $x = L, t \geq 0$ given temperature at $u(x = 0, t)$. For $L = 1$, with an initial temperature distribution $u(x, 0) = 0, x \geq 0$ and boundary constraint, $u(\infty, t) = 0, t \geq 0$, this corresponds to a transfer function

$$H(s) = \frac{U(1,s)}{U(0,s)} = e^{-\sqrt{s}} \tag{4.2}$$

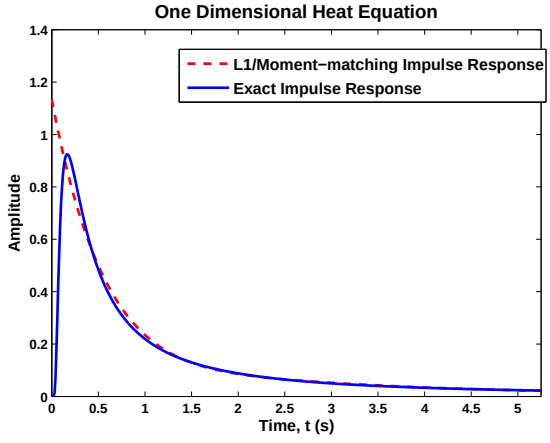
where $U(\cdot, s)$ represents Laplace transform of the $u(\cdot, t)$. The corresponding impulse response is given by

$$h(t) = \frac{1}{\sqrt{4\pi t^3}} e^{-\frac{1}{4t}}, \quad t > 0 \tag{4.3}$$

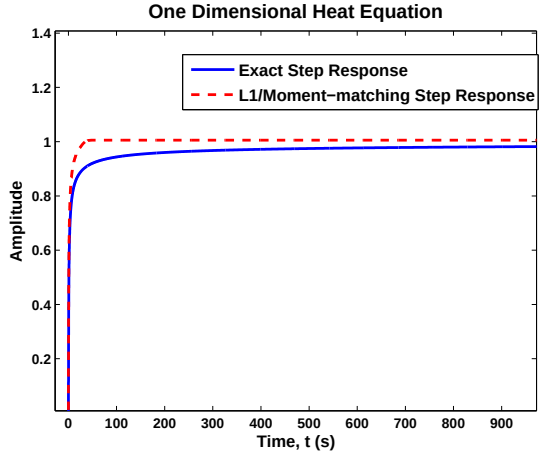
Note that the transfer function $H(s)$ described by equation (4.2) is infinite dimensional and can not be expressed as rational function of s. Because of this reason conventional LTI MOR techniques can not be used to obtain a reduced order model of such systems directly. However, with the L_1-MOR we can obtain rational approximation of $H(s)$ in an automated fashion directly.

Using the procedure described in Section 3 we obtain a reduced order model of size 10 for this system with additional property that the DC gain of the reduced order model and original model match. With an initial guess $\alpha_0 = 51$, the final value of α minimizing the L_1 norm was found to be 0.48 by the L_1-MOR. This value of α bounds the error system norm $\|h - h_r\|_1$ by 0.21. Hence for any bounded input $x(t)$, we are guaranteed that $|y(t) - y_r(t)| \leq 0.21\|x\|_\infty$. Fig. 4.1(a) shows the impulse response of the reduced order model and the original system. Corresponding to this, Fig. 4.1(b) shows the step response of the reduced and the original system. It can be easily verified that the maximum deviation is less than 0.21 between the output of the reduced order system and the original system as guaranteed by L_1-MOR and that their steady state responses converges to one. The value of α was obtained to be 0.5 using the gridding procedure and solving an LP for every grid point in [6] and $min(|h(t) - h_r(t)|_1)$ was found to be 0.206. Hence, we obtain approximately the same bounds on L_1 norm using the nonlinear formulation in an automated fashion.

In the nonlinear formulation, the objective function and constraints are non-convex in nature. Hence, a global optimum is not guaranteed by the optimizers starting from any initial guess of unknowns, a_k's and α. Therefore, we start with different initial guess for α (initial guess for $\{a_k\}_{k=1}^{10}$ was set to be zero) and see the robustness of the method. Fig. 4.2 shows the results obtained starting with different initial guess for α. It can be seen that we obtain similar results as obtained before for a wide range of initial guess.



(a) Impulse Response



(b) Step Response

FIG. 4.1. *Comparison of the impulse responses of original and reduced order model for 1-D heat equation*

4.2. RLC Filter with Time Delay. The second example is of a RLC bandpass filter along with an ideal transmission line. The ideal transmission line provides a time delay, $t_D = 1$. For $R = 1, C = 1/10001, L = 1$, the transfer function of the circuit shown in Fig. 4.3 is given as

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = e^{-s}\frac{2s}{(s+1)^2 + 10,000} \tag{4.4}$$

with corresponding impulse response as

$$\begin{aligned} h(t) &= e^{-(t-1)}(2\cos 100(t-1) - 0.02\sin 100(t-1)) \quad t \geq 1 \\ &= 0 \quad t < 1 \end{aligned} \tag{4.5}$$

We apply the L_1-MOR to find a rational, reduced order model of this infinite dimensional system subject to constraint that $H(100j) = H_r(100j)$, i.e., the response of the reduced order

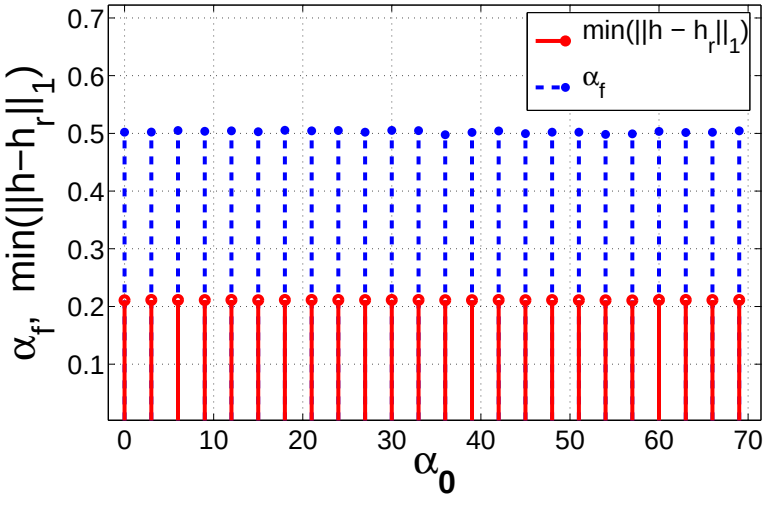


FIG. 4.2. *Optimized L_1 norm with different initial guess, α_0, for one dimensional heat equation*

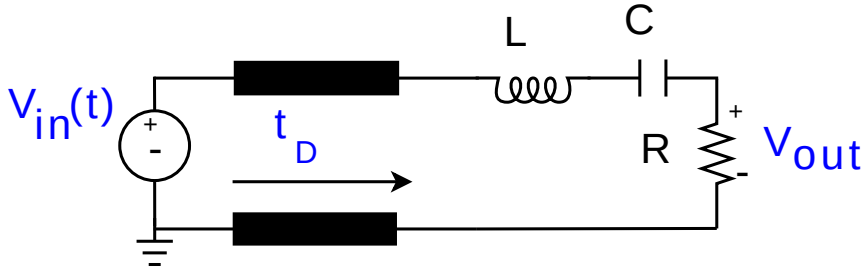


FIG. 4.3. *RLC bandpass filter with an ideal transmission line*

model and the original model matches at the resonant frequency. Note that the impulse response of the original system is oscillatory in nature. Hence, we choose to approximate the impulse response of the original system by the basis of the following form

$$G_k(s) = \left(\frac{\alpha}{s+\alpha+j\omega}\right)^k \quad k = 1, 2, \ldots, N \tag{4.6}$$

This corresponds to the fact that we approximate the original impulse response by

$$h_r(t) = \sum_{k=1}^{N} (a_{2k-1} g_{2k-1}(t) + a_{2k} g_{2k}(t)) \tag{4.7}$$

where

$$\begin{aligned} g_{2k-1}(t) &= \alpha \frac{(\alpha t)^{k-1}}{(k-1)!} e^{-(\alpha t)} \cos(\omega t) \\ g_{2k}(t) &= \alpha \frac{(\alpha t)^{k-1}}{(k-1)!} e^{-(\alpha t)} \sin(\omega t) \end{aligned} \tag{4.8}$$

for $k = 1, 2, \ldots, N$. Using a 12th order model and with an initial guess as $\alpha_0 = 1$ and $\omega_0 = 50$, the L_1-MOR gives us $\alpha = 4.108$ and $\omega = 100.04$. With these values L_1 norm $\|h - h_r\|_1$ is

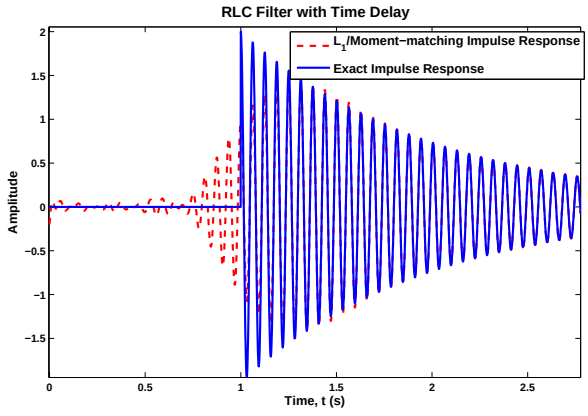
bounded by 0.229. Fig. 4.4(a) shows the impulse response of the reduced order model and the original system. Corresponding to this, Fig. 4.4(b) and Fig. 4.4(c), shows the response of the reduced and the original system driven by sinusoidal signal $\cos(2\pi 100t)$. It can be seen that there is an excellent match between the waveforms generated by the L_1-MOR generated reduced order model and by the original system simulation. Specifically, the reduced order model waveforms matches exactly with the original system at matching frequency in the steady state (Fig. 4.4(c)) which is expected because of the moment matching constraint.

In [6], the value of ω was fixed to be 100 and gridding procedure was applied on α and subsequently solving an LP for every grid point. The value of α was obtained to be 3.25 and $min(|h(t) - h_r(t)|_1)$ was found to be 0.297. Hence, we similar bounds on L_1 norm using the nonlinear formulation in an automated fashion. To see the robustness of the method, we start with different initial guesses for α and ω (initial guess for $\{a_k\}_{k=1}^{24}$ was set to be zero). Fig. 4.5 shows the results obtained starting with different initial guess for α. In Fig. 4.5, the *blue* dots corresponds to the failed results obtained from nonlinear formulation and the *red* dots corresponds to the success of the nonlinear formulation based method. The failed results are defined for which the minimum L_1 norm is unacceptable. For example, in this case we declare the non linear formulation as failure when $min(|h(t) - h_r(t)|_1) > 0.27$. Clearly, as it can be seen in Fig. 4.5, the total success rate is high and is about 89% for various initial guess for α and ω.

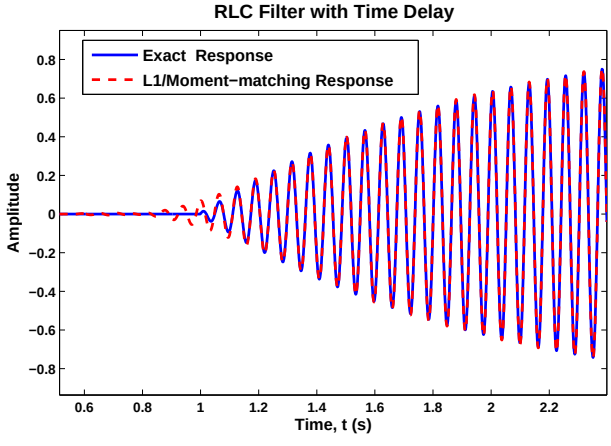
4.3. RC Transmission Line. Our last example is of an RC transmission line. Transmission lines are used to model interconnects in VLSI circuits. Circuit simulators, such as SPICE, models the transmission line as a lumped model. The lumped model consist of RC ladder with hundreds of RC block. This lumped model approximation of transmission line is useful for early design exploration however when more accuracy is desired distributed model should be used[7]. We start directly from the distributed model to obtain Telegraph's equation accurately describing the transmission line. But, the Telegraph's equation gives rise to an irrational transfer function and hence conventional MOR techniques can not be applied directly. However, with L_1-MOR we can directly obtain a reduced order model from Telegraph equation's. For a transmission line driven by a voltage source with source resistance $R_s = 100\Omega$ having resistance per unit length as $r = 0.0075\Omega/\mu m$, capacitance per unit length as $c = 0.038fF/\mu m$, length $l = 3000\mu m$ and load capacitance value $C_L = 0.01pF$ the transfer function from input voltage source to voltage on load capacitance is given by $H(s)$ in Eqn. (4.9).

$$\begin{aligned} Z_L(s) &= \frac{1}{sC_L},\ Z(s) = \frac{r}{sc},\ Z_s(s) = R_s,\ Y(s) = \frac{1}{Z(s)} \\ \gamma(s) &= \sqrt{src}, \\ H(s) &= \frac{Z_L(s)}{[\sinh(\gamma(s)l)(Z_L(s)Z_s(s)Y(s) + Z(s)) + \cosh(\gamma(s)l)(Z_s(s) + Z_L(s))]} \end{aligned} \tag{4.9}$$

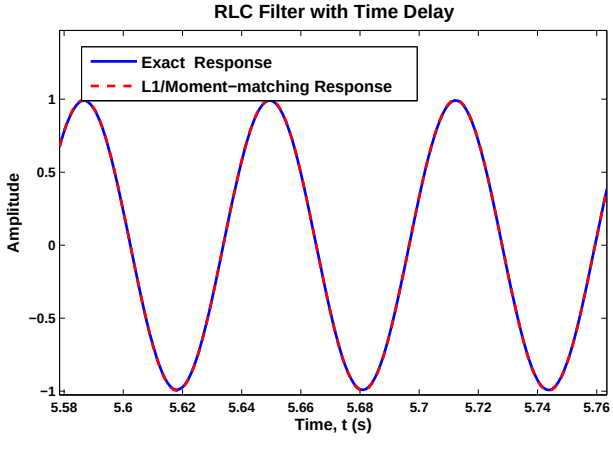
This is an irrational transfer function and it is difficult to obtain the corresponding impulse response of the transmission line analytically or based on inverse Laplace transform tables. Therefore we use an IFFT based technique, outlined in [2], to obtain the impulse response numerically. Using a 20th order model, the L_1-MOR gives us L_1 norm ($||h-h_r||_1$) to be bounded by 0.0528. Fig. 4.6(a) shows the impulse response of the reduced order model and original system. Corresponding to this, Fig. 4.6(b) shows the step response of the reduced and the original system. Clearly, the reduced order model provides an excellent match for the waveforms obtained using the original system simulation.



(a) Impulse Response



(b) Sinusoidal Input Response



(c) Sinusoidal Input Response Zoomed

FIG. 4.4. *Comparison of the impulse responses of original and reduced order model for RLC filter*

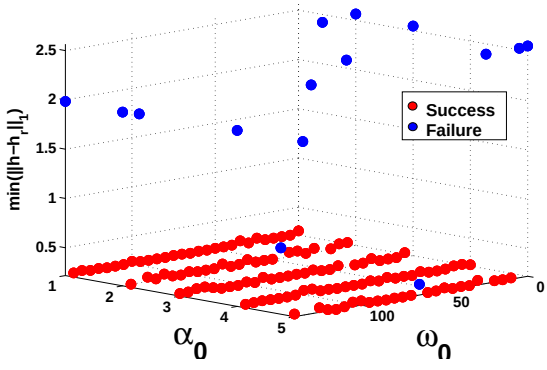
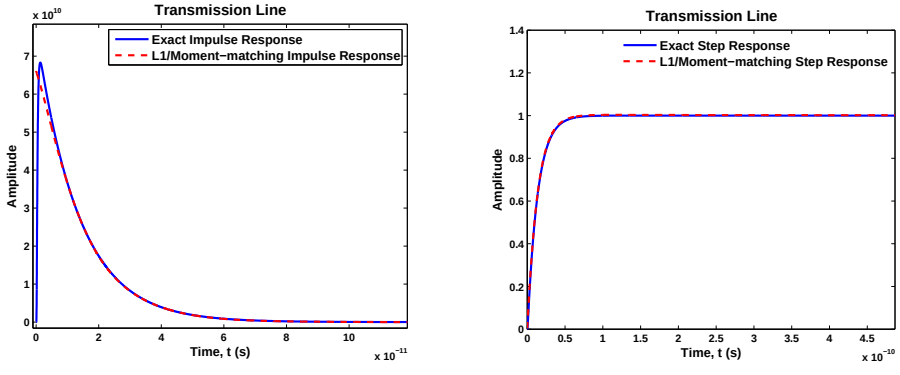


FIG. 4.5. *Optimized L_1 norm with different initial guess, (α_0, ω_0), for the RLC filter*



(a) Impulse Response (b) Step Response

FIG. 4.6. *Comparison of the impulse responses of original and reduced order model for RC transmission line*

5. Conclusions and Future Work. We proposed an automated model order reduction framework called L_1-MOR for linear time invariant (LTI) systems (including the infinite dimensional systems) via nonlinear optimization techniques. The reduced order model generated by L_1-MOR of the one dimensional heat equation, an RLC filter and an RC transmission line offer an excellent match for the exact simulation results of the original infinite dimensional system.

Our future work would involve to use L_1-MOR for lossy *RLGC* transmission line, coupled transmission line networks, and including passivity constraints in the current model order reduction framework.

6. Acknowledgements. The authors would like to thank the following: Eric Keiter, Todd Coffey, Heidi Thornquist, Denis Ridzal and Pavel Bochev (Sandia National Laboratories) for valuable discussions related to the work.

REFERENCES

[1] A. C. ANTOULAS, D. C. SORENSEN, AND S. GUGERCIN, *A survey of model reduction methods for large-scale systems*, Contemporary Mathematics, 280 (2001), pp. 193–219.

[2] P. Gomez and F. A. Uribe, *The numerical laplace transform: An accurate technique for analyzing electromagnetic transients on power system devices*, International Journal of Electrical Power and Energy Systems, 31 (2009), pp. 116–123.
[3] E. J. Grimme, *Krylov projection methods for model reduction*, tech. rep., 1997.
[4] B. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, Automatic Control, IEEE Transactions on, 26 (1981), pp. 17–32.
[5] J. Phillips, L. Daniel, and L. Silveira, *Guaranteed passive balancing transformations for model order reduction*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 22 (2003), pp. 1027–1041.
[6] K. R. Santarelli, *A framework for reduced order modeling with mixed moment matching and peak error objectives*, SANDIA REPORT, (2009).
[7] B. Wong, A. Mittal, Y. Cao, and G. W. Starr, *Nano-CMOS Circuit and Physical Design*, John Wiley and Sons, 2004.

Architecture and Systems Software

Articles in this section discuss advances in high–performance computing architectures and systems software that enhance performance of real–world scientific and engineering applications. Topics include new enhancements to the MPI message passing library that can take advantage of multi-core processors; managing failures in every-increasingly complex high performance computing systems; and developing effective tools to model and simulation next generation computing systems.

Z. Wen
S.S. Collis

January 11, 2010

AN API FOR SMARTMAP AND ITS APPLICATIONS

RON BRIGHTWELL, ZHAOFANG WEN* AND JUNFENG WU, LIANG ZHAO†

Abstract. SMARTMAP is a special operating system capability that enables processes of an application running on a multi-core processor to directly access each other's data in the on-node shared memory. This paper presents an API to expose the SMARTMAP capability to the application programmers, so that they can write algorithms to directly control data exchange between processes running on the same node. Data exchange across nodes still uses message-passing (e.g MPI) over the network. In comparison, MPI programmers on multi-core clusters treat all the inter-process data exchanges the same way; it is up to MPI to separate and schedule the on-node traffic (serviced by shared memory) and the network traffic (serviced by network communication). The SMARTMAP API is compatible with MPI; when used together, they provide more opportunities than MPI alone for algorithms to exploit both the parallelism at the cores and the parallelism between the nodes. Algorithms that express the inherent parallelism in the application problem in two levels (two-level algorithms) would be well-suited; but algorithms expressing parallelism in one level (one-level algorithm) can also benefit from programming in MPI plus SMARTMAP API. For performance comparison of MPI alone vs. MPI plus SMARTMAP API, data from several micro applications are presented.

1. Introduction. The main trend of computer processor development since 2005 has turned from increasing working frequency to increasing the number of cores on a chip. And the core count on processors used for high performance computing continues to increase, which will double every two to three years based on the prediction of the 2007 International Technology Roadmap for Semiconductors(ITRS)[1]. This hardware trend demands fundamental changes in parallel software, including programming environments to suit the new architecture and ways to design efficient parallel algorithms and applications.

Cluster computing is the main form of parallel computing. To look into how multi-core processors affect cluster computing, we investigate the communication of multi-core computer. The target architecture is such a cluster that its nodes are equipped with multi-core processors and each core of these many-core processors is as functional as a whole mono-core processor in the past. Note that on each node, the multiple processor cores use the same Network Interface Controller(NIC), causing large contentions that significantly slow down the communication. So it is necessary that the software consciously instructs these cores to alleviate these contentions. Furthermore, the fact is that the communication between two cores of the same node has no need to flow through the NIC since the on-node shared memory can serve the purpose much better, as long as the software consciously informs the cores to exploit it. There is much previous work on how to utilize the multi-core cluster for high performance computing, especially based on MPI. For example, a hybrid MPI/OpenMP parallel programming model is one of proposed solutions to this issue [8]. It uses OpenMP for parallelization inside the node and MPI for message passing between nodes. However, from a programmer's point of view, traditional MPI ignores the fact that cores inside a single node work on shared memory. It can be employed right away on the multi-core cluster without changes to existing code, which saves a lot of work while on the other hand, it certainly adds to overhead since all communication between processes on the same node goes through the MPI software layers. Thus a lot of work has been done using shared memory for intra-node MPI communications.

A recently introduced operating system capability, Simple Mapping of Address Region Tables for Multi-core Aware Programming, or SMARTMAP[4, 2], provides an efficient way in which applications access on-node shared memory. This scheme offers us an opportunity to alleviate the network contentions and exploit the on-node shared memory, granting

*Sandia National Laboratories,{rbbrigh,zwen}@sandia.gov,
†Syracuse University, {juwu, lzhao04}@syr.edu

the parallel programs the ability to consciously unleash the computing power of a multi-core cluster. Kevin Pedretti and Ron Brightwell have modified an MPI implementation using SMARTMAP and proved that the SMARTMAP capability is able to deliver significant performance improvements for MPI point-to-point and collective operations.

This paper presents an API to expose the SMARTMAP capability to the application programmers, so that they can write algorithms to directly control data exchange between processes running on the same node. Data exchange across nodes still uses message-passing (e.g MPI) over the network. In comparison, MPI programmers on multi-core clusters treat all the inter-process data exchanges the same way; it is up to MPI to separate and schedule the on-node traffic (serviced by shared memory) and the network traffic (serviced by network communication). The SMARTMAP API is compatible with MPI; when used together, they provide more opportunities than MPI alone for algorithms to exploit both the parallelism at the cores and the parallelism between the nodes. Algorithms that express the inherent parallelism in the application problem in two levels (two-level algorithms) would be well-suited; but algorithms expressing parallelism in one level (one-level algorithm) can also benefit from programming in MPI plus SMARTMAP API. For performance comparison of MPI alone vs. MPI plus SMARTMAP API, data from several micro applications are presented.

Section 2 begins with a brief description of SMARTMAP. Section 3 describes the function and the design of this MPI extension. And Section 4 provides some implementations on two-level algorithm using this MPI extension.

2. SMARTMAP. SMARTMAP [2, 4] is a virtual memory mapping technique in the operating system that allows for direct access shared memory between the processes running on a multi-core processor. It is implemented in the Catamount lightweight kernel for the Cray XT [3]. The Catamount lightweight kernel is a third-generation compute node operating system developed by Sandia National Laboratories and Cray, Inc., as part of the Sandia/Cray Red Storm projects. Catamount has several unique features that are designed to optimize performance and scalability specifically for a distributed memory message passing-based parallel computing platform.

SMARTMAP takes advantages of the fact that Catamount only uses a single entry in the top-level page table mapping structure (PML4) on each X86-64 (AMD Opteron or Intel EM64T) core. Each PML4 slot covers 39 bits of address space, or 512 GB of memory. Normally, Catamount only uses the first entry covering physical addresses in the range 0x0 to 0x007FFFFFFFFF. The X86-64 architecture supports a 48-bit address space, so there are 512 entries in the PML4.

Each core writes the pointer to its PML4 table into an array at core 0 startup. Each time the kernel enters the routine to start a new context, the kernel copies all of the PML4 entries from every core into every other core. This allows every process on a node to see every other process's view of the virtual memory across the node, at a fixed and easily computed offset into its own virtual address space.

SMARTMAP also takes advantages of Catamount's physically contiguous address space mapping and the fact that the address mappings are static. This means that the virtual address of local variables within a SPMD (Single Program Multiple Data) program is the same everywhere. The following runtime can be used for converting a local virtual address into a remote virtual address for a process on a different core:

```
static inline void * remote_address( unsigned core, void * vaddr)
{
  uintptr_t addr = ( uintptr_t) addr;
  addr |= (( uintptr_t ) ( core + 1 )) << 39;
  return ( void* ) addr;
}
```

In other words, a SMARTMAP address is an integer with 64 bits, in which the lower 39 bits hold the private address of the region in the process allocating it, while the higher bits are set to the core IDs of the region-allocating process plus 1, to distinguish the SMARTMAP addresses from private address. For example, for an on-node shared memory region allocated by core 3 and having private address 0x80465a8b on that core, the SMARTMAP address is $0x80465a8b + 2^{39}(3 + 1) = 0x020080465a8b$.

3. A SMARTMAP API and Extensions. This SMARTMAP API is developed for two purposes: (1) we provide this SMARTMAP API to explore ways for applications to use the SMARTMAP capability directly, through library interfaces that allow processes to do direct remote loads and stores. (2) SMARTMAP only works for static variables and we will extend it to dynamic variables.

3.1. Design and Specification of The SMARTMAP API. This SMARTMAP API includes three basic operations for an on-node shared memory region: (i) remote get of data in a region, (ii) remote put of data in a region, and (iii) fetching remote array address. Extended from the basic SMARTMAP capability, these API functions are applied to both static and dynamic variables in different ways. For a static variable, these functions are just the direct application of SMARTMAP. Two other operations, allocate() and free(), are available for dynamic variables. In order to support these operations on a dynamic variable, a special static data structure, called SMARTMAP REGION TABLE, is used in the library to implement this function. (Note: One such special table is maintained on each core.) The SMARTMAP REGION TABLE records the location of dynamically allocated space, by region ID, which is also used by the library to find the dynamic space when the API functions are called. Specifically, the basic API functions are listed below.

```
void SMAP_initialize(int *argc, char **argv);
     Initialize the SMAPMAP environment. Must call this
     function before the SMAPMAP extension functions are
     applied. It is only called once for
     each program.

void SMAP_finalize();
     Wrap up the SMAPMAP applications. Call this function
     at the end of SMAPMAP extension functions. It is called
     only once for each program.

void *SMAP_allocate(int region_id, size_t size);
     By calling this function, a process (core) allocates
     a piece of dynamic space of 'size'; and this space is
     registered with the 'region_id'.

void SMAP_free(int region_id);
     De-allocate the dynamic space by 'SMAP_allocate()' as
     referred by 'region_id'.

void SMAP_put_to(bool is_dynamic, int core_id, int region_id,
                 uintptr_t offset, void *data, size_t size);
     Copy (this core's) 'data' to remote 'core_id' at space
     'region_id'.The copied data is put to the position
     'offset' and is has 'size'.

void SMAP_get_from(bool is_dynamic, int core_id,
        int region_id, uintptr_t offset, void *data, size_t size);
     Copy remote data from 'core_id' at space 'region_id'
     into local variable 'data'. The data copied starts
     from 'offset' and is with length 'size'.
```

```
void * SMAP_array_address(bool is_dynamic, int core_id,
                          int region_id);
     Get the address of the remote data on core_id
     registered with region_id. The address returned can
     be used locally just like a local array address.
```

In addition to the basic API functions for the SMARTMAP, there are some useful utility functions, listed below.

```
int SYS_proc_count();
     This function returns the total number of processes
     in the program, which is also the total number of
     cores in the SPMD program.

int SYS_node_count();
     This function returns the total number of nodes used
     by the program.

int SYS_core_count();
     This function returns the total number of cores on this node.

int SYS_my_proc_id();
     This function returns an integer, between 0 and
     SYS_proc_count - 1, assigned to each process.
     And it equals to (SYS_node_id * SYS_core_count + SYS_core_id).

int SYS_my_node_id();
     This function returns an integer between 0
     and SYS_node_count - 1, assigned to each node
     used in the program.

int SYS_my_core_id();
     This function returns an integer between 0
     and SYS_core_count - 1, assigned to each
     core on each node.

MPI_Comm SYS_proc_comm();
     This function returns an communicator for all of the
     processes in the program.  This is just a wrapper of
     the MPI communicator.

MPI_Comm SYS_node_comm();
     This function returns an communicator for all of
     the cores on the same node.

void SYS_barrier_node();
     This function returns a barrier for all of the
     cores on the same node.
```

3.2. One Example of Extension to This SMARTMAP API. We want to mention that the APIs listed above are just the basic functions that we need for the communication of two cores. But it also provides us some opportunities to create other more complex functions based on those API's. To end this section, we would like to present an example using these basic SMARTMAP API functions to create the on-node barrier, the barrier for all of the cores on the same node. For the well structured two-level algorithm, it is necessary to use an on-node barrier synchronization to control processes. This SMARTMAP API provides a mechanism for users to synchronize the processing on the same node without using NIC. Here is an example to show how to create such a barrier. We can create an array, in which each array element corresponds to a core on the node, with the array elements all initialized to 0, serving as a flag to the status of the corresponding cores. According to these flags, we

can lock and unlock each of these cores. And this idea can also be applied to create a barrier synchronization for any subgroup of the cores on the same node. The algorithm to implement this barrier is given below.

Input: The list of the IDs of the cores to participate in the barrier synchronization (this can be an array containing the IDs of the cores).

Output: None.

Extensions to the Basic API: Function SMAP_initialize() dynamically creates a special vector (”barrier vector”) and another variable (”message”) using two special ”region IDs” that are constants visible to the API users. The element count of the special vector is equal to the number of cores on the node; and every element is initialized to 0. Variable ”message” is initialized to false.

Algorithm

- For the first core,
  - Scan its own ”barrier vector”. It will not scan the next element until the current element is modified to 1.
  - Set its own ”message” to be true.
- For the other cores, do the followings in parallel,
  - use SMAP_put_to_ to modify the corresponding element of the the ”barrier vector” on the first core to 1.
  - use SMAP_get_from_ to get the value of variable ”message” from the first core repeatedly until the result is ”true”.

4. Applications. This SMARTMAP API enables direct data access between cores on the same node. Combining MPI and SMARTMAP API supports algorithms with one-level parallelism or two levels of parallelism. For experiment and performance comparison with pure MPI, we implemented three parallel micro applications. The first one application, Jacobi Solver, is based on an algorithm with one-level parallelism. The second application is Conjugate Gradient solver(CG). For the MPI implementation, the CG algorithm used has one-level of parallelism; every core is treated by MPI as if the core is a standalone node. The (MPI + SMARTMAP API) implementation of CG is based on a two-level parallel algorithm, data exchange between cores on the same node is done using the SMARTMAP API; while communication between nodes is done using MPI, with one core per node for handling the communication traffic for the node. The third application is Matrix-Vector multiply for dense matrix; the MPI implementation is based on an algorithm with one-level of parallelism; and the (MPI + SMARTMAP API) implementation is based on an two-level parallel algorithm.

4.1. Machine Platform. The Red Storm qualification system is a single-cabinet Cray XT3/4 system with 80 compute nodes. Approximately half of the nodes have AMD Opteron 2.2 GHz quad-core processors with 4GB of DDR2 667 MHz memory per node. The other compute nodes have 2.4 GHz dual-core processors. The system runs UNICOS 2.0.62 and the compute nodes run the Catamount N-Way light weight kernel operating system.

4.2. Application 1: Jacobi Solver. The Jocobi method is an iteration method to solve discrete Poisson equation. The domain of the Poisson equation solved here is a $[0, 1] * [0, 1]$ square in 2-D space. For the simple algorithm introduced in [6], the $[0, 1] * [0, 1]$ square is divided into a $N * N$ mesh. And each process is responsible to compute one part of the points in this mesh. Each process has to exchange data, those so called ghost points, with its neighborhoods at each iteration step. We realized that this algorithm involves a lot of point-to-point communications. And most of these point-to-point communications are actually between two cores on the same node, which gives us an opportunity to improve the performance using this SMARTMAP API. Here, the two implementations using MPI and us-

ing (MPI + SMARTMAP API) are based on the same algorithm with one level of parallelism. The difference is that in the (MPI + SMARTMAP API) implementation, data exchange with the ghost-points on the same processing node is done using the SMARTMAP API; while data exchange with ghost-points on different processing nodes is done using MPI. We expected a better performance using this SMARTMAP API based on the advantages of SMARTMAP.

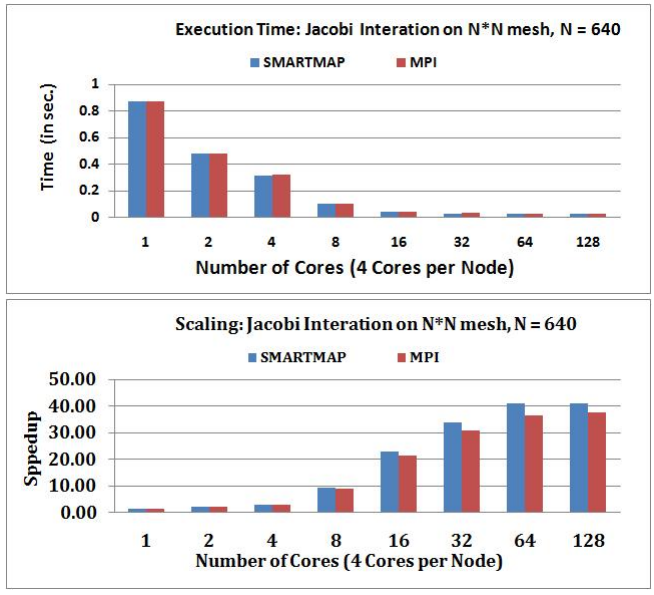


FIG. 4.1. *Execution time and the scaling of the Jacobi method*

However, seen from figure 4.1, there is basically no difference between the performance using MPI only and the application using (MPI + SMARTMAP API). We think the reason is that we apply the MPI algorithm directly to the application of this SMARTMAP API. And the MPI probably has been optimized for the point-to-point communication on the same node. The other reason is that the communication pattern in this algorithm is very regular and not too much data is involved in the communication. Thus there is not too much contention in NIC. So this test tells us that to fully utilize this SMARTMAP API and take advantage of the on-node shared memory, we need to design the more delicate and efficient two-level algorithms.

4.3. Application 2: Conjugate-Gradient Solver. Designing an efficient two-level algorithm is not easy. The first two-level algorithm we provide here is the Conjugate Gradient(CG) solver for sparse matrix. The conjugate gradient method is an iterative method, so it can be applied to sparse systems that are too large to be handled by direct methods such as the Cholesky decomposition [5]. And the parallel algorithm for CG is also widely studied, such as in [7]. The linear system solved in this program is from the diffusion problem on 3D chimney domain by a 27 point implicit finite difference scheme with unstructured data formats and communication patterns. There are three basic algebra operations in CG, sparse matrix-vector multiplication, vector inner product and saxpy. Saxpy is embarrassingly paralleled while there is no communication among different processes in this step. Vector inner product is also very well formulated in MPI, which can be done by a single function call MPI_Reduce and the communication is also very regular. Thus, in our two-level CG algorithm, we do not change anything in these two operations compared to MPI. We focus on the

most time-consuming step in this method, sparse matrix-vector multiplication. In the MPI implementation, each process bundles up the communication messages before sending them over the network. And each process will probably communicate with any other process due to the irregularity of the sparse matrix. In our two level algorithm, we choose only one core on each node to communicate with other nodes to release the burden of NIC and reduce the communication cost. Thus the data on one node is actually stored on one core, which is responsible for communication. And other cores on each node will do their own computation and use the SMARTMAP API to move the data between cores on the same node. Specifically, the two-level sparse matrix-vector multiplication algorithm is described as the following:

- The data of one node is stored on only one core of the node. Each core uses a SMAP_dynamic_ address to refer to the data it operates to and does some work to bundle the communication information.
- Only the core storing data on each node does the communication.
- Each core uses SMAP_dynamic_ address again to refer to the data assigned to it and does the computation.

Thus, this algorithm is still work balanced considering each process will do the same work including bundling communication messages and doing computations, though it is not storage balanced. This strategy reduces the total number of communication and we do expect some performance improvement from this two-level algorithm. We apply both of two-level algorithm and the MPI algorithm to different sizes of matrix with the same structure.

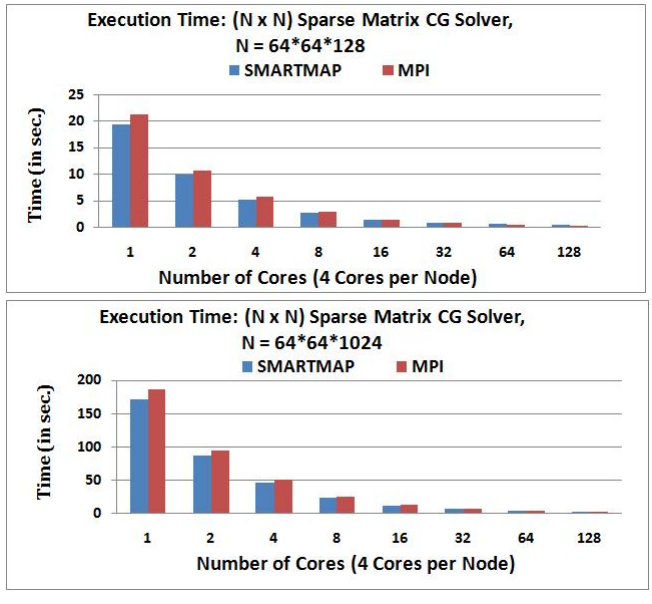


FIG. 4.2. *Execution time of CG solver for different sizes of sparse matrix*

However, from figures 4.2 and 4.3, we can see that the implementation using (MPI + SMARTMAP API) is actually slower than the MPI implementation, especially for the smaller size of matrix. To explore the reasons for this, let us look at the major differences of these two algorithms. In the (MPI + SMARTMAP API) implementation, one core (the communicating core) on each node is responsible for calling MPI to do remote data transfer, but before that, there is one step: all the cores on the node are responsible for preparing the data bundles; the prepared bundles are then moved to a memory region for the communicating core to invoke MPI to do the remote data transfer. This preparation step requires one barrier synchronization.

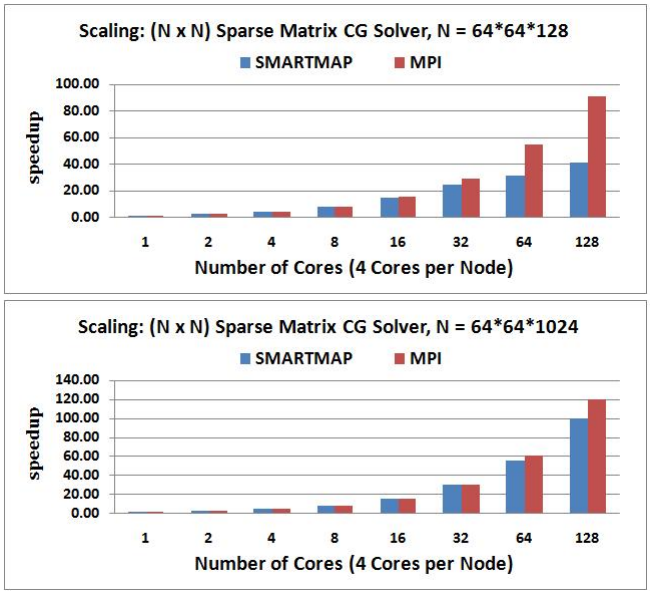


FIG. 4.3. *Scaling of CG solver for different sizes of sparse matrix*

Our explanation for the reason why (MPI + SMARTMAP API) implementation is slower is because the saving in avoid the contention in the NIC by cores (as in the MPI implementation) is probably outweighed by the cost of this extra core-level synchronization.

4.4. Application 3: Matrix Vector Multiplication. The third application is a Matrix-Vector multiplication. Its algorithm can take full advantage of the two level parallelism, which is readily supported by the (MPI + SMARTMAP API). For this parallel operation, each process will store one part of the vector and we need to carry out an all-to-all communication before computation. Thus every process needs to send and receive information from every other core. This is a very complicated collective communication. Though it can be done by one MPI call MPI_Alltoall in the one level algorithm, there is much improvement that can be done to take more benefits from the multi-core cluster. We adopt the fancier two-level algorithm appeared in [9] in our implementation of (MPI + SMARTMAP API). Specifically, for an all-to-all collective communication performed on n cores of a cluster, which has P cores on every node, it can be broken down to the following three steps.

Step 1 let the cores on each node aggregate their outgoing messages into n-1 aggregated messages according to the receiver nodes, and label these outgoing aggregated messages with indices 0,1, ... , n-2. This is an inter-node communication, which can be done using this SMARTMAP API.

Step 2 in the inter-node communication, let core $l(l = 0, 1, ...P - 1)$ send the outgoing aggregated messages of indices

$$i : [(n-1)l/P] < i < [(n-1)(l+1)/P]$$

and receive the incoming aggregated messages from the receiver nodes of these outgoing aggregated messages. This can be done using the common MPI receive and send calls.

Step 3 let the cores on each node to distribute the incoming messages and the intranode

messages according to the receiver cores. Again, all of the communications involved in this step are intra-node communications, which SMARTMAP API can realize very well.

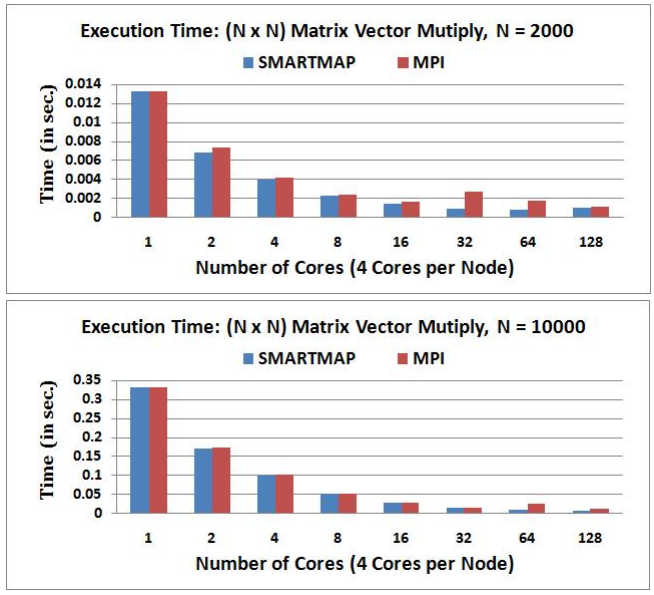


FIG. 4.4. *Execution time of different sizes of matrix vector multiply*

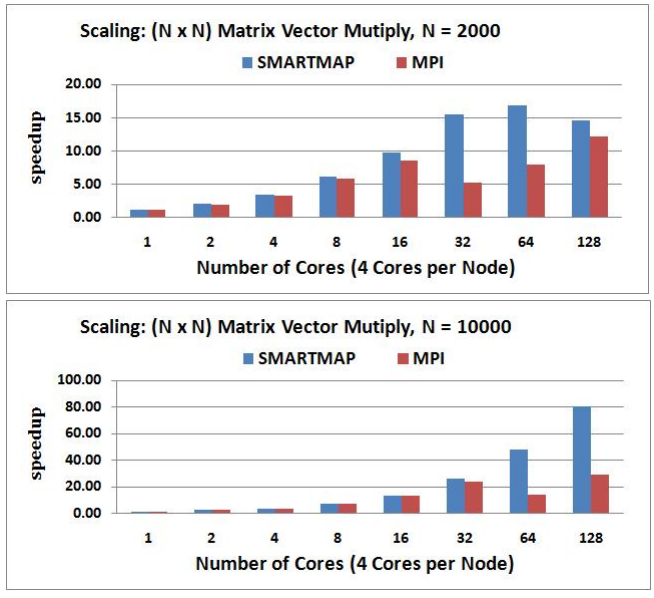


FIG. 4.5. *Scaling of different sizes of matrix vector multiply*

The comparison of the one level algorithm using MPI and the two-level algorithm using (MPI + SMARTMAP API) is shown in figures 4.4 and 4.5. We can see that the two level algorithm using (MPI + SMARTMAP API) receives a much better result compared to MPI

algorithm. And the advantage increases dramatically as the number of cores increases. The reason is that we regularize the all-to-all communication into three steps as shown above. And the NIC is not involved in the first and last step, which is realized by the SMARTMAP API. Considering that this all-to-all communication pattern is complicated and the number of messages communicated is large, our two-level algorithm significantly releases the burden of NIC and thus has a better result.

5. Concluding Remarks. We have presented an API for SMARTMAP and serveral applications, which are implemented using (MPI + SMARTMAP API), based one-level and two-level parallel algorithms. These implementations are compared for performance against the MPI implementations (based on one-level algorithms). For the dense matrix-vector multiply problem, which has a two-level algorithm, the implementation using (MPI + SMARTMAP API) is several times faster than the MPI version. However, in the CG Solver for a very sparse matrix, which also has a two-level algorithm, the implementation using (MPI + SMARTMAP API) does not have clear performance advantage, probably because there is not enough inter-core data traffic for the benefit of SMARTMAP to outweight the overhead of an extra synchronization needed in the parallel phase in which cores on the same node work together in computation and intra-node data exchange. For the Jacobi Solver based on a one-level algorithm, the implementation using (MPI + SMARTMAP API) only shows slight performance gain over the MPI implementation.

Overall, it seems that very significant application performance gain can be obtained if the application is implemented using a two-well algorithm to map to two levels (core-level and node-level) of parallelism of the architecture, and there is enough data exchange. Otherwise, the performance benefit is not significant. Since the experiments are done on a machine with a cluster of quad-core processors, it remains to be seen whether consistent significant performance gains can be achieved on a cluster of processors with many cores, say 16 and beyond. Another interesting area of research is to find general ways to design two-level parallel algorithms.

6. Acknowledgments. We would like to express our appreciation to Kevin Pedretti, who provided the Kitten operating system for us to test SMARTMAP functions. We are also grateful to Kurt Ferreira for their help on Catamount. Suzanne Kelly gave a lot of care to this research in many different ways. She also helped review this paper and caught an error in the Example in Section 3.2; and her constructive comments have greatly improved the presentation of this paper. Any errors that remain are solely the responsibility of the authors.

REFERENCES

[1] *ITRS, international technology roadmap for semiconducors (2007 edition)*, (URL:)http://www.itrs.net/Links/2007ITRS/ExecSum2007.pdf.

[2] R. Brightwell, *A prototyple implementation of MPI for SMARTMAP*, Lecture Notes in Computer Science, 5205 (2008), pp. 102–110.

[3] R. Brightwell, M. Heroux, A. Geist, and G. Fann, *Catamount n-way performance on xt5*, Workshop Paper, Cray Users Group Conference, (2009).

[4] R. Brightwell, K. Pedretti, and T. Hudson, *SMARTMAP: Operating system support for efficient data sharing among processes on a multi-core processor*, Conference on High Performance Networking and Computing, (2008).

[5] J. Demmel, *Applied numerical linear algebra [illustrated]*, SIAM, (1997).

[6] W. Gropp, *The 2-d poisson problem*, Sourcebook of parallel computing, (2003), pp. 469–480.

[7] M. Heroux, Z. Wen, J. Wu, and Y. Xu, *Initial experiences with the BEC parallel programming environment*, Parallel and Distributed Computing, 2008. ISPDC '08. International Symposium on, (2008), pp. 205–212.

[8] R. Rabenseifner, G. hager, and G. Jost, *Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes*, 2009 parallel, Distributed and network-based processing, (2009), pp. 427–436.

[9] J. Wu, Y. Xu, and L. Zhao, *Reformulating data exchange for scientific many-core cluster computing*, Printed.

ROOT CAUSE ANALYSIS OF ERRORS FOR HIGH PERFORMANCE COMPUTING

JOEL M. VAUGHAN*, JON R. STEARLEY†, SCOTT A. MITCHELL‡, AND GEORGE MICHAILIDIS§

Abstract. A supercomputer is a complex network of CPUs, switches, routers, and cables. When an error occurs, such as a job halting, a message being dropped, or a corrupted message arriving, the root cause of the error may be difficult to assess. When multiple errors occur over different jobs, users, executables, and subsets of the network, it may be possible to combine the data to gain insight into likely root causes. Currently, the process of locating the root cause of these faults is carried out by system administrators, who use their expertise with a particular system to comb through the logs and determine the most likely causes. However, as supercomputers grow in size and complexity, this process will become more costly, both in terms of resources spent to isolate the source of the faults, and the compute time lost as the failure is corrected. We present a statistical method to assist in determining the root cause of failures. The method is discussed, and real failures on a current production system are analyzed.

1. Introduction. Faults in supercomputers are expensive, both in terms of the efforts to locate and fix the problem, and the cost incurred to users not having the computer available for their efforts. Unfortunately, faults are common because of the large number of components, and due to dynamic complexity, it is not trivial to isolate the root cause of the fault. Possible causes include a variety of hardware components, including compute nodes (composed of CPUs, memory, and network interfaces), login nodes, I/O nodes, network components such as links and routers, and storage devices, including hard disks and their network interface. Hardware components are not the only possible causes, however. Applications, users, and interactions among hardware components (such as subtle timing interactions) and/or between hardware and users or applications are also possible causes of faults. Throughout this work, we will use the term *components* to refer to all possible root causes considered, whether they be hardware, user, or software version, etc. Furthermore, as supercomputers become more powerful, the number of hardware components increases, as does the complexity of the interactions among various components, making the problem even more difficult, and this research more timely.

In addition to the challenges introduced by the sheer number of components to consider, the available information is subject to significant uncertainty. Only some types of hardware components, such as compute nodes, report errors when they malfunction. (Although it should be noted that these components do not *always* report faults. For example, if a fault is severe enough, a compute node would not be functioning sufficiently to report the error.) Others relevant components, such as the cables connecting network routers, never report errors. Additionally, there does not exist a direct mechanism for reporting errors associated with non–hardware components (such as users) or with interaction of components. Furthermore, errors caused by interactions between components (and some other errors) are intermittent. Finally, it should be mentioned that the errors are reported in the format of unstructured text in various machine loges, making the collection of this information a non–trivial task.

The varying amount of information available results in varying degrees of difficulty of the problem of finding the cause of the faults. Table 1.1 describes varying levels of difficulty. In the table, it is clear that the easiest problem to solve (although still not trivial) is the case where the component responsible for the fault always reports the problem. The problem becomes more complex as the frequency of the error reporting decreases and as the location of the error reporting moves away from the responsible components. It should be noted that some types

*University of Michigan(while working at Sandia National Laboratories), rsnation@umich.edu
†Sandia National Laboratories, jrstear@sandia.gov
‡Sandia National Laboratories, samitch@sandia.gov
§University of Michigan, gmichail@umich.edu

of components, such as compute nodes, have the possibility of fitting into difficulty 1, since there are types of errors that they will (almost?) always report. However, other component types are necessarily more difficult, such as the network cables, for which there is no direct reporting mechanism, or "user" who typically do not report themselves as the source or cause of the errors in question.

Difficulty		**Given a problem, evidence is exhibited on:**		
(1=lowest)	$\mathbb{P}(d\|p)$	causing component:	affiliated component:	$\mathbb{P}(i\|p)$
1	1	always	never	0
2	0	never	always	1
3	< 1	sometimes	never	0
4	0	never	sometimes	< 1
5	< 1	sometimes	sometimes	< 1
beyond scope	0	never	never	0

TABLE 1.1

Taxonomy of Problem Difficulty. d indicates direct evidence, exhibited on the causing component, i indicates indirect evidence, exhibited on an affiliated component, and p represents a problem.

It is the goal of this work to investigate statistical methods to determine significant associations between error observations and components, for all five levels of difficulty described in Table 1.1. The problem is approached statistically from a multiple hypothesis framework, using the FDR correction of Benjamini and Hochberg [2]. The specific levels of difficulty from the table dictate the data structure and hypotheses which are tested.

The task of finding the root cause of faults on supercomputers is an important endeavor in itself. However, it should be noted that supercomputers are essentially complex networks, and that the algorithms being explored have direct application to other event–cause analysis problems on complex networks. For example, origin determination of security events on large networks of personal computers is another possible application.

The rest of this discussion is organized as follows. Section 2 discusses the approach used to solve the problem. Section 3 discusses some preliminary results. Section 4 discusses challenges in investigating this problem, including some that have been overcome and some that still remain. Finally, section 5 discusses the future directions and applications of this work.

2. Approach. Initially, three approaches were considered in solving this problem: a General Linear Model (GLM) approach (see e.g. [1]), an empirical Bayes method proposed by DuMouchel [4] , and a multiple hypothesis testing framework via a False Discovery Rate (FDR) correction attributed to Benjamini and Hochberg [2]. While in in the process of obtaining the data from the supercomputer analyzed in this document, simulations were used to explore the merits of these three approaches. Each approach had weaknesses. The GLM approach suffered due to many contingency table entries of 0. The numerical optimization required by DuMouchel's method were unstable with this type of data. The FDR approach does not directly address the issue of interactions between components. The FDR approach was selected because of advantages in statistical rigor, success in initial simulations, and the ability to be easily generalized to increasingly complicated situations. Below, we briefly describe the FDR procedure in general and describe how it is implemented in the context of this problem.

2.1. Multiple Hypothesis Testing for Fault Problem. First we briefly review some terminology from statistical hypothesis testing, then describe a multiple hypothesis testing framework for the fault–detection problem. In statistical hypothesis testing, one typically

has two models for the data: a null model (H_0) and an alternative model (H_1). The goal of hypothesis testing is to establish whether the data contains convincing evidence in favor of the alternative model. This is typically done by calculating the probability of obtaining the data observed (or data more in favor of the alternative model) while assuming the null model to be true. If this probability is sufficiently small, one rejects the null model in favor of the alternative, and indicates that there is sufficient evidence in favor of the alternative model. Since the truth of the situation is not known, it is possible to make errors. A Type I error, of false positive, occurs when the null model is actually true, but it is rejected in favor of the alternative. A Type II error occurs when the null model is not rejected, but the alternative is true. Statistical power is thought of as the ability to reject the null model when the alternative model is true. (Technically, Power = $\mathbb{P}$(Reject $H_0|H_1$)) A typical strategy in designing a test is to attempt to maximize statistical power while controlling the probability of a Type I error at or below some pre–defined level.

The fault problem naturally fits into the hypothesis testing framework. However, because we have many components, we wish to conduct multiple tests simultaneously. For example, consider a set of I components, $C_i, i \in 1, \ldots, I$. Over the period of time in question, each component C_i experiences a vector of error measurements $\vec{X_i} = (X_{i,1}, \ldots, X_{i,N_i})'$, where N_i, the length of each vector, may differ for different components. (The precise definitions of these observations will be made in Section 2.3.) Defining

$$\bar{X_i} := \frac{1}{N_i} \sum_{j=1}^{N_i} X_{i,j},$$

the usual sample mean, it would be natural to use this sample mean to test hypotheses of the form

$$H_0 : \mu_i \leq a \text{ vs } H_1 : \mu_i > a, \quad \forall i \tag{2.1}$$

where a is some pre–determined threshold, and μ_i represents the population mean number of errors for component $i \in 1, \ldots, I$. Intuitively, we are testing to see whether each component generated significantly more errors than could be attributed to it by random chance. However, in applying this framework to the fault–cause problem it is important to consider both the issue of lack of power due to the large number of tests involved and the fact that components are dependent on one another. The next paragraphs discuss these issues in greater detail, and the next section discusses their resolution via an FDR correcting approach.

The multiple testing problem is well studied in the statistical literature. Traditionally, the overall procedure is designed to control the overall chance of a Type I error, also known as a false positive. That is, the rules to reject the null hypotheses are designed such that the overall probability of incorrectly rejecting even a single null hypothesis is less than or equal to some pre–defined level. This is also referred to as controlling the family–wise error rate (FWE). It is well known in the statistical literature that in order to maintain such an overall level, the statistical power of the test decreases with the number of tests. If there is a sufficiently large number of tests being considered, one needs overwhelming evidence to find a significant result. In practice, the tests become so hard to reject that no progress can be made in identifying significant results. It should be noted that in the context of the fault–cause problem, it would be better to determine a larger set of possible root causes for experts to investigate, with a small subset being incorrect, than an empty list that provides no direction at all.

Another relevant feature of the multiple hypotheses considered in the case of the fault–cause problem is that various hypotheses are dependent. User 1 and User 2 might have used

the same host in the time period being considered, and hence are both dependent on this host. Thus, the individual hypotheses tests are not independent, and any method used should incorporate this feature. Indeed, both of these issues are addressed by using the FDR correction rather than the tradition family–wise error rate approach, as described in the next section.

2.2. Multiple Hypothesis Testing with FDR Correction. The fundamental ideas of the FDR was first discussed by Benjamini and Hochberg in their 1995 paper: [2]. Since then, it has gained wide use in situations where large numbers of hypotheses are tested, such as experiments involving gene micros. The basic ideas of the procedure are described below:

Rather than controlling the FWE like more traditional methods, the new method controls the false-discover rate (FDR), which is the expected proportion of erroneous rejections among all rejections. In addition, the authors show that if all hypotheses are true, controlling the FDR is equivalent to controlling the FWE, but is more powerful. This is because when many null hypotheses are false, controlling the FDR to reject more null hypotheses, although it is more likely that some of these hypotheses are actually true. It remains for the user to investigate the rejected hypotheses. Using the FDR rather than the FWE is more useful in situations where one would want to control the proportion of false rejections rather than the overall chance of making at least one mistake. (This is our operating assumption for the fault–cause problem.)

The FDR procedure is now described briefly. To fix notation, assume that there are m hypotheses, and that m_0 of them are true null hypotheses (the number and identity are unknown). Furthermore, let q be the user–determined false discovery rate. The procedure has the following steps:

1. Conduct each hypothesis test independently, and record the observed p–values, denoted $p_i, i \in 1, \ldots, m$.
2. Rank the observed p–values: $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$
3. Define

$$k = \max \left\{ i : p_{(i)} \leq \frac{i}{m} q \right\}, \tag{2.2}$$

4. Reject $H^0_{(1)}, \cdots, H^0_{(k)}$. If no such i exists, reject no hypotheses.

This process was shown to control the FDR at level $q\frac{m_0}{m} \leq q$ by Benjamini and Hochberg in the case of independent test statistics [2].

In an important follow–up paper, Benjamini and Yekutieli [3] illustrate that the above procedure still controls the FDR in situations where the individual hypotheses are positively correlated. Specifically, the procedure controls the FDR in families of positively correlated test statistics. Examples of such families include multivariate normal test statistics with $\Sigma_{ij} \geq 0$ on the set of true null hypotheses, latent variable models with positive dependency structures, absolute values of multivariate normal and t statistics, and studentized multivariate normal distributions. Also, Benjamini and Yekutieli show that a modified version of the procedure controls the FDR regardless of the type of dependency, but at a loss of statistical power.

Thus, the FDR procedure adequately accounts for the previously mentioned features of multiple–hypothesis testing in the fault–cause problem.

2.3. Application of FDR procedure to Supercomputer Fault Problem. The FDR framework can be applied to the fault cause problem in different ways to examine different types of components under consideration. In particular, we make a distinction between situations where the components being examined either self–report, or do not self–report. The difference in the way the technique is applied primarily resides in the manner that the observations are defined. Currently, we are can test each type of component from a given

set of data, but we control the FDR for each component independently, rather than the overall FDR. Also, some normalization strategies are discussed below. Determining other useful normalization strategies is an important part of our ongoing research.

If components self–report: In this case, the situation is relatively straight forward. As observations, we use error counts per job as the observations. That is, each component's test vector has a length of N_i, where N_i is the number of jobs that involved component i. Independent tests are then carried out on these vectors, and are corrected via the FDR procedure.

Alternatively, rather than each observation being the count of errors in that job, one could do the error count divided by the duration of the job. The tests are then carried out on these vectors. This approach takes into account the intuition that if components are used more frequently, one would expect a greater number of random errors.

If components do not self–report: The situation in this case is generally more difficult. This corresponds to difficulties 2,4, and 5 from Table 1.1. First, we define some notation. Let $\mathcal{H}$ denote the set of components in question and $\mathcal{J}$ denote the set of jobs in the window of time studied. Further, let μ_h represent the overall average error messages per node hour of jobs associated with component h.

The algorithm used is as follows:

1. For each job $j \in \mathcal{J}$, define the statistic

$$z_j := \frac{\text{Total Error Messages in Job } j}{\text{Number of Hosts in Job } j \times \text{Time (in Hours) of Job } j}$$

   This is essentially observed errors per node hour for each job, or observed error rate per job.
2. For each component $h \in \mathcal{H}$, collect the z_j for those jobs which utilized component h into a vector denoted $\vec{z}^{(h)}$.
3. For each component h, conduct an independent test of the form $H_0 : \mu_h \leq a$ vs $H_1 : \mu_h > a$ for a threshold a. We use as data the vector $\vec{z}^{(h)}$, the collection of z_j such that component h is used in job j. (Note: One threshold explored in practice is $a = \bar{z} = \frac{1}{n_j}\sum_{\mathcal{J}} z_j$, where n_j is the number of jobs. This represents the overall average observed error messages per node hour.)
4. Continue the FDR procedure from section 2.2 on the resulting p-values.

This method can be used for types of components that do not report error messages directly, whether they are hardware components, software components, or users.

3. Results. The initial analysis involved analyzing the logs of Glory, an unclassified capacity system. Glory consists of 272 compute nodes. Each compute node is made up of four sockets, and each socket contains a quad–core processor, for a total of 4,352 processor cores. These compute nodes are networked together in a redundant tree topology. (Note: The results of Gibson et al [5] indicate that supercomputer failures generally increase as the number of sockets increase. The more than 1,0000 sockets in Glory are a sufficient first study, but a system with more sockets will be necessary to study. See section 5 for more details.) We examined two types of components: *users* and *compute nodes* (referred to as "hosts" in the following). For each component, we considered several types of errors. The errors considered for this preliminarily investigation were primarily located in either the syslog or the SLURMctld logs. Table 3.1 shows the error strings considered, the log that the error was found in, and the number of such errors over the time period investigated.

3.1. Example 1: Possible Indication of User–Caused Fault. In this section, we examine 43 "/ram/tmp" messages from the SLURMctld logs. The analysis (shown in Table 3.2) suggests that this particular error is significantly associated with particular users, rather

Error String:	Source:	Count:
ECC	syslog	16643
MCE	syslog	8833
MCE NOT (log statistics)	syslog	1876
oom	syslog	205
/ram/tmp	SLURMctld	43
reason ECC	SLURMctld	3
reason stuck	SLURMctld	0
reason not responding	SLURMctld	0
reason EDAC	SLURMctld	0
reason needs new power supply	SLURMctld	0
reason replace power supply	SLURMctld	0
reason found not ACTIVE HCA	SLURMctld	0
reason needs reboot	SLURMctld	0
reason failed to allocate	SLURMctld	0
reason Out of memory	SLURMctld	0

TABLE 3.1
Error strings considered in the initial investigation.

than particular hosts. Thus, even though the error is of a memory type, it is more closely related with a particular user rather than a set of nodes. This would suggest that the system administrators should work with these users or examine their user logs to try to resolve these problems, rather than replace hardware.

3.2. Example 2: Possible Independent User and Host Cause. In this example, we examine 16,643 ECC messages from syslog. This is an interesting situation where the analysis flags both a user and a host as significant. The results shown in Table 3.3 indicate that Host 237 produces a significantly high number of this type of error. On the other hand, we also see that User 0 tends to be in jobs which experience significantly high numbers of these very same errors. However, ECC errors are primarily hardware type errors. The possibility existed that User 0 simply used Host 237 when the host was experiencing errors. To further investigate this possibility, we examined the jobs shared by this user and host over the time period in question. These results are shown in Figure 3.1. Although User 0 and Host 237 did have 4 jobs in common and did experience errors on each of these jobs, these jobs are certainly not the largest sources of errors for either of these components. To further the examine the interactions, we removed these four jobs from the analysis, and both Host 237 and User 0 remained significant.

4. Challenges. In pursuing a full solution to this problem, several challenges have been encountered. Below, we detail challenges which we have overcome and challenges that will need to be addressed as we continue working on this important problem.

Obtaining and Processing Data: The records that we need to consider are found primarily in various types of logs generated on the systems. These logs contain copious amounts of largely unstructured text, while our methods required that error messages be aggregated by job, with the relevant information for each job recorded. (See [6] for a discussion of some of the difficulties in working with system logs.) After exploring several options, we ended up using a product called Splunk [7] to perform this function. Splunk is a software tool designed to index and search various types of logs encountered by information technologies professionals including system logs, network logs, configuration files, and database tables. It

Counts:

Host ID	Count	Jobs	p-val
143	2	146	0.079
36	2	50	0.08
218	1	258	0.16
277	1	231	0.16
267	1	227	0.16
286	1	194	0.16
278	1	162	0.16
287	1	152	0.16
201	1	142	0.16
284	1	135	0.16

Normalized Counts:

Host ID	Sum	Jobs	p-val
143	11	146	0.12
36	5.5	50	0.14
218	41	258	0.16
277	13	231	0.16
267	6.8	227	0.16
286	2	194	0.16
278	40	162	0.16
287	7	152	0.16
201	9.1	142	0.16
284	41	135	0.16

User ID	Count	Jobs	p-val
15	24	157	0.00022
24	12	109	0.00055
19	3	40	0.042
88	2	261	0.079
22	1	224	0.16
79	1	15	0.17
0	0	36	0.5
1	0	72	0.5
2	0	352	0.5
3	0	121	0.5

User ID	Count	Jobs	p-val
15	195	157	0.00024
24	377	109	0.033
88	3240	261	0.08
19	2.2	40	0.099
22	0.083	224	0.16
79	3	15	0.17
0	0	36	0.5
1	0	72	0.5
2	0	352	0.5
3	0	121	0.5

TABLE 3.2

Example of User Error: Results of 43 /ram/tmp messages from SLURMctld logs. Yellow highlighting indicates that the component in question was determined to be significant by the given test.

has the capability to monitor and index logs, and then search the logs based on automatically generated key words. Although primarily designed to be used through a web interface, it is also possible to interface with the program via its REST API. Using this interface, we were able to conduct repeated searches, first to identify a record of each job, and then to search for specific types of errors within a job based on that job's host list, start, and end times.

It should be noted that although Splunk is able to aggregate information from various (largely unstructured) sources, the process does take time. This time will undoubtedly increase with larger systems. Furthermore, it is not practical to be used in on–line detection schemes, so another solution would be needed for an active monitoring implementation of the techniques discussed here.

Eliciting Sufficient Knowledge from Experts: Even after obtaining the system logs and the technology to search the logs, it was still necessary to determine the appropriate errors to investigate. For this, we need the knowledge of experts, in this case the system administrators of the systems in question. Initially, we needed to elicit from them which error messages are both common and relevant to the problem. Later in the project, it was necessary to communicate the results, and ask for their help in evaluating the results. These communications are often difficult for a variety of reasons. Because we do not possess intimate knowledge of the system operations, it is difficult to form questions that lead to the information we need to know to evaluate our methods. Additionally, the system administrators are (rightly) focused on the day–to–day operation of the system, and do not have the time to respond to all our queries and analyses when they come up. Additionally, even when such knowledge is ob-

Counts:

Host ID	Count	Jobs	p-val
237	83	94	0.00019
259	190	99	0.0016
209	118	121	0.0099
228	658	76	0.034
183	3	53	0.042
45	295	16	0.049
34	9	48	0.053
246	2	60	0.08
207	8	52	0.086
82	91	76	0.087

Normalized Counts:

Host ID	Sum	Jobs	p-val
259	49	99	0.0024
46	149872	16	0.063
273	317	100	0.066
82	3.6	76	0.076
228	655	76	0.078
45	11034	16	0.084
216	48	149	0.092
209	913	121	0.11
207	19137	52	0.15
34	23	48	0.16

User ID	Count	Jobs	p-val
0	66	36	0.00077
80	27	27	0.011
72	49	174	0.033
14	5	324	0.048
78	21	261	0.05
73	22	94	0.065
7	403	60	0.069
23	12	187	0.07
1	66	351	0.075
31	2	344	0.079

User ID	Count	Jobs	p-val
0	4.2	36	0.0013
80	1.5	27	0.02
72	9.9	174	0.034
78	27220	261	0.065
14	131	324	0.067
35	4.1	6	0.072
30	0.46	19	0.087
1	1.2	351	0.087
31	32	344	0.098
73	217	94	0.11

TABLE 3.3

Example of Hardware and User Error: Results of 16,643 ECC messages from syslog. Yellow highlighting indicates that the component in question was determined to be significant by the given test.

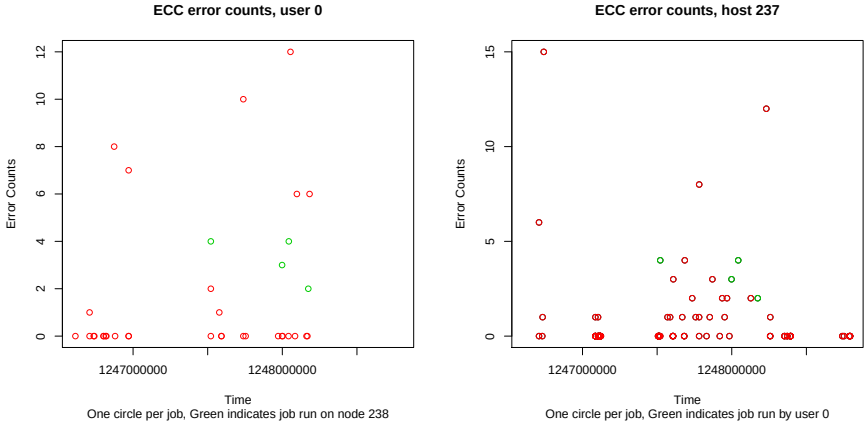
tained, the challenge of quantifying this knowledge for later use remains an open challenge.

Determining Appropriate Threshold: One of the difficulties of the method is that the methods described in section 2 require a threshold value to test against. Determining an appropriate value is important, but not trivial. Changing the test value can greatly impact the results in some of the data sets observed, and it is important to tune the procedure while incorporating feedback from those who understand the system best (i.e., the system administrators).

Presenting Results: It is important to present the results of these methods in an informative way. Deciding the appropriate amount of information to present and the best format has been a challenge throughout the project. Currently, the analyses are carried out by an R script which produces a pdf of ranked tables, similar to those in Tables 3.2 and 3.3. However, it is desirable to display more information, perhaps on the actually computer graph. This leads to the issue of visualization scalability of these graphs, as it is difficult to visually inspect graphs with thousands of nodes.

5. Future Work. There is still much work to do on this problem. First, we have only considered two types of components. It is important to include other types of components, particularly:

- Application
- Network Components, including routers and cables
- I/0 Hardware
- Software version



(a) User View (b) Host View

FIG. 3.1. *Time view of the ECC errors experienced by user 0 and those experienced by host 237.*

In addition to these system components, meetings with system administrators have revealed other factors to consider. For example, are the frequencies of certain types of failures correlated with whether the sun is out or not (as one might suspect due to differences in the frequency of cosmic rays). As another example, are the errors produced by a compute node related to its position in the rack (due to heat flow issues)? We also hope to investigate the relationship of errors with the structure and dependencies imposed on components throughout the network topology and routing, as well as the effects of hardware swapping as a solution to faults. We are interested in pursuing these questions, and others that might be suggested by the system administrators in the future.

We have been working with error counts, and treating all errors equally. However, certain errors may be more informative than others. As the work continues, we hope to explore using this information in the analysis.

Finally, we thus far only examined a single system. In order to further investigate the usefulness of our methods, we hope to analyze the logs from other systems. In particular, we plan to investigate errors on a system that has a 3D grid topology, rather than the tree topology found in Glory. Methods such as ours are even more necessary for a 3D grid topology, since the systems are more complex and the components are more interdependent. Additionally, based on the work of Gibson et al [5] and the desire to test the relevance of this work for future computer systems, a system with a greater number of processors should be studied.

As we continue this effort, it is important to remember that we are hoping to develop a methodology that is practical to implement, and will substantially aid in the resolution of the cause of faults in these systems. Furthermore, we hope to extend the methodologies developed to more general event–cause detection problems on other types of networks.

6. Acknowledgments. The authors would like to thank the Glory system administrators for their help in obtaining and interpreting the data. Thanks to Randall Laviollette, Vitus Leung, and Sue Kelly for their helpful discussions. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

REFERENCES

[1] A. Agresti, *Categorical Data Analysis*, John Wiley and Sons, 2002.
[2] Y. Benjamini and Y. Hochberg, *Controlling the false discovery rate: a practical and powerful approach to multiple testing*, Journal of the Royal Statistical Society, Series B., 57 (1995), pp. 289–300.
[3] Y. Benjamini and D. Yekutieli, *The control of the false discovery rate in multiple testing under dependency*, The Annals of Statistics, 29 (2001), pp. 1165–1188.
[4] W. DuMouchel, *Bayesian data mining in large frequency tables, with an application to the fda spontaneous reporting system*, The American Statistician, 53 (1999), pp. 177–190.
[5] G. Gibson, B. Schroeder, and J. Digney, *Failure tolerance in petascale computers*, CTWatch Quarterly, 3 (2007).
[6] A. J. Oliner, A. Aiken, and J. Stearley, *Alert detection in system logs*, in ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Washington, DC, USA, 2008, IEEE Computer Society, pp. 959–964.
[7] Splunk, *http://www.splunk.com/product*. World Wide Web, 2009. Retrieved August 2009.

USING BLOCK RAM TO ACCELERATE MATRIX-VECTOR PRODUCT CALCULATIONS IN FPGAS

DEREK WOODMAN*, DWIGHT DAY†, AND DOUGLAS DOERFLER‡

Abstract. A major bottleneck in scientific computing today involves actually getting the data to the processor. Memory is clocked at a slower speed than the processor, and the processor usually has to do address calculations before it can ask for data. All of this together leads to the processor spending more time waiting on the data than actually working on meaningful calculations. In this paper, we focus mainly on calculations done on sparse matrices since they are very memory intensive. First, we explain how typical DDR2 SDRAM works. Then, we explain why this doesnt work well for sparse matrix calculations. Finally, we present a solution to the problem, which involves using an FPGA and the on board BRAM.

1. Introduction. As described by Moores Law, the number of transistors on a chip doubles about every two years. Processing power has consistently risen each year because of the increased number of transistors on each chip [2]. However, memory chips have not increased with the same leaps and bounds as the microprocessor. Because of this, the performance impact of memory accesses has consistently increased. Techniques like prefetching work great when the software execution is predictable, but this isnt always the case. For example, in sparse matrix operations, mainstream memory controllers are not designed for the non-unit stride memory operations. The data needed from operation to operation is generally spaced far apart in memory and takes longer to access than if the requested data was continuous. The following section explains how SDRAM works.

2. Previous Work. Previous work at Sandia has influenced this research. The preceding research was done on accelerating the pHPCCG benchmark in hardware on an FPGA [1]. The first step involved designing a 32 by 32 double-precision floating-point dot product in hardware. Performance of the dot product was increased from 6 MFlops using a FPU to 20 MFlops using the FPGA accelerator. To truly test the performance, the pHPCCG benchmark was implemented. This involved memory that was not packed. Since the accelerator relied on SDRAM, a situation arose that involved waiting on memory for often than actually computing. In turn, this research was created to solve that problem.

3. SDRAM. Most types of RAM store data in a row/column manner, and SDRAM is no exception. While the process for reading and writing is well defined, you cannot just write an address and expect data to be immediately available. There are certain latencies that must be met. Most RAM is row orientated, meaning consecutive data is on the same row. To read/write data, one must activate the wanted row, wait a certain amount of latency cycles, select the wanted column, wait a certain amount of latency cycles, and then the data can finally be read or written. Anytime a new row is needed, the current row must be precharged to deactivate it, which also has a few clock cycles of latency associated, and the whole process must be repeated. Figure 3.1 on the next page, graphically shows this (without precharging).

3.1. Sparse Matrices. Sparse Matrices are used extensively in scientific computing. For example, when modeling the motion of a material, one can look at the influence of each particle on its surrounding particles. In a material of any size greater than the particle itself, there are going to be a large number of particles it doesnt influence. In addition, the material is generally 3D, while matrices are 2D. So while the particles are next to one another in 3D, they wont necessarily all be together in the 2D matrix. This means for each row of the sparse

*Arizona State University, Derek.Woodman@asu.edu,
†Kansas State University, day@ksu.edu,
‡Sandia National Laboratory, dwdoerf@sandia.gov

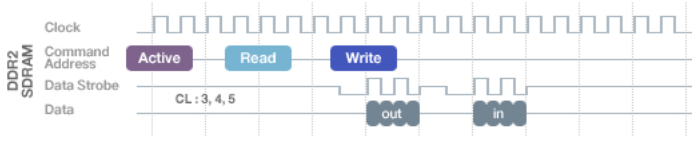


FIG. 3.1. *The process of reading and writing SDRAM. The Active block selects the row while the Read and Write blocks select the column. As shown, this takes several clock cycles. Adapted from http://www.samsung.com/global/business/semiconductor/products/dram/Products_SDRAM.html*

matrix, there are a few nonzero values spread throughout a large number of zeros. Because of all the zeros in the sparse matrix, it is not stored in its true form. Instead it is broken into two matrices–one to hold all the non-zero numbers, and another to store the original position of each non-zero number. This takes up far less space in memory and stores the data continuously. Figure 3.2 below, shows how the conversion is accomplished.

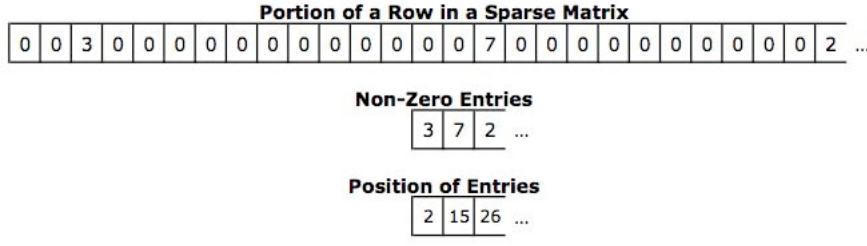


FIG. 3.2. *A portion of a row of a sparse matrix and how it is stored in memory with two other matrices. In a real sparse matrix with 27 non-zero entries and over 15,000 x 15,000 total entries, a lot of memory will be saved.*

This way of condensing the sparse matrix saves a considerable amount of space, but storing it is only half the battle. It must be used, and therein lies the problem. Since the Non-Zero Entries Matrix (NZEM) and Position of Entries Matrix (POEM) will be accessed continuously, getting the data from RAM doesnt have a large associated latency issue. However, dot products are often performed on sparse matrices. This involves using the data in the POEM to access another vector to get the desired data for multiplications. The problem here is the position of the data in the sparse matrix jumps around. For example, in a particular row it may go from location 200 to 10215. And these large jumps happen several times during each dot product. Since the data isnt close by in memory, the row being accessed is going to change for every data access.

This means RAM must be precharged every time, adding a lot of latency. For each dot product, this happens around 27 times (average number of nonzero values in a row), but a dot product will have to occur for the tens of thousands of rows in the sparse matrix, so the latency associated with memory access becomes a limiting factor. The portion of code on the following page shows the double memory access to the multiplying vector (MV):

```
for (i=0; i<nonZeroLength; i++){
   sum += NZEM[i] * MV[POEM[i]];
}
```

3.2. Solution. To solve this problem, we developed custom hardware using the Xilinx Virtex-5 FPGA on the ML510 development board [4]. It has a PowerPC embedded into the fabric of the FPGA. This platform was handy because it allowed us to run standard C code on the PowerPC, and offload data to our hardware design in the FPGA to try to achieve accelera-

tion. As it turns out, the problem is real, and a dramatic speed up can be realized. The solution to the increased latency due to the large address space jump in accessing the multiplying vector was to store the data on the FPGA itself. The Block RAM (BRAM) on the FPGA is able to read and write any address in one clock cycle, no matter what address it had accessed previously. This is much better than the latency associated with SDRAM. The amount of BRAM is also immense. It was easily able to store 15,625 double-precision floating-point numbers (64-bits each). This number was chosen purely to test the sparse matrix test bench. Figure 3.3 below, uses Chipscope to show that the data is available immediately after giving the BRAM a new address [3].

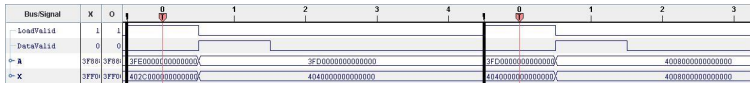


FIG. 3.3. *A Chipscope output during a request to read BRAM. Chipscope samples during the middle of each clock cycle and is triggered when LoadVaild goes high. The LoadValid signal being asserted signals the BRAM to read data from the address provided on the incoming bus. The A data was already provided on the incoming bus as the data from the sparse matrix. The DataValid signal indicates that the data for X has been read from BRAM. As shown, this process only takes one clock cycle.*

With this design, data can be accessed in 10 ns, with the constraint being the clock controlling the BRAM. The fabric of our design was only running at 100 MHz but could be increased, which would in turn decrease BRAM access time. Although there are a wide array of SDRAM chips on the market, the following statistics are for the chip on our board. It was a DDR2 chip running at 667 MHz with a 5-5-5 latency (activate row- select column-precharge). SDRAM is also usually pipelined, so a couple extra clock cycles are required for the data to actual appear on the bus. Therefore, our chip took about 51 ns to access data on different rows. Achieving 1/5 the latency is a substantial speed up in an operation that would read data as often as a sparse matrix dot product.

Although no data analysis was done on an actual sparse matrix, the design was checked to make sure it could perform a sparse matrix-like dot product. Sixty-four-bit floating-point values were chosen so their products equal values from one to twenty- seven. The dot product then requires the summation of these values. As shown in Figure 3.4 below, the summation of 1-27 equals 378 (0x4077a00000000000).

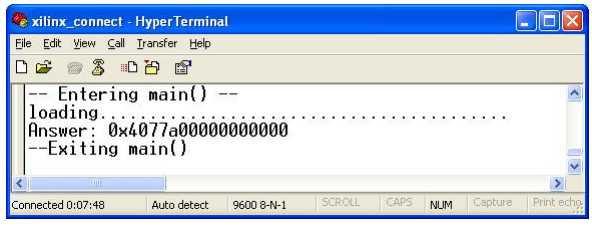


FIG. 3.4. *Output received through HyperTerminal from the FPGA. Answer is the result of the dot product performed.*

4. Conclusion. In sparse matrix multiplication, a high amount of latency is involved in retrieving data from the multiplying vector. This is because the data is spread out, and SDRAM must precharge, active the row, and select the column every time. All this latency can truly become a huge bottleneck when this operation is performed millions of times. Our proposed solution of storing data in BRAM in the fabric of the FPGA for immediate access cut the latency associated with this operation by one fifth. It also completely eliminates any

latency associated in our design in the eyes of the PowerPC. It can move along and do other calculations until the FPGA returns the dot product result. We suggest further research in this area. Eliminating other forms of latency due to memory could further speed up the sparse matrix operation. Currently the PowerPC has to send data to our hardware on the FPGA. One possible performance increase could involve having the memory interface integrated directly with the hardware. This way the data from memory wouldnt have to route through the PowerPC. Our improvement alone could save valuable time, but many other optimizations are also possible.

REFERENCES

[1] D. Day, *Developing an advanced architecture research capability using FPGAs*, Sandia National Laboratory CSRI Workshop Presentation. June 17, 2009.

[2] Intel, *Moores law: Made real by intel innovation*, http://www.intel.com/technology/mooreslaw/.

[3] Xilinx, *ChipScope Pro 10*, http://www.xilinx.com/support/documentation/dt_chipscopepro_chipscope10-1.htm.

[4] ———, *Xilinx: Products: Development boards: Xilinx ml510 reference designs*, http://www.xilinx.com/products/boards/ml510/reference_designs.htm.

PSST: A MODULAR PROCESSOR SIMULATOR FOR THE STRUCTURAL SIMULATION TOOLKIT

CHAD D. KERSEY* AND ARUN RODRIGUES†

Abstract. The current landscape of computer architecture simulators is crowded with a multitude of simulators, each written for a niche purpose. The Structural Simulation Toolkit provides a unified framework for cycle accurate component level simulations of computer architectures. The motivation for PSST, a Processor for SST, is to provide a universal CPU microarchitecture simulator based on an interchangeable instruction set level VM and a collection of low-level modules each simulating its own orthogonal aspect of the processor's physical behavior. By separating different aspects of simulation into modules, PSST provides for the selection of only those aspects that are pertinent to the simulation being performed. The separation between instruction set architecture and physical models enables evaluation of new processor implementations with existing software.

1. Introduction. The Structural Simulation Toolkit, SST [4], is a modular framework for the distributed simulation of high performance computer architectures. Within SST, components of a computer system are mapped to *components* of the SST sense, linked together through *links*, which may represent, for example, network links between processors and routers. SST uses Sandia's own Zoltan [2] library to partition the simulated components for scalable distributed execution.

Currently, within SST, there is a PowerPC-based processor frontend, which runs applications in user mode. Timing is provided by a backend, which simulates a simple out-of-order processor pipeline. Because it is tied heavily to the PowerPC architecture, does not provide realistic simulation of the operating system, and does not allow fine-grained selection of a point along the simulation speed/accuracy axis, this component's utility is limited.

PSST was created to address these faults. By separating the VM from models of physical characteristics of hardware, PSST avoids relying on any one instruction set. The first and currently only VM module to be supported, based on a modified QEMU [1], provides support for x86-64. QEMU itself supports many more architectures, including i386, PowerPC, ARM, and MIPS, and VMs supporting these may become available as modules for PSST as well. Time in the VM advances in a piecewise linear fashion, based on an execution rate in instructions per cycle provided by the timing model; a method borrowed from Hewlett Packard's COTSon [3].

2. Modular Architecture. Processor simulators are often characterized as being either "functionally accurate" emulators or "cycle accurate" simulators. This nomenclature is ambiguous, especially when components are designed to operate in parallel, because timing and function are inseparably intertwined in parallel computing systems. If the user of a simulator wishes to obtain insight into the behavior of a timing-dependent computer system, the terms "ISA-level" and "physical" or simply "high-level" and "low-level" circumvent this ambiguity. PSST divides the processor into a high-level simulator, or *VM*, and low-level simulation components, or *models*. The VM provides variable-speed execution of an instruction stream, about which it emits information of three distinct types. *Instruction* events contain information about each instruction committed. *Memory operation* events describe loads and stores, other than instruction fetches. *Magic instruction* events can be used to provide services through PSST that would not otherwise be available, such as faking hardware accesses or inventing new instructions. Magic instructions, in x86, are simply `cpuid` instructions launched with invalid values of `%eax`. These are treated as `nop`s on real hardware.

*Georgia Institute of Technology, cdkersey@gatech.edu
†Sandia National Laboratories, afrodri@sandia.gov

Using these types of events, models can determine a rate of execution and current simulation time, and then provide that back to the VM. The following interface is provided to the models to allow these types of transactions. It is separate from the interface provided to the VM module itself, and is managed by PSST:

```
void callbackRequest(CallbackHandler* m, uint64_t cycles);
void setIpc(double ipc);

uint64_t getCycle(int vm_idx);

uint8_t memRead(int vm_idx, uint64_t addr);
void memWrite(int vm_idx, uint64_t addr, uint8_t data);
uint64_t memSize(int vm_idx);
```

It should be noted that the `vm_idx` parameter is included to enable future support of multiple VMs per PSST instance, and is currently assumed to be 0. `setIpc()` is the primary interface for timing adjustment. It is used to set the execution rate of the next instruction block based on the computed performance of the previous block. This creates a lag between events that affect performance and the actual result in simulation time, that can only be mitigated by adjusting the IPC more often when the simulation calls for it.

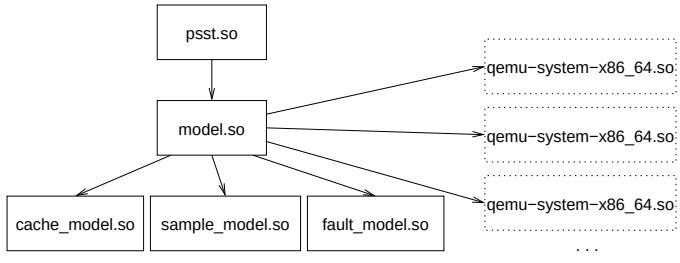


FIG. 2.1. *Dynamically loaded object hierarchy in PSST. Arrows can be read as "loaded by." QEMU objects are copied to new files in* `/tmp` *before loading.*

The functionality of PSST is spread across several dynamically loaded modules, shown in Figure 2.1. Using the `dlopen()` interface provided by the OS, these modules are loaded by PSST, and PSST itself is loaded by SST. The dynamically loaded module format was a natural choice for the models, providing a convenient way for models, open or closed-source, to be developed and distributed. Since the VM module is based on an existing application, the choice to use a dynamic module format for it was motivated by a desire to run the VM without importing all of its global variables into PSST's namespace. Because the dynamic loader will not reload the same library more than once, the VM objects are physically copied to temporary files prior to loading, so that each instance of the VM in a simulation can have its own global namespace.

`model.so` contains the interface between the VMs and models entirely, so that the PSST component itself is primarily responsible for instantiating a context within `model.so`, loading the VM, loading models, and telling `model.so` when the VM should advance. The interface between `model.so` and the PSST component is narrow but expressive, consisting of a set of functions to load VMs, load models, advance VMs to the current clock cycle, and tell `model.so` which PSST instance is currently running.

3. The VM Module.

3.1. Virtual Machine API. The VM module must implement the following procedures and variable:

```
int main(int argc, char** argv);
int run_vm(int instructions, double ipc, uint64_t start_cycle);
uint8_t memRead(uint64_t addr);
void memWrite(uint64_t addr, uint8_t data);
uint64_t memSize();

long clock_freq;
```

`main()` must provide initialization based on command line arguments, but unlike the `main()` procedure of a typical application, the VM's `main()` must exit. The VM is only advanced through use of the `run_vm()` procedure, through which the current cycle, the number of dynamic instructions for the VM to run, and the current execution rate in instructions per cycle. Translation of instruction count to wall clock time is done using the `clock_freq` variable. The ability for models to modify memory locations was initially added to support fault modeling for memory errors. Other potential applications of these functions include the use of models to emulate memory mapped hardware.

Although SST, and all PSST models, are implemented in C++, care was taken to make the VM interface compatible with C. This is because there are many VMs, including QEMU, implemented entirely in C and incompatible with C++ to the point that even getting the application to compile with a C++ compiler would require a major refactoring effort. For this reason, the VM module can be implemented entirely in C, and the interface to the VM requires no C++ features.

3.2. The QEMU VM Module. QEMU [1] is a full system emulator designed with portability in mind, originally written by Fabrice Bellard. It was chosen as the basis for the initial VM module because of its wide portability and support for multiple processors, including PowerPC, ARM, and x86. Central to QEMU is a dynamic recompiler with a two-step translation process, first translating all guest instructions to an intermediate form and then to host instructions. Because of this retargetable design, QEMU only requires $O(n + m)$ code modules to be written to support n guests on m hosts, instead of the $O(nm)$ required by traditional recompilers, a novel approach at the time [1] was written.

Since QEMU was a standalone program, a substantial bit of modification was required to adapt it to use as a VM module. Instrumentation functions, and calls to them, were added to the translation infrastructure, so memory accesses and instruction commits could be tracked. Then, the control flow of QEMU had to be changed so that `main()` would return and `run_vm()` would replace it as the entry point to the main loop. Linking QEMU as a shared object also necessitated subtle changes to the translation architecture. On x86-64 machines, global variables within shared objects can be placed anywhere within the virtual address space. The translator, generating 32-bit relative jumps, emitted code that would then fail since all of the jumps were to addresses beyond $\pm 2^{31}$ from their origin. QEMU's internal representation of time is designed to mirror time on the host machine. While this is fine for simple emulation, many OS facilities are triggered by the system timer, so the system timer was modified to mirror simulation time as closely as possible. With these modifications, and a change to the build process to make it build as a shared object, QEMU is ready to serve PSST as a VM module.

4. Preliminary Physical Models. There are currently two physical models of interest with very different goals. `faultmodel.so` is a very simple transient memory error injector

and `cachemodel.so` is a simple cache-only timing model. With `cachemodel.so`, the first results of real simulations could finally be obtained.

4.1. A Resiliency Model- `faultmodel.so`. The effect of transient errors in caches and processor pipelines is a very important problem. An initial exploration into using PSST to analyze the effects of these faults is `faultmodel.so`. `faultmodel.so` simply flips a single bit somewhere in physical memory, waits a random number of cycles, then, if it hasn't been overwritten, flips it back, waits a random number of cycles, and repeats. Since only one fault is active at any time, `faultmodel.so` also compares every memory access to the address of the fault and alerts the user via the console when one of these addresses is read. Visible results of booting a system and running applications with `faultmodel.so` activated have included system crashes, wrong results from applications, and file corruption.

4.2. A Timing Model- `cachemodel.so`. `cachemodel.so` is a model of an arbitrarily-deep uniprocessor cache hierarchy. With fill times, depth, and per-level cache dimensions given in the configuration file, `cachemodel.so` simulates all of these levels with a random replacement policy. In `cachemodel.so`, a processor is assumed to run at one instruction per cycle assuming that every memory operation hits in the L1 cache. When a miss occurs in any level, the time required for a fill in that level is added to the total cycle count for a given time quantum. When that time quantum is finished,

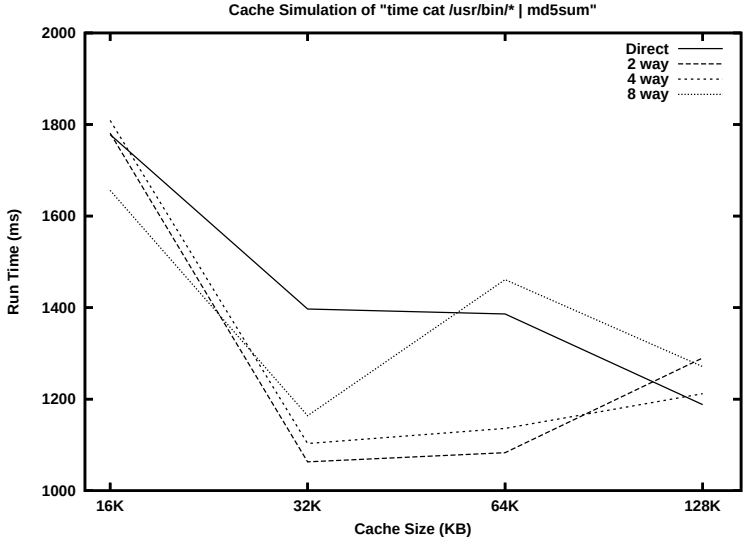


FIG. 4.1. *Cache simulation results of simple command under Linux, average of 2 trials, 16B cache lines, varying L2, fixed L1 and icache.*

Figure 4.1, though fraught with noise due to the low number of trials and unpredictable behavior of the Linux scheduler, is the first example of simulation results created by running actual applications, albeit very simple ones, on PSST. As expected, bigger cache means the program runs faster. However, some of the properties of this graph remain unexplained, such as the apparent minimum at 32K, and more trials will have to be done before these can be explained adequately.

5. Future Work. For the immediate future, there is a clear path from parallel emulation of multiple uniprocessors to simulation of parallel computing architectures. What is needed is

an interface for and software implementation of a NIC that can communicate through SST's protocol. Once these are available for PSST, its utility as a processor simulator component for a parallel architecture simulation framework will be realized.

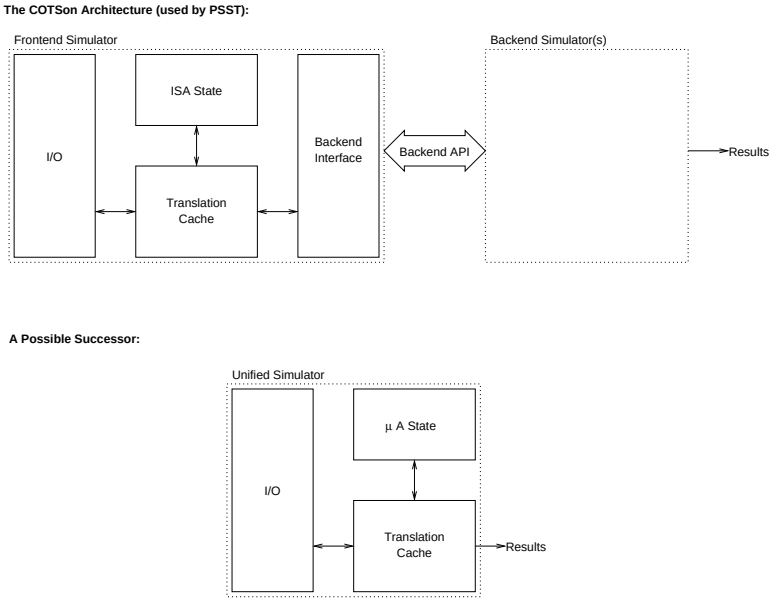


FIG. 5.1. *A possible future direction; combination of models with VM in a unified dynamically recompiled architecture.*

Further out, the author is reconsidering the fundamental software architecture of simulators like COTSon or PSST. Figure 5.1 illustrates both the current architecture and a possible future direction that combines the model A microarchitecture model is shown but other models could also be combined with the dynamic recompiler based emulator to create a dynamically generated simulator.

6. Conclusions. Though it is still very much under development, the preliminary results from PSST show it to be a promising direction in simulation research. The fact that the PSST simulation runs fast enough to be used interactively, even with a basic timing model running holds promise for debugging and software development. A possible development strategy using PSST in SST is to build a program, transfer it to the simulated machine, run it once on the simulated machine interactively with a lightweight timing model and a debugger, and if the results are promising, then run it again with a more robust timing model. PSST, as implemented is a retargetable microarchitecture simulator extensible by modules, which can be rapidly designed to target specific aspects of processor architecture.

REFERENCES

[1] F. BELLARD, *QEMU, a fast and portable dynamic translator*, in Annual Technical Conference, USENIX, 2005.
[2] U. CATALYUREK, E. BOMAN, K. DEVINE, D. BOZDAG, R. HEAPHY, AND L. RIESEN, *Hypergraph-based dynamic load balancing for adaptive scientific computations*, in Proc. of 21st International Parallel and Distributed Processing Symposium (IPDPS'07), IEEE, 2007.
[3] A. FALCÓN, P. FARABOSCHI, AND D. ORTEGA, *Combining simulation and virtualization through dynamic sampling*, in ISPASS, Apr 2007.
[4] A. RODRIGUES, *The structural simulation toolkit*, Sep 2007. `http://www.cs.sandia.gov/sst/`.

Applications

Necessity is the mother of invention and, ultimately, applications drive the advances in computational science, mathematics, and algorithms. The papers in this section span several disciplines and utilize advanced mathematical and computational tools to address important problems and applications in their respective fields. Application areas covered are diverse and include molecular dynamics, modeling cubits and tunneling diodes, plasma simulation, and verification of complex interface tracking algorithms.

Z. Wen
S.S. Collis

January 11, 2010

MOLECULAR DYNAMICS SIMULATIONS OF ELECTRO-ACTIVE SILICA NANOPARTICLE DECORATED WITH RIGID POLYMERS

SABINA MASKEY*, FLINT PIERCE†, DVORA PERAHIA‡, STEVEN J. PLIMPTON§, AND GARY S. GREST¶

Abstract. Molecular dynamics (MD) simulations have been utilized to study the conformation and structure of silica nanoparticles grafted with dialkyl poly *para* phenyleneethynylene (PPE) electro-active polymers. The conformation of PPEs determines the conjugation length and their assembly mode which in turn affects their electro-optical properties. In solution the conformation of the PPE is determined by molecular parameters including the length of the polymer and nature of the side chains coupled with molecular interactions. The present study investigates the structure and conformation of ethylhexyl PPE attached to silica nanoparticles in both a good (toluene) and poor solvent (vacuum) as a first step in understanding how these nanoparticles associate in solution.

1. Introduction. Nanoparticles (NP) exhibit unique properties that differ from those of bulk materials and molecules. Much of the interest in NPs arises from their large surface area and their small size which is comparable to that of many molecules together with their unique electro-optical and magnetic properties. Their electro-optical characteristics differ from that of bulk due to quantum confinement on the nanoscale. An immense effort is under way to form defect free NP arrays of noble metals, harnessing their unique properties for highly miniaturized device. New device applications arise from existing knowledge together with immerging new phenomena, as a result of the interactions between the NPs. Among these are plasmonic devices in which noble metal NPs support localized surface plasmon resonances which exhibit a high degree of optical field confinement with a notable sensitivity to their local environment. In an ordered NP array coherent interactions arise from multiple scattering as light that is scattered or emitted undergoes multiple scattering by the regularly spaced particles [1, 18].

Multiple studies have demonstrated the potential applications of metal NPs for highly sensitive photonic devices controlling, manipulating, and amplifying light, where the sensitivity to the environment offers a means for developing chemical and biological sensing devices. These applications as well as potential nanoelectronics and memory storage, all require stabilization of the NPs and provide pathways for assembly without altering the unique characteristics that result from the nanometer dimension. With a plethora of techniques available for patterning NPs on surfaces, obtaining defined large scale organized arrays remains a challenge.

Controlling the interaction between nanoparticles is a key to their utilization. Modifying the surface of NPs allows control of adhesion and electro-optical communication. Organic chains are often used where the chemical nature depends on the specific application. The collective behavior of these functionalized nanoparticles depends on the interactions between the modifying groups and the nanoparticles as well as on interaction between the grafted chains themselves.

Molecular dynamic simulations (MD) provide a powerful tool to investigate the molecular conformation of the modifiers and elucidate the interaction in nanoparticle complexes. Lane et al. for example have used molecular dynamics (MD) simulations to determine the forces between silica nanoparticles functionalized with poly(ethylene oxide) (PEO) in wa-

*Department of Chemistry, Clemson University,smaskey@clemson.edu,
†Department of Chemistry, Clemson University,fpierce@clemson.edu,
‡Department of Chemistry, Clemson University,dperahi@ces.clemson.edu,
§Sandia National Laboratories,sjplimp@sandia.gov,
¶Sandia National Laboratories,gsgrest@sandia.gov

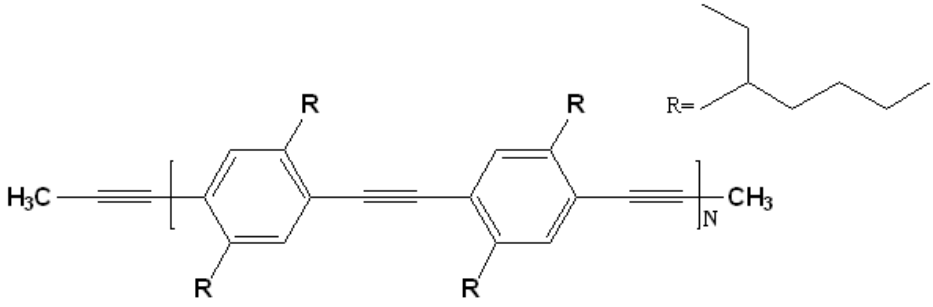


FIG. 1.1. *The chemical structure of poly para phenyleneethynylene(PPE). Here R represents ethylhexyl.*

ter [9]. They characterized the correlation between the chemical nature of the coating and the interactions between nanoparticles and the surrounding solvents. MD simulation have also been carried out to understand the physical interaction between poly(dimethylsiloxane) (PDMS) melts and bare silica nanoparticles with emphasis on the structure and dynamics of PDMS near silica surfaces [16]. Simulations have been carried to determine the forces between nanoparticle in both implicit and explicit solvents [3, 14, 13].

Using MD simulations, the current study investigates the conformation and structure of dialkyl *para* phenyleneethynylene (PPEs) grafted on Si nanoparticle. PPEs are electro-optically active polymers. Their applications from sensing to organic electronics rely on their response [2, 6, 15, 17]. PPE coated nanoparticles form an electro-optical responsive system where the PPEs would tune the interactions between the NPs. As a first step we have coated a silica nano particle with short PPE chains and investigated the conformation of the PPEs at the interface of the particle in vacuum and in contact with a solvent.

As shown in Figure 1.1, the backbone of PPE consists of alternate single and triple bond in conjunction with aromatic rings. The single bonds along the backbone allow the aromatic ring to freely rotate along the long axis of the molecule. When the aromatic rings are confined in a plane, the overlap of π orbitals result in delocalization of electrons along extended segment of the backbone and the backbone of the PPE becomes fully conjugated. As a result, PPE are intrinsic semi-conductor and often emit light on excitation.

The experimental studies carried out by small angle neutron scattering (SANS) have shown that di-alkyl PPE in toluene, which is a good solvent for backbone, forms complex phase diagram depending on the concentration and temperature including molecular solutions, micellar structures and fragile gels [11]. The conformation of the PPEs in these three phases depends on the degree of constrain of the molecule.

Conformational studies of single chains of various di-substituted di-alkyl PPEs have been carried out by the authors using MD simulations in explicit and implicit solvents and in vacuum. These studies have shown that these PPEs molecules assume rigid rod-like to worm-like structures. These molecules do not form random coils or collapsed structures for degrees of polymerization up to N=480. Figures 1.2a and 1.2b show the snapshots of octadecyl PPE (N = 60) in toluene and in vacuum, respectively. As seen from these snapshots in vacuum, a poor solvent, side chains lie along the backbone and are associated with each other and forms energy driven configuration. In case of toluene, a good solvent, the side chains are dispersed with little or no interaction with each other and with backbone giving rise to entropy driven configuration.

Using MD simulation the current study is aimed to determine the conformation and struc-

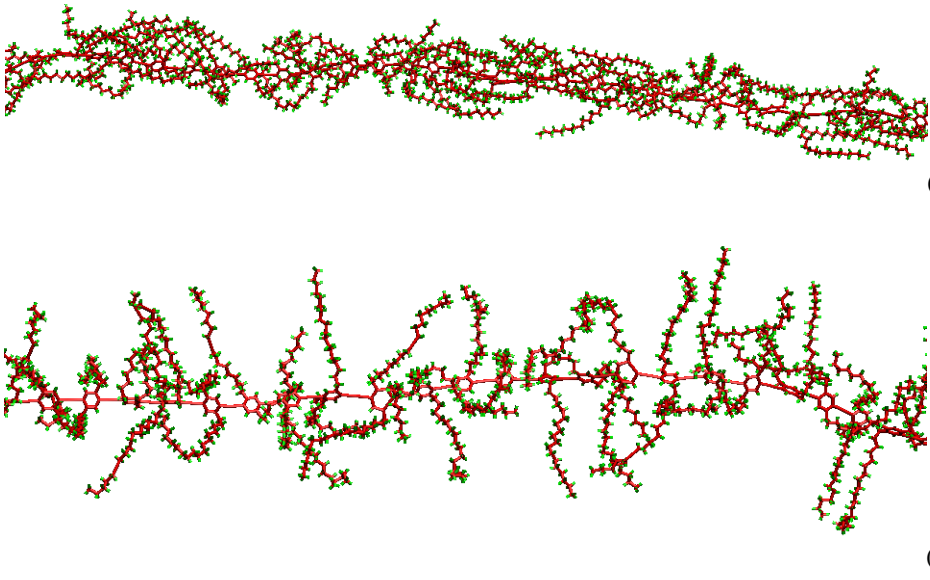


FIG. 1.2. *Snapshot of middle section of an equilibrated state of octadecyl PPE* ($N = 60$) *in (a) vacuum and (b) toluene. For clarity only 20 segments of the PPE chain is shown.*

ture of ethylhexyl PPE attached to a silica nanoparticle in both good and poor solvents. By coating silica nanoparticle with PPE, we have formed a new class of electro-active nanoparticles. This study is a first step in an effort to understand the properties of electro-active nanoparticle coated with rigid polymers to reach our goal of designing nanoparticles that can effectively communicate with each other via electron or photon transfer for sensors and photovoltaic devices.

2. Simulation Methodology and Results. The ethylhexyl PPE and toluene molecules as well as silica nanoparticle core used in this work are modeled using the standard OPLS-AA (Optimized Parameter for Liquid Simulator) framework of Jorgensen*et al.* [7, 8] which shall be simply referred here as OPLS. OPLS has several potential terms,

$$U_{OPLS-AA} = U_{nb} + U_{bond} + U_{ang} + U_{tor} + U_{imp} \tag{2.1}$$

Nonbonded interactions U_{nb} are a sum of standard 12-6 Lennard-Jones (LJ) and electrostatic potentials [7, 8],

$$U_{nb}(r_{ij}) = U_{LJ} + U_{coul} = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + k_{coul}\frac{q_i q_j}{r_{ij}} \tag{2.2}$$

where ϵ_{ij} is the LJ energy and σ_{ij} is the LJ diameter for atoms i and j, q_i and q_j are their partial charges. For atoms of different species, geometric mixing rules are used, $\epsilon_{ij} = (\epsilon_i\epsilon_j)^{\frac{1}{2}}$ and $\sigma_{ij} = (\sigma_i\sigma_j)^{\frac{1}{2}}$. Nonbonded interactions are calculated between all atomic pairs on different molecules in addition to all pairs on the same molecule separated by three or more bonds, though the interaction is reduced by a factor of $1/2$ for atoms separated by three bonds. All LJ interactions were cut off at 12Å. All coulomb interactions for atom pairs closer than 12Åare

calculated in real space, those outside this range calculated in reciprocal (Fourier) space by use of a standard particle-particle particle-mesh (PPPM) algorithm [4] and a precision of 0.001. Bond and angle potentials are harmonic in form

$$U_{bond,ang}(r_{ij}, \theta_{jk}) = k_{r,\theta}(r_{ij}, \theta_{jk} - r_0, \theta_0)^2 \tag{2.3}$$

with r_{ij} the distance between atoms i and j, r_0 their equilibrium separation, θ_{ijk} the angle between the vectors $\mathbf{r_{ji}}$ and $\mathbf{r_{jk}}$, and θ_0 the equilibrium value. The torsional (dihedral) component of the OPLS-AA potential is given by

$$U_{tor} = \sum_n k_n \cos^n \phi_{ijkl} \tag{2.4}$$

where ϕ_{ijkl} is the dihedral angle, and k_n depends on the identities of atoms i, j, k, and l. In the OPLS framework, the out of plane (improper) potential has the same form as that of the torsional potential, but for the sake of simulation in the LAMMPS molecular dynamics package, this potential has been recast in the following form

$$U_{imp} = K_{imp}\left[1 + d\cos\left(n\theta_{imp}\right)\right] \tag{2.5}$$

For our initial study we attached ethylhexyl PPE chains of 6 repeat units to a 5nm diameter nanoparticle with the resulting coverage of 0.56 chains per nm^2. The construction of the PPE as well as toluene samples was done by using Polymer Builder and Amorphous Cell modules of Material Studio from Accelrys Inc ©. The as-built samples were originally given conformations consistent with polymer consistent force field (pcff) as OPLS potential is not implemented in Material Studio. An in-house conversion utility was used to modify Material Studio data files into LAMMPS data files. All potential parameters were converted from pcff to OPLS.

A 5nm diameter silica nanoparticle was cut from bulk amorphous silica which was generated from a melt-quenched process as described by Lorenz *et al.* [10] and then annealed to produce a surface OH concentration consistent with the experimental values. The PPE chains were attached to the nanoparticle by removing 50 hydrogen atoms from surface OH groups with cut off distance of 0.8 nm between two H atoms. A hydrogen atom was subsequently removed from one of the terminal carbons on the PPE chain and a new bond formed between the carbon of PPE chain and the surface oxygen of the indicated. The resulting coverage was 0.56 chains per nm^2. To maintain charge neutrality the partial charges on the oxygen atom on the surface and the two hydrogen atoms and the carbon atom were slightly modified based on the charges of ether in OPLS. The charge of H atoms were changed from −0.06 to 0.03, charge of O atom was changed from −0.683 to −0.43 and charge of C atom was modified to 0.14 from −0.12.

The toluene sample was equilibrated for 1 ns at $T = 300$ K and zero pressure using an NPT ensemble using Nose-Hoover temperature thermostat at 1 fs timestep[5]. The resulting density was 0.867 g/cc. We used 43,200 toluene molecules (648,000 atoms) which was sufficient to completely solvate the nanoparticle which contained 23,365 atoms (4965 atoms of silica nanoparticle and 18,450 atoms for the 50 chains of PPE with each chain containing 369 atoms). A hole in the center of the toluene sample to accommodate the nanoparticles was created by using the (indent) fix in LAMMPS. The radius of the hole was increased slowly from an initial radius of zero to approximately the diameter of nanoparticle core plus the PPEs. The system was further equilibrated for 0.5 ns before the nanoparticle and solvent were merged. The combined system was equilibrated for 1 ns in the NPT ensemble, after which the system was run for 6 ns in the NVE ensemble with a Langevin. The length of the final simulation

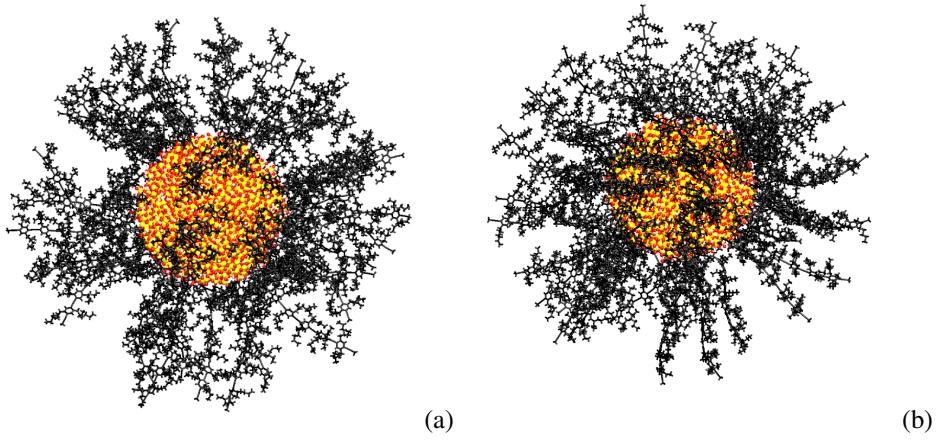
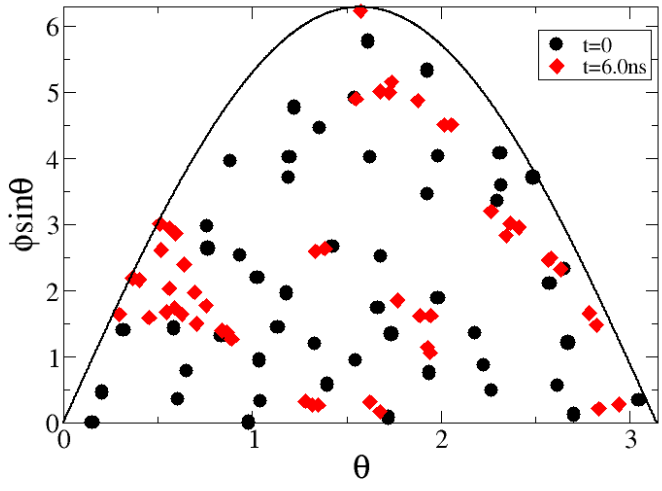


FIG. 2.1. *Snapshot of silica nanoparticle with 50 PPE chains of in (a) vacuum and (b) toluene. For clarity chains are colored in black.*

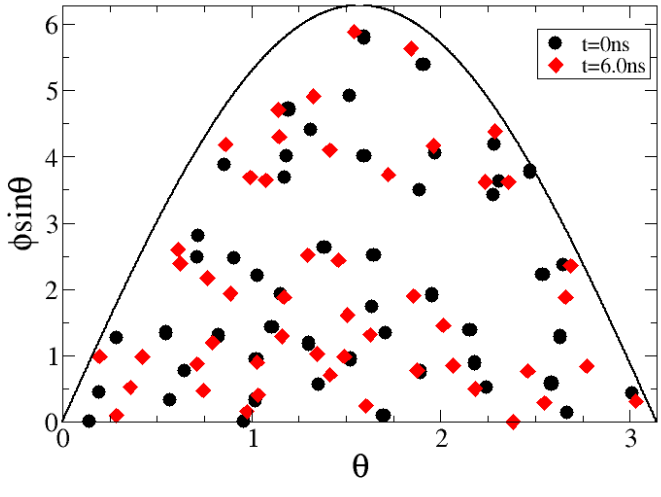
cell $L \approx 20$ nm was approximately 25% larger than diameter of silica nanoparticle decorated with the PPE which was sufficient to prevent the interaction of PPE with itself through the periodic boundaries. All simulations were carried out using the LAMMPS classical MD code [12]. Numerical integration was performed using velocity-Verlet algorithm with 1 fs time step. Most of the simulations were carried out in an NVE ensemble using a Langevin thermostat with damping $\Gamma = 0.01\ fs^{-1}$ at 300K. The nanoparticle core was simulated as a rigid atom. For comparison we also simulated the nanoparticles coated with PPE molecules in vacuum using NVE ensemble with a Langevin thermostat for 6 ns at time-step of 1 fs.

Figures 2.1a and 2.1b show silica nanoparticles coated with 50 chains of PPE in vacuum and toluene. For clarity, PPE chains have been colored black and toluene is not shown. Visually, there are few differences between the orientation of chains of PPE in vacuum and in toluene. In toluene PPE chains are more separated from each other than in vacuum where PPE chains form clusters.

The focus of this initial study is to determine the conformation and dynamics of PPE chains attached to a nanoparticle core in various solvent environments. It further investigates the inter chain interactions. In solutions,in good solvents toluene, flexible polymers assume random coil conformations while collapsed structures are formed in poor solvents. For rigid polymers, single chains form extended conformations in both good and poor solvents as shown in Figure 1.2. For multiple chains of rigid polymers attached isotropically to a nanoparticle core, one might expect these chains to be 'dissolved' in a good solvent and exhibit extended conformations with little or no association between neighboring chains, while for a poor solvent, attached chains may form clusters as they avoid solvation in an attempt to minimize energy. The morphologies of coated nanoparticles are reflected in such conformational changes of the attached polymers. By analyzing the association or dissociation of polymer chains, one can discern the effect of a solvent on the nanoparticle. In order to understand the association of polymer chains, one can utilize the orientation of the chain. The location of the terminal carbon atom of each chain relative to the nanoparticle's center of mass was calculated as a function of time. From this position vector, the azimuthal (θ) and polar (ϕ) angles, representing the chains orientation, were determined. The range of ϕ is 0 to 2π and that of θ from 0 to π. An area-conserving mapping was used to convert spherical angular coordinates into rectangular dimensions, as seen in Figure 2. Depending on the relative



(a)



(b)

FIG. 2.2. *Plot of $\phi \sin(\theta)$ versus θ for silica nanoparticle with 50 chains of PPE in (a) vacuum and (b) toluene.*

orientation of the chains which are represented by the points in the plot, it is possible to tell whether there is an isotropic or anisotropic distribution of the chains. A constant density of points on the mapping corresponds to an isotropic and unassociated ('dissolved') distribution of chains around the nanoparticle. An anisotropic distribution implies chains are clumped or associated with each other in an undissolved state.

The plots shown in Figures 2.2a and 2.2b represent the relative position of chains at two different times for two solvent conditions, vacuum and toluene. The first time is at the beginning of the simulation, and the second is after 6 ns. At the beginning of the run, for

both environments, we see dispersed distributed points which shows that there is an isotropic distribution of chains around the nanoparticle core. After 6 ns, in both vacuum and toluene, we see a change in orientation of the chains as none of the chains coincides with their starting positions. In vacuum, there are a number of clusters of chains after 6 ns, indicating an anisotropic distribution and association of the chains. Conversely, in toluene, an isotropic distribution is maintained by the chains which indicates that there is no association between the chains. These chains remain solvated by the toluene. Hence, chains are stretched out in a good solvent (toluene) whereas in a poor solvent (vacuum) they form clusters.

Another method of determining the conformational changes is by calculating the nanoparticle's radius of gyration (R_g), which depends on the quality of the solvent. Despite the obvious differences in the conformations of the chains as discussed above in toluene and vacuum, the calculated R_g (of both chains and nanoparticle core) varied by less than 1 %, principally due to the large mass of the nanoparticle core and rigidity of the attached PPE molecules.

3. Conclusions. MD simulations were used to investigate the conformation and structure of silica nanoparticle coated with ethylhexyl PPE, a highly rigid and electro-active polymer in both good and poor solvents. PPE-grafted nanoparticle was successfully made and inserted into solvents. In contrast to grafted flexible hydrocarbon chains, the PPE molecules remains stretched out away from the surface of the NP. Solvents affect the distribution of the chains around the NP. In vacuum which is a poor solvent for the PPEs, the chains cluster and the distribution around the NP diverges from isotropic. The study showed that in toluene, a good solvent, PPE chains were stretched out and isotropically distributed. This study will be further extended to quantify the solvent effects and ultimately define the parameters that control the interactions between PPE grafted nano particles. In future, we will carry out further studies to determine the effects of varying the length of polymer chains, the nature of the side chains, and the coverage. The forces between two nanoparticles as a function of distance will also be determined.

4. Acknowledgement. The authors thank J. Matthew D. Lane for providing silica nanoparticle data file. This work was made possible by advanced computational resources deployed and maintained by Clemson Computing and Information Technology and also would like to acknowledge the support of staff from the cyber infrastructure and Technology Integration group, Clemson University.

REFERENCES

[1] B. Auguie and W. L. Barnes, *Collective resonances in gold nanoparticle*, PRL, 101 (2008), p. 3902.
[2] A. J. Heeger, *Semiconducting and metallic polymers: The fourth generation of polymeric materials*, Angew. Chem. Int. Ed., 113 (2001), pp. 2591–2611.
[3] B. J. Henz, T. Hawa, and M. R. . Zachariah, *Mechano-chemical stability of gold nanoparticles coated with alkanethiolate sams*, Langmuir, 24 (2008), pp. 773–783.
[4] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Adam Hilger-IOP, Bristol, 1988.
[5] W. G. Hoover, *Canonical dynamics: Equilibrium phase-space distributions*, Phys. Rev. A, 31 (1985), pp. 1695–1697.
[6] C. Hoven, R. Yang, A. Garcia, A. J. Heeger, T.-Q. Nguyen, and G. Bazan, *Ion motion in conjugated polyelectrolyte electron transporting layers*, J. Am. Chem., 129 (2007), pp. 10976–10977.
[7] W. L. Jorgensen, J. D. Madura, and C. J. Swenson, *Optimized intermolecular potential for liquid hydrocarbons*, J. Am. Chem. Soc., 106 (1984), pp. 6638–6646.
[8] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, *Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids*, J. Am. Chem. Soc., 118 (1996), pp. 11225–11236.
[9] J. M. D. Lane, A. E. Ismail, M. Chandross, C. D. Lorenz, and G. S. Grest, *Forces between functionalized silica nanoparticles in solution*, Phys. Rev. E, 79 (2009), pp. 050501–050504.

[10] C. D. Lorenz, E. B. Webb III, M. J. Stevens, M. Chandross, and G. S. Grest, *Frictional dynamics of perfluorinated self-assembled monolayers on amorphous sio$_2$*, Trib. Letter, 19 (2005), p. 93.

[11] D. Perahia, R. Traiphol, and U. H. F. Bunz, *From molecules to supramolecular structure: Self assembling of wirelike poly(p-phenyleneethynelenes)s*, Macromolecules, 34 (2001), pp. 151–155.

[12] S. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, J. Comp. Phys., 117 (1995), pp. 1–19.

[13] Y. Qin and K. Fichthorn, *Molecular dynamics simulation of the forces between colloidal nanoparticles in lennard-jones liquid*, J. Chem. Phys., 119 (2003), p. 9745.

[14] Y. Qin and K. A. Fichthorn, *Molecular dynamics simulation of the forces between colloidal nanoparticles in n-decane solvent*, J. Chem. Phys., 127 (2007), p. 144911.

[15] B. J. Schwartz, *Conjugated polymers as molecular materials:how chain conformation and film morphology influence energy transfer and interchain interactions*, Annual Rev. Phys. Chem., 54 (2003), pp. 141–172.

[16] J. S. Smith, O. Borodin, G. D. Smith, and E. M. Kober, *A molecular dynamics simulation and quantum chemistry study of poly(dimethylsiloxane)-silica nanoparticle interactions*, J. Polym. Sci. B, 45 (2007), pp. 1599–1615.

[17] S. W. Thomas III, G. D. Joly, and T. M. Swager, *Chemical sensors based on amplifying conjugated polymers*, Chem. Rev., 107 (2007), pp. 1139–1386.

[18] J. Zhao, L. J. Sherry, G. C. Schaltz, and R. P. V. Duyne, *Molecular plasmonics: Chromophores-plasmons coupling and single-particle nanosensors*, IEEE Journal of Selected Topics in Quantum Electronics, 14 (2008), pp. 1418–1429.

3D TCAD MODELING OF CANDIDATE STRUCTURES FOR THE SILICON QUBIT

NICOLE L. ROWSEY*, RICHARD P. MULLER†, AND RALPH W. YOUNG‡

Abstract. This paper describes the TCAD modeling of possible candidate structures for the realization of the solid-state silicon qubit. The typical Poisson+Drift/Diffusion TCAD solver should at the very least implement Fermi-Dirac statistics, incomplete ionization models, and temperature-dependent band structures to account for carrier densities down to 50K. The resulting electrostatic potential can be used iteratively with an outside Schroedinger solve in a small "quantum region" to calculate quantum dot energy levels more accurately. Ideally, a full Schroedinger-Poisson solver should be implemented in one tool. Preliminary results of implementation of these models using the FLOODS TCAD solver are presented and compared to industry tools, as well as experiment.

1. Introduction. Quantum computing is a highly-sought-after goal of many research groups, as several important yet classically-intractable problems in fields from cryptography to molecular dynamics become tractable in the quantum domain. To this end, physical quantum bits, or physical qubits, have been demonstrated in several systems. For example, nuclear spins have been probed with a nuclear magnetic resonance (NMR) spectrometer, and spin-controllable regions of confined charge, or quantum dot structures, have been realized in Gallium Arsenide (GaAs) semiconductor devices. However, quantum computation requires not just the realization and control of a single qubit, but the controlled interaction of several qubits, including error-correcting bits, to perform logic operations; in other words, quantum computation requires a logical qubit, which in fact consists of multiple interacting physical qubits. Systems such as NMR and GaAs possess inherent physical limitations that make it difficult to extend these examples into logical qubits. NMR qubits are not suitable because they occur relatively infrequently, and thus far apart, in their host material, and so it is difficult to maintain coherence between more than a few qubits at once. GaAs systems are fabricated using modern integrated circuit (IC) processing techniques, which gives a high degree of control over placement of the qubits, as well as the ability to place the qubits close together. However, GaAs nuclei have a large nuclear spin, which interferes with the qubit spin and leads very quickly to decoherence.

Silicon, on the other hand, has no nuclear spin. Furthermore, Si is the backbone of the modern IC industry, and is therefore the subject of a vast amount of research for the past 50 or 60 years. Silicon qubit designers thus have at their disposal a large body of knowledge and understanding, including not only the fabrication and operation of such devices, but also robust modeling tools that can be used to design and understand them.

Technology Computer Aided Design (TCAD), in particular, has a long successful history of modeling charge transport and electrostatic potential in semiconductor devices, especially Si. Robust finite-volume (FV) methods have been developed to solve Poisson's equation, coupled with the electron and hole continuity equations, in 1 to 3D semiconductor solids. The main limitation on a TCAD tool is mesh size, or how many points are required to ensure that the finite-volume discretization scheme is a stable and accurate approximation to a continuous system. On the average computer, only a few semiconductor devices can be simulated at a time. In more complicated device structures, sometimes only a small important region of a device can be simulated. Therefore several other tools and modeling techniques are needed in addition to TCAD for a full picture of a semiconductor system, and quite powerful ones for a whole IC. For example, density-functional theory (DFT) or molecular dynamics (MD)

*University of Florida, nrowsey@ufl.edu
†Sandia National Laboratories, rmuller@sandia.gov
‡Sandia National Laboratories, rwyoung@sandia.gov

calculations are needed to model electronic and material properties before simulating at the device level. At the device level, TCAD tools are best at modeling the electronic structure of semiconductor systems, handling larger structures and accounting for boundary conditions more realistically. Finally, the results of TCAD simulations are used to create a simplified picture of devices, which is used by compact-model circuit simulators to connect lumped-element devices together in a large network.

In this work, we present preliminary simulation results of Si metal-oxide-semiconductor (MOS) -based quantum dot structures using the Florida Object-Oriented Device Simulator (FLOODS) TCAD package, currently in development. Our preliminary simulation results compare well to Sentaurus Device, an industry TCAD solver, as well as experimental data of fabricated structures. We plan to extend this work by building a Schroedinger-Poisson capability into FLOODS that would be able to quantum-mechanically treat the quantum dot regions of our devices.

2. Basic TCAD: A Poisson and Drift/Diffusion Solver. TCAD solvers solve 3 things on a grid: Electrostatic potential, via Poisson's equation, and electron and hole concentration, via the electron and hole continuity equations. On a physical contact, electrostatic potential translates to voltage (V), and net charge flux (holes - electrons) translates to electric current (I). This results in a basic 3 equations, 3 unknowns problem.

Poisson's equation is as follows:

$$\epsilon \nabla^2 \Psi = -Q, \tag{2.1}$$

where ϵ is the material-dependent permittivity, Ψ is electrostatic potential, and Q is charge density. In the most common TCAD case, Q is accounted for by adding up the concentrations of electrons (n), holes (p), and n- and p-type ionized impurities, such as arsenic or boron, called donors and acceptors (N_d^- and N_a^+). Finally, multiplying by q, the charge on an electron, we have:

$$Q = q(n - p + N_d^- - N_a^+). \tag{2.2}$$

The continuity equations for electrons and holes are as follows:

$$\begin{aligned} \frac{dn}{dt} &= \frac{1}{q}\nabla \cdot (q\mu_n n\mathscr{E} + qD_n \frac{dn}{dx}) \\ \frac{dp}{dt} &= -\frac{1}{q}\nabla \cdot (q\mu_p p\mathscr{E} - qD_p \frac{dp}{dx}), \end{aligned} \tag{2.3}$$

where $\mu_{n,p}$ are the electron and hole mobilities, $\mathscr{E}$ is electric field, and $D_{n,p}$ are the electron and hole diffusion coefficients. Equations (2.3) state that there are two main contributions to the transport, $\frac{d}{dt}$, or motion over time of our two free charged carriers, electrons and holes. The first term, called the "drift" term, arises from any electric field gradient ($\mathscr{E}$) in the device. It is the first term in the parenthetical sums in (2.3). Its experimentally-determined proportionality constant, μ, is called mobility. Electric field gradients arise from the distribution of the mobile charges in the device, as well as any bias voltage conditions on the contacts, and are calculated from the electrostatic potential, Ψ, via

$$\mathscr{E} = -\nabla \Psi. \tag{2.4}$$

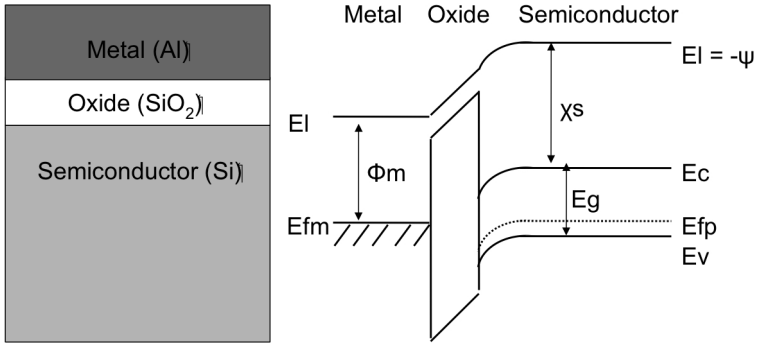
The second contribution to carrier transport is from any concentration gradient of n or p, the $\frac{dn}{dx}$ term in (2.3). This is a "diffusion" term, and its experimentally-determined proportionality constant is D, diffusivity.

Equations (2.1) and (2.3) present three coupled partial differential equations with three variables, Ψ, n and p. The effects of drift and diffusion coupled with Poisson's equation do not by any means give an exhaustive account of all the physics involved in the operation of a semiconductor device. For example, these equations neglect collision or scattering terms [6]. Furthermore, the continuity equations as expressed in (2.3) above are derived from Boltzmann statistics, and are an approximation to the more accurate way of expressing transport as $n\nabla E_{Fn}$ and $p\nabla E_{Fp}$, where E_F is the Fermi level. Equation (2.3) assumes Maxwell-Boltzmann (MB) statistics in that it uses the Einstein relation

$$\frac{kT}{q} = \frac{D}{\mu}. \tag{2.5}$$

where k is Boltzmann's constant, and T is temperature. This Maxwell-Boltzmann simplification does not apply under all conditions. More advanced models that build on (2.3), as well as a discussion of when these models are needed, are presented in later sections.

3. Contact Equations. In order to solve the 3 equations and 3 unknowns described above in section 2, we need boundary conditions. All of the candidate structures we consider here are based on the metal-oxide-semiconductor field-effect-transistor (MOS or MOSFET). Figure 3.1(a) shows a cartoon of a typical MOSFET. There are two different types of boundary conditions relevant to this kind of structure.



(a) Basic MOS Structure (b) MOS Energy Band Diagram

FIG. 3.1. *(a) A basic MOS device. The Si section is the top few microns of a Si wafer. SiO_2 is grown on the top surface of the wafer through chemical reaction with oxygen in the air. Finally, a metal is deposited on top. (b) An energy band diagram for a MOS device with p-type bulk and metal-gate work-function contact.*

The first is a simple metal-contact work-function difference. Figure 3.1(b) shows an energy band diagram for a MOS device with p-type bulk and a metal contact. The work function, Φ_m, which we chose to measure in volts, is the minimum energy required to move an electron from the metal (residing in the Fermi level of the metal, E_{Fm}) to the vacuum level E_l. This is similar to the electron affinity, χ_S in Si, the minimum energy required to move an electron from the conduction band, E_C, to the vacuum. If we define electrostatic potential as the band-bending in the vacuum level,

$$\Psi = -E_l, \tag{3.1}$$

with both Ψ and E_l measured in volts; and if we define the zero reference point as the Fermi level in the metal (E_{Fm}) at zero applied bias,

$$E_{Fm} = 0 \text{ at } V_{applied} = 0, \tag{3.2}$$

then we can express our work-function-difference boundary condition as

$$-\Psi - \Phi_m = E_F, \tag{3.3}$$

on all the nodes that touch both metal and oxide, where in general Ψ is constant within a metal and E_F is continuous between materials.

An ohmic contact is more complicated. Physically, it is a contact to the bottom of the Si wafer that is required to keep both the actual and simulated system from floating, or having undefined voltage values. We model this as a boundary condition "somewhere in the bulk," which in practice means "on the first Si nodes that are far enough away from the wafer surface to be considered bulk." The bulk is in equilibrium, which means that, first, the Fermi levels are equal to each other, and equal to the voltage applied at the ohmic contact:

$$E_{Fn} = E_{Fp} = V_{applied}; \tag{3.4}$$

Also, local conservation of charge is maintained:

$$Q = q(n - p + N_d^- - N_a^+) = 0. \tag{3.5}$$

Equations (3.4) and (3.5) provide three boundary conditions for our three variables. Other boundary conditions necessary for simulation include Dirichlet and Von Neumann boundary conditions on non-contacted edges, and the condition that Ψ is continuous everywhere.

4. Fermi-Dirac Statistics. We have, in our initial discussion of the continuity equations, and again in defining ohmic-contact boundary conditions, used the concept of the Fermi level, E_F. The Fermi level is an important concept from statistical mechanics. It is the lowest energy level in a solid that is populated with electrons. Its position in a metal is inside the conduction band, (the band of energy levels in which conduction of electrons occurs). This accounts for why metals are good electrical conductors. The Fermi level in an insulator resides near the center of a large band gap, E_g, far away from the conduction- or valance-band-edges, E_C or E_V, such that negligible conduction occurs. In a semiconductor, however, the Fermi level resides somewhere within a moderate-sized band gap, and its location can be controlled via the level of impurity doping. If E_F is placed sufficiently near E_C, as is the case for n-type doping, we can apply enough of a voltage to push E_F into E_C, controlling the current transport in a semiconductor device via externally applied voltage. A similar case is made for p-type doping, illustrated in figure 3.1(b), where E_F is placed near E_V for conduction of holes.

If we take into account the fact from quantum mechanics that the electron is a Fermion, a particle with half-integer spin, the Fermi level in semiconductor physics is defined by the Fermi-Dirac (FD) statistics of the carriers,

$$\begin{aligned} n &= N_C F_{\frac{1}{2}}\left(\frac{(E_{Fn} - E_C)}{kT}\right) \\ p &= N_V F_{\frac{1}{2}}\left(\frac{(E_V - E_{Fp})}{kT}\right), \end{aligned} \tag{4.1}$$

where $N_{C,V}$ are the effective density of states in the conduction and valance bands, E_C and E_V are the conduction and valance band edges, k is Boltzmann's constant, T is temperature, and $F_{\frac{1}{2}}$ is the Fermi-Dirac integral of order $\frac{1}{2}$:

$$F_{\frac{1}{2}}(\eta) = \int_0^\infty \frac{x^{\frac{1}{2}}\, dx}{1 + \exp(x - \eta)}. \tag{4.2}$$

This integral is intractable. However, several practical approximations are available to us if we wish to carry out calculations on our system.

The most basic of these approximations is to interpret MB carrier statistics,

$$\begin{aligned} n &= N_C \exp(\frac{(E_{Fn} - E_C)}{kT}) \\ p &= N_V \exp(\frac{(E_V - E_{Fp})}{kT}), \end{aligned} \tag{4.3}$$

as a classical approximation to FD statistics. Maxwell-Boltzmann statistics do not take into account the quantum mechanical idea of particle spin, treating all particles as bozons, or particles of integer spin. However, the approximation is valid under certain physical conditions.

Mathematically and physically the approximation is justified when $E_F n$ is sufficiently (several kT) below the conduction band E_C (or $E_F p$ is sufficiently above E_V). If this is the case, we can use a Taylor series expansion on the exponential term in (4.2), reducing FD statistics to the more manageable equations in (4.3). The MB equations invert easily to give a tractable expression for E_F that can be used in the ohmic contact equations (3.4) above:

$$\begin{aligned} E_F n &= E_C + kT \ln(\frac{n}{N_C}) \\ E_F p &= E_V - kT \ln(\frac{p}{N_V}). \end{aligned} \tag{4.4}$$

We can see from the discussion above that the Fermi level depends on many parameters, including doping, temperature, carrier density, and electrostatic potential (since $E_C = -\Psi - \chi$, and $E_V = -\Psi - \chi - \frac{E_g}{2}$, from figure 3.1(b)). Thus, in determining whether E_F is close enough to the band edge, many different aspects of our physical operating environment must be considered.

First, the semiconductor must be non-degenerately doped; in other words, the semiconductor must have low impurity density. Low impurity density is generally considered to be in the range of $N_a, N_d < 1e18/cm^3$ or $1e19/cm^3$. Candidate structures for the Si qubit are undoped, to avoid noise. In semiconductor parlance, "undoped" does not translate to carrier concentrations of 0. In fact, it is impossible from a manufacturing standpoint to purify Si to this degree. The best manufacturers can do is a p-type impurity concentration of approximately $1e14/cm^3$. "Undoped" simply means that there was no intentional doping added. (For reference, the highest possible doping attainable in a semiconductor is approximately $1e20/cm^3$ or even $1e21/cm^3$ in the case of polySi - the closest approximation doped Si can come to a metal.)

Since $1e14 << 1e18$, our undoped candidate structures satisfy the low doping requirement for MB statistics. However, we must also consider that our candidate structures will be operated very low temperatures. Solid-state qubit systems are never operated at room temperature due to the high random thermal noise of carriers compared to spin state energies. In fact, these systems are operated at as close to $0K$ as possible, which for us is $100mK$.

If we consider MB as a classical approximation to FD statistics, then we also must consider that, to be able to treat a particle system as classical, the average inter-particle spacing, $\bar{R}$ of the system must be much greater than the statistical average de Broglie wavelength $\bar{\lambda}$ of our particles:

$$\bar{R} \gg \bar{\lambda} \approx \frac{h}{\sqrt{3mkT}} \propto \frac{1}{\sqrt{T}}, \tag{4.5}$$

where h is Plank's constant, m is the effective mass of the electron, $\lambda = \frac{h}{p}$ is de Broglie wavelength, $p = mv_{th}$ is momentum, and $v_{th} = \sqrt{\frac{3kT}{m}}$ is thermal velocity.

From equation (4.5), we see that as temperature decreases, the classical approximation becomes less and less valid, and indeed is invalid at $50K$ and $100mK$. Therefore, we must use FD statistics in simulating our candidate qubit structures, which will require a new trick to deal with the intractability of (4.1).

A widely used and successful treatment of this problem is the Joyce-Dixon approximation [2], which uses a power-series expansion to make a polynomial expression for both $F_{\frac{1}{2}}$ and E_F.

$$F_{\frac{1}{2}} = g_1 \exp(\eta) - g_2 \exp(2\eta) + ... + (-1)^{m+1} g_m \exp(m\eta)... \tag{4.6}$$

where η is shorthand for

$$\begin{aligned} \eta_n &= \frac{E_{Fn} - E_C}{kT} \\ \eta_p &= \frac{E_V - E_{Fp}}{kT}, \end{aligned} \tag{4.7}$$

and the g_m are the Laplace transform of $x^{\frac{1}{2}}$:

$$g_m = \sum_{m=1}^{\infty} \frac{\sqrt{\pi}}{2} m^{\frac{1}{2}}. \tag{4.8}$$

The reverted series for η can be expanded out to make a tractable expression for E_f, the first few terms of which are included below:

$$E_F = Ec + kT(ln(\chi_S) + A_1(\chi_S) + A_2(\chi_S)^2 + A_3(\chi_S)^3 + A_4(\chi_S)^4) \tag{4.9}$$

$$\begin{aligned} A_1 &= 3.53553...e-1 \\ A_2 &= -4.95009...e-3 \\ A_3 &= 1.48386...e-4 \\ A_4 &= -4.42563...e-6, \end{aligned}$$

where χ_S denotes either $\frac{n}{N_C}$ or $\frac{p}{N_V}$. In (4.9) above, $E_C + kT \ln(\frac{n}{N_C})$ is recognizable as the Boltzmann approximation, and the sum over $A_m(\chi_S)^m$ can be interpreted as a FD correction term. This power series is valid for $\chi_S < 8.0$, otherwise we can use the Sommerfeld expansion [7]:

$$E_F = Ec + kT((\frac{3\sqrt{\pi}}{4}\chi_S)^{\frac{4}{3}} - \frac{\pi^2}{6})^{\frac{1}{2}}. \tag{4.10}$$

Finally, equations (4.9) and (4.10) gives us a tractable and physically representative expression for the Fermi level that can be used in the boundary condition equation (3.4).

Implementing equation (4.9) on the ohmic boundary contacts is a required step. However, as the continuity equations (2.3) are also a simplification based on MB statistics, we must also develop a treatment for the bulk. We follow a method outlined in Mark Pinto's 1990 Stanford thesis [3]. From first principles, FD is accounted for by deriving a modified Einstein relation similar to (2.5). However, then we may not use the Scharfetter-Gummel finite-volume discretization scheme, which would sacrifice stability and demand impractically small grid spacing. Instead, Pinto treats the effect of FD statistics as a perturbation to the MB approximation,

$$\begin{aligned} n &= \gamma_n N_C \exp(\eta_n) \\ p &= \gamma_p N_V \exp(\eta_p), \end{aligned} \tag{4.11}$$

where the perturbation γ is defined as

$$\begin{aligned} \gamma_n &= F_{\frac{1}{2}}(\eta_n)\exp(-\eta_n) \\ \gamma_p &= F_{\frac{1}{2}}(\eta_p)\exp(-\eta_p). \end{aligned} \tag{4.12}$$

Then, effective fields $\mathscr{E}_n$ and $\mathscr{E}_p$ are introduced

$$\begin{aligned} \mathscr{E}_n &= -\nabla(\Psi + kT\ln(\gamma_n)) \\ \mathscr{E}_p &= -\nabla(\Psi - kT\ln(\gamma_p)). \end{aligned} \tag{4.13}$$

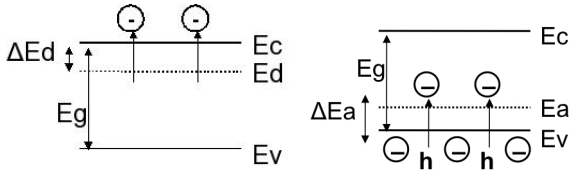
These effective fields are used in the continuity equations (2.3) instead of (2.4). In our FLOODS implementation, the Joyce-Dixon approximation for $F_{\frac{1}{2}}$ was used to define γ in the continuity equations. Pinto's thesis suggests implementing a modified Einstein relation such that

$$kT \rightarrow kT\frac{F_{\frac{1}{2}}(\eta)}{F_{-\frac{1}{2}}(\eta)}, \tag{4.14}$$

using this as a coefficient outside the continuity equations (2.3). This has not yet been implemented in FLOODS.

5. Incomplete Ionization of Impurities. Usually, we assume that all impurities introduced into the Si are ionized, that is, each impurity replaces a Si atom in the crystalline lattice and is ionized to $+1q$ or $-1q$ charge because it fully contributes 1 electron or hole as a mobile carrier to the semiconductor transport system. Mathematically, we express this as $N_d^+ = N_d$ in the case of electron donor impurities and $N_a^- = N_a$ in the case of electron acceptor (hole donor) impurities. Note that electron donors, N_d, are ionized as positive because they give away an electron, and vice versa for electron acceptors.

At low temperatures a phenomenon called carrier freezeout becomes significant. We account for this effect with a model first published by Jaeger et al [1]. Donor and acceptor energy levels of commonly used dopants, such as boron and arsenic, reside very close to the conduction and valance band edges. These levels are labeled with the aid of $\triangle E_d$ and $\triangle E_a$ in figure 5. At room temperature, sufficient thermal energy exists to excite carriers into or



(a) Donor Energy Levels (b) Acceptor Energy Levels

FIG. 5.1. *Donor and acceptor energy levels in the MOS band diagram.*

out of the close-by impurity bands. However, at lower temperatures, and especially at 50K or 100mK, there is not enough thermal energy to excite all donors and acceptors in the system. Only a certain percent are excited, and this percent is characterized again by FD statistics:

$$\begin{aligned} N_d^+ &= \frac{N_d}{1 + 2\exp((E_{Fn} - E_C + \triangle E_d)/(kT))} \\ N_a^- &= \frac{N_a}{1 + 4\exp((E_V - E_{Fp} + \triangle E_a)/(kT))}, \end{aligned} \tag{5.1}$$

where we recognize the general form from (4.1). N_d and N_a are generally known quantities, as they are the doping concentration intentionally implanted in the Si by manufacturers, whereas equations (5.1) are the number of active dopants that we must include in Poisson's equation, (2.1). The coefficients 2 and 4 on the exponential term in the denominator account for degeneracies: a spin degeneracy of 2 in the case of electrons, and both a spin and valance band degeneracy of 2 each for holes.

At room temperature, $\triangle E_d$ and $\triangle E_a$ are taken as constants from experiment. However, they are in general dependent on both carrier concentration and temperature. Historically, these dependencies have been accounted for by this model by Shaheed et al [5],

$$\begin{aligned} \triangle E_d &= E_{d300} - (3.1e-8)N_d^{\frac{1}{3}} + (200T^{-1.0} - 0.66) \\ \triangle E_a &= E_{a300} - (3.037e-8)N_a^{\frac{1}{3}} + (200T^{-0.95} - 0.88), \end{aligned} \tag{5.2}$$

which is a power-law fit to experimental data. $E_{d,a300}$ is the experimental constant specific to the impurity at $300K$, the next term accounts for concentration dependence and the third for temperature dependence.

Since [5] provides well-matched experimental data on sheet-resistance versus temperature down to 60K, we attempted to implement equation (5.2) along with FD statistics and incomplete ionization. However, upon even rudimentary examination we discovered that $\triangle E_d$ and $\triangle E_a$ took on values that we believe are unphysical. In the example table 5.1 below, we have printed the values of $\triangle E_d$ and $\triangle E_a$ resulting from (5.2) at various temperatures, for $N_{d,a} = 1e14/cm^3$, using boron as an example acceptor impurity. Room-temperature values for $E_{d,a300}$ have been taken from Sze [8], as suggested by [5]. Equation (5.2) claims that the

TABLE 5.1
Purported Change in Impurity Level Due to Temperature

	$T = 300$	$T = 200$	$T = 100$	$T = 50$
$\triangle E_d$ (eV)	0.0490	0.3890	1.3890	3.3890
$\triangle E_a$ (eV)	0.0450	0.4669	0.1681	4.0278

change in impurity energy level due to temperature is quite large, and at low temperatures much larger, even, than the band gap of Si. The temperature dependence model gives similarly unbelievable values for higher doping as well. Therefore we have not included this temperature dependence of $\triangle E_{d,a}$ in our simulations.

6. Temperature Dependence of the Band Structure. In addition to the explicit temperature dependencies we have already seen in kT terms, the band structure of Si is also temperature-dependent. The temperature dependencies of electron affinity, χ, band gap, E_g, and electron and hole density of states, N_C and N_V, are modeled after Sze [8] as follows:

$$\chi(T) = \chi(0) - \frac{\alpha T^2}{2(T+\beta)} = \chi(300) + \alpha\left(\frac{300^2}{300+\beta} - \frac{T^2}{2(T+\beta)}\right) \tag{6.1}$$

$$E_g(T) = E_g(0) - \frac{\alpha T^2}{T+\beta} = E_g(300) + \alpha\left(\frac{300^2}{300+\beta} - \frac{T^2}{T+\beta}\right) \tag{6.2}$$

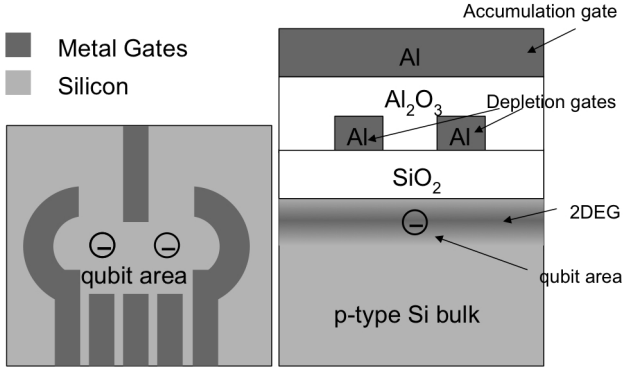
$$\begin{aligned} N_C(T) &= N_C(300)\left(\frac{T}{300}\right)^{\frac{3}{2}} \\ N_V(T) &= N_V(300)\left(\frac{T}{300}\right)^{\frac{3}{2}} \end{aligned} \tag{6.3}$$

where $\chi(300)$, $E_g(300)$, $N_C(300)$, $N_V(300)$, are experimental constants at 300K, taken as default from the Medici TCAD tool manual as $4.17V$, $1.08V$, $2.8e19/cm^3$, and $1.04e19/cm^3$. In the future we will take these defaults from the Sentaurus Manual for better matching.

7. Numerical Results. We implemented the models discussed above in FLOODS, a 3D TCAD solver, which uses tetrahedral meshes and a finite-volume discretization scheme. Finite-volume discretization is necessitated by the continuity equations, which require the Scharfetter-Gummel finite-volume method for stability.

The structures we will discuss are all MOSFET-based structures with p-type Si, a depletion oxide and depletion gate, and an accumulation oxide and accumulation gate (see figure 7.1(a)). The depletion oxide is thin SiO_2, to get a good interface with the Si, as defect-free as possible. The accumulation oxide requires no interface with the Si, so Al_2O_3 is used, which is easier to deposit. The gates are either aluminum, which has a work-function of $4.1V$, or n-type polySi, which we initially model as a metal with work-function difference of $4.16V$. We can also model polySi as heavily-doped Si, but initially we wanted to match the methods used in the Sentaurus Device industry tool for verification of our implementation.

The purpose of the structure in 7.1(a) is to electrostatically (with gate-voltages only) define and control a small dot of charge in the Si. The top accumulation gate attracts or accumulates a 2D electron gas (2DEG) near the Si surface via application of a positive voltage to that gate. Then, a negative voltage is applied to the depletion gates to push away most of this charge, except in a very small region where we wish to confine a quantum dot and control it as a qubit. Figures 7.1(a) and 7.1(b) give an illustration of this structure and indicate where the qubit would reside.



(a) Top Down View of Qubit Gates (b) Cross-Section of Qubit Gates

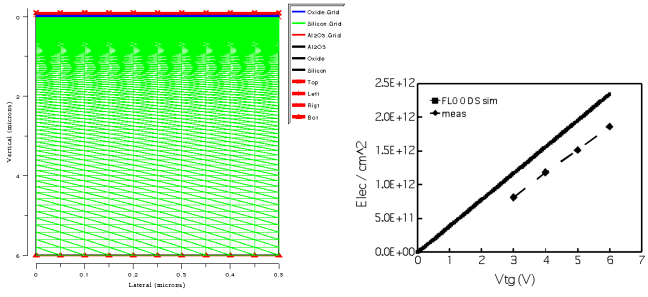
FIG. 7.1. *The MOS gate-topology proposed to electrostatically define a qubit. Charge is drawn to indicate how the charge will be controlled. (a) Top-down view (b) Cross-section view.*

7.1. Capacitor. The first test of our code was to simulate a simple capacitor-like structure simplified from figure 7.1(a) and compare a FLOODS calculation of the capacitance to experimental measurement of the same structure.

Our capacitor structure is an undoped p-type capacitor (n-type MOSFET) $0.5\mu m$ long in y, $5\mu m$ wide in z, and $6\mu m$ deep in x. The deep x dimension is necessary to approximate a bulk-type equilibrium behavior at the ohmic contact at the back. The structure has two gate oxides. The first is a $35nm$ depletion-gate oxide of SiO_2 on the surface of the Si. The second oxide is a $95nm$ accumulation gate oxide of Al_2O_3 on top of that. The accumulation gate on

top is Al, and has no simulated height, as fringe effects will not affect a block structure like this. No depletion gates were included in this structure.

A 2D picture of our structure and grid is in figure 7.2(a) As our structure is undoped, we



(a) 2D Capacitor Grid (b) Inv.-Layer Charge vs. V_g

FIG. 7.2. *(a) 2D grid and structure of MOSFET-like capacitor (b) Inversion-layer charge versus top-gate Voltage in Electrons/cm^2 for simulated structure and measured data. Slope of these curves is capacitance. Capacitance matches.*

chose $N_a = 1e14/cm^3$ in the bulk, with very small ($10nm * 10nm * 20nm$) source/drain-like regions on the left and right of the structure at $z = 0$ and $N_d = 2e18/cm^3$. These source/drain-like regions are necessary for convergence, as they account for boundary decoupling due to depletion region charge.

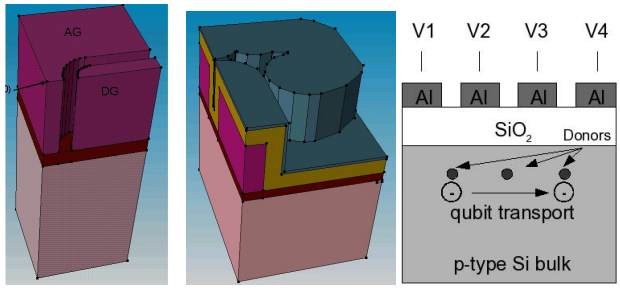
Figure 7.2(b) is a plot of inversion-layer charge versus top-gate voltage for both a measured device and a 2D simulated device. The slope of these curves is the capacitance of the structure, and both match. We believe the 1V offset is due to differences in definitions of material parameters, such as band-gap and electron affinity. Because the capacitances match, we assert that our models appropriately account for the relevant physics of this test structure.

8. Future Work. The capacitor study was important to see if our tools matched experiment. In the future, we would like to use FLOODS to understand and predict how the size of, spacing between, and bias configurations on the gates can be varied to electrostatically define a quantum dot, if possible. We plan to increase the complexity of our simulated structure in 2 stages (figures 8.1(a) and 8.1(b)). These are small representative regions of the total proposed structure, figure 7.1(a).

8.1. Single Poly Structure. A simplified structure of 7.1(a) is shown in figure 8.1(a). This has been named a single-poly structure because both the accumulation and depletion polySi gates are fabricated in one layer.

8.2. Double Poly Structure. Finally, we would like to simulate a double-poly structure, with 2 layers of polySi gates, both accumulation and depletion. We must take care to simulate a structure that has the conformal deposition of the upper layers that will occur during fabrication (figure 8.1(b)).

8.3. CTAP. A related structure is called CTAP, which stands for coherent tunneling adiabatic passage. Illustrated in 8.1(c), this structure has been proposed for the control of qubit transport. Tight-binding calculations on this structure have suggested the existence of a CTAP pathway in such a structure [4]. FLOODS calculations could be used in conjunction with tight-binding methods to provide more representative boundary conditions for electrostatic potential calculations.



(a) Single Poly (b) Double Poly (c) CTAP Structure

FIG. 8.1. *Future Work Structures (a) A simplification of 7.1(a) with polySi accumulation and depletion gates on the same level. (b) A small representative region of 7.1(a) with polySi accumulation and depletion gates. (c) A proposed MOS-based structure for CTAP transport (unrelated to 7.1(a)*

8.4. Poisson-Schroedinger Solvers. Although TCAD solvers like FLOODS are very good at simulating electrostatic potential, regions such as quantum-dot regions must be treated quantum-mechanically, as the charge levels there are below what can be accounted for with classical models. This can be achieved by solving Poisson's equation coupled with Schroedinger's equation, iterating between two different solvers, such as FLOODS and another tool, or performing a full Schroedinger-Poisson solve in FLOODS. The latter involves some challenges, especially because the continuity equations require the Scharfetter-Gummel finite-volume discretization, whereas a Schroedinger-Poisson solve is best accomplished with a finite-element discretization scheme. We will look into these options in the future.

9. Conclusions. In this paper we have outlined models necessary for the TCAD simulation of device behavior in Si MOS-based qubit candidate structures. We have implemented Fermi-Dirac statistics, incomplete ionization, and temperature-dependent band structure models in the FLOODS TCAD package. Our preliminary simulation results on a MOS capacitor structure show good agreement with both the Sentaurus Device industry TCAD solver, as well as experimental measurements. Additional simulations can now be carried out to study proposed structures for the demonstration of the Si qubit. In the future, a full Schroedinger-Poisson solve should be implemented, either by iterating a FLOODS solve with another tool, or by implementing a Schroedinger-Poisson solver in FLOODS.

REFERENCES

[1] R. C. JAEGER AND F. H. GAENSSLEN, *Simulation of impurity freezout through numerical solution of poisson's equation with application to mos device behavior*, IEEE Trans. Electron Devices, 27,5 (1980), pp. 914–920.
[2] W. B. JOYCE AND R. W. DIXON, *Analytic approximations for the fermi energy of an ideal fermi gas*, Applied Physics Letters, 31,5 (1977), pp. 354–356.
[3] M. PINTO, *Comprehensive semiconductor device simulation for silicon ulsi*, (1990), pp. 237–242.
[4] R. RAHMAN, S. H. PARK, J. H. COLE, A. D. GREENTREE, R. P. MULLER, G. KLIMECK, AND L. C. L. HOLLENBURG, *Atomistic simulations of adiabatic coherent electron transport in triple donor systems*, (2009).
[5] M. R. SHAHEED AND C. M. MAZIAR, *A physically based model for carrier freeze-out in si- and si-ge base bipolar transistors suitable for implementation in device simulators*, Bipolar/BiCMOS Circuits and Technology Meeting, (1994), pp. 191–194.
[6] R. K. SMITH AND J. W . M. COUGHRAN, *Computational challenges in simulations of ulsi semiconductor devices*, IEEE Int. Conference on System Sciences, 27 (1994), pp. 7–15.
[7] A. SOMMERFELD, Z. Phys., 47,1 (1928).
[8] S. M. SZE AND K. K. NG, *Physics of semiconductor devices*, (2007).

PARTICLE MESH METHODS FOR PLASMA SIMULATION

MAXX C. KURECZKO* AND DAVID M. DAY†

Abstract. Plasmas are highly ionized gases that are ubiquitous in chemistry and physics. Our interests lie in simulating plasmas with little to no magnetic fields. We implement and test several numerical schemes for stochastic, or so-called "hot" plasma simulations. Based off of some previous work, we analyze methods that conserve energy or momentum. During this process, it became clear that the nature of these discretizations and the general behavior of plasmas make code implementation and verification difficult. Our results were rarely consistent with the established theory. Several revisions and tests were performed to improve code correctness. Preliminary results of this process and our simulation are presented here, while future drafts of this report will present more accurate data.

1. Introduction and summary. Plasmas are highly ionized gases that are ubiquitous in chemistry and physics. Core applications here at Sandia National Laboratories involve plasmas with negligible magnetic fields. In these areas, simulation capabilities are catching up with experiments.

This paper begins with an exposition on plasmas and an outline the Particle-Mesh (PM) numerical methods for modeling plasmas. A variety of Finite-Element methods (FEM) are discussed for the potential equaiton. Numerical results are presented for some one-dimensional model problems.

Particle-Mesh methods are sometimes called "Particle-in-Cell" (PIC) or "Direct Simulation Monte Carlo" methods. These numerical methods involve, among other things, n_p particles at locations $\{x^i\}_{i=1}^{n_p}$ with velocities $\{v^i\}_{i=1}^{n_p}$ and charges $\{q^i\}_{i=1}^{n_p}$. In Section 2, the conservation properties of the plasma equations are derived. Numerical experiments in Section 4 confirm the theory. While [3] is an in-depth source on plasmas, this paper fills in some numerical analysis issues not discussed there.

The ultimate goal of our work is to model hot plasmas. In these plasmas, the standard deviation of the velocities is proportional to the plasma temperature; particle velocities are random variables. Such simulations begin with a set of charged particles uniformly distributed in space having a random velocity assignment from a Maxwellian distribution. Particle dynamics is governed by Newton's Law. In Particle Mesh methods, the electric field driving the particles is computed from the electrostatic potential.

The original goal of this work concerned smoothing. A necessary condition for the stability of PM methods is that, in the discretization of the potential equation, element diameter be less than the Debye length [3]. The Debye length,

$$\lambda_D = \sqrt{\frac{\varepsilon_o k_B T}{nq^2}},$$

depends on Boltzmann's constant, k_B, the absolute temperature, T, the electric constant, ε_o, the particle density n, and the fundamental charge of an electron q [2, 4]. This constraint is a severe limitation for the plasmas of interest. The precise form of the constraint depends on the specific discretization. Essentially, smoothing techniques are discretizations that remain stable with element diameters noticably larger than λ_D.

However, as our work proceeded, we realized that code verification would be challenging for hot plasma simulations, due to their stochastic nature and difficulty in finding a method that involved physically correct scaling of the discrete equations. In particular, a code may inherit the conservation properties of the plasma yet still be unphysical. Standard validation

*New Mexico Institute of Mining and Technology, mkureczk@nmt.edu
†Sandia National Laboratories, dmday@sandia.gov

problems, such as plasma sheaths, are computationally intensive. Instead a"cold" plasma [2] problem ("cold" plasmas are deterministic) was used for code-to-code verification. These fluid equations were solved using a spectral method.

The rest of this paper will begin with an overview of conservation laws and what they imply for plasma simulations in Section 2. The discussion will delve a little deeper in Sections 3 and 3.1, where discretization techniques for the conservations of energy and momentum will be considered. Finally, in Section 4, we will verify and test our simulations and analyze the results.

2. Plasma equations. Discrete methods for partial differential equations are fundamentally variational. For this reason, the plasma equations are given here in their variational form. These equations are derived from the formulation of plasma physics in terms of probability density functions called Vlasov's equation [3]. However, due to space limitations, the description of Vlasov's equation is terse.

The variational equations of interest may be derived from the action integral

$$I[\Phi, \mathbf{x}] = \int_{t_0}^{t_1} \mathcal{L} dt \tag{2.1}$$

for Vlasov's equations (see Section 5.5 of [3] for background), where Φ is the potential at the location $\mathbf{x}$, and $\mathcal{L}$ is the Lagrangian.

In theory, charge density ρ and particle density n are given by

$$\rho(x) = \sum_{i=1}^{n_p} q^i \delta(x - x^i), \quad n(x) = \sum_{i=1}^{n_p} \delta(x - x^i). \tag{2.2}$$

However, in simulations, a numerical particle represents a particle cloud containing an astronomical number of physical particles whose densities are mollified or smoothed out. Also, note for further reference that the electric field is given by

$$E = -\nabla\Phi. \tag{2.3}$$

The plasma equations arise by approximating the probability density function in Vlasov's equation to be a finite sum of delta functions, (one per particle), leading to the Lagrangian

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n_p} m_i |v_i|^2 - q_i \Phi(x_i, t) + \frac{1}{2} \int |\nabla\Phi|^2 dx. \tag{2.4}$$

The Lagrandian is the difference between the kinetic and potential energy plus field energy [3]. Minimizing the functional $I[\Phi, \mathbf{x}]$ gives the equations satisfied by the particles. Variation of (2.1) with respect to position gives Coulomb's Law,

$$m\dot{\mathbf{v}} = qE. \tag{2.5}$$

Equation (2.5) is sometimes called the Momentum Equation or the Lorentz Force equation (in our case, $B = 0$). Furhtermore, variation with respect to potential yields

$$-\varepsilon_o \nabla^2 \Phi = \rho \tag{2.6}$$

The Lagrangian, equation (2.4), gives insight into different discretizations of equations (2.5) and (2.6).

The conservation of the energy of each individual particle,

$$\frac{d}{dt}\left(\frac{1}{2}m_i|v_i|^2 + q_i\Phi(x_i,t)\right) = 0$$

follows from equation (2.5). The Lorentz Force equation, equation (2.5), has the form $dp^i/dt = q^iE$, where p^i is the momentum of the ith particle. Equation (2.5) combined with the charge density, equation (2.2), implies the conservation of momentum as described in Section 6.9 of [4]. The sum of the momenta, p_V, of all the particles in a given volume V satisfies

$$\frac{dp_V}{dt} = \int_V \rho E d^3x. \tag{2.7}$$

3. Galerkin discretizations. Continuous in time, discrete in space discretizations are not studied in [1, 3]. In simulation, these methods do not conserve energy completely, as their continuous-in-time integrations become discretized numerically.

Here we will generalize the idea of energy-conserving methods from [1, 3] to an arbitrary Galerkin-type method. The result is that once a discretization for the Poisson operator is chosen, one and only one discrete charge density conserves energy.

The potential is approximated by an element of a vector space with basis functions. If the basis functions are $\{W_p(\xi)\}_{1\le p\le r}$, one may define the vector-valued function $\mathbf{w}(xi)$ by

$$\mathbf{w}(\xi) = [W_1(\xi), \ldots, W_r(\xi)]$$

The discrete potential $\phi(x)$ at x is a linear combination of the $\{W_p(\xi)\}_{1\le p\le r}$. That is, there exists a vector u of expansion coefficients so that $\phi(x) = u' * w(x)$. The Lagrange equation for the potential is

$$\varepsilon_o K u = \sum_{i=1}^{n_p} q_i w(x_i), \tag{3.1}$$

where K is known as the stiffness matrix.

Here, it is worth discussing Galerkin's method in a general sense. We have obtained the differential equation

$$-\varepsilon_o\nabla^2\Phi = \rho.$$

Multiplying both sides by a test (trial) function $W(x)$ and integrating over the domain gives

$$-\varepsilon_o\int_\Omega \nabla^2\Phi W = \int_\Omega \rho W. \tag{3.2}$$

Introducing some convenient notation, let

$$\langle u, v\rangle = \int u(\xi)v(\xi)d\xi$$

be the inner product of u and v. Also, let

$$B(u,v) = \varepsilon_o\int_\Omega \nabla u\nabla v.$$

Then (3.2) can be written as

$$-\varepsilon_o\langle\nabla^2\Phi, W\rangle = \langle\rho, W\rangle. \tag{3.3}$$

Integration by parts of (3.3) yields

$$\varepsilon_o \langle \nabla\Phi, \nabla W\rangle = \langle \rho, W\rangle = f(W). \tag{3.4}$$

Define $H^1([a,b]) = \left\{ f : \int_a^b |\nabla f|^2 < \infty \right\}$. The objective now is to find $\Phi \in H^1$ such that $\forall W \in H^1$, $B(\Phi, W) = f(W)$. By way of the basis functions W_i,

$$\Phi(x) = \sum_i^{\infty} \alpha_i W_i(x)$$

for some coefficients α_i. The problem has now turned into a linear system. For the FEM, solving $B\left(\Sigma_i W_i \alpha_i, W_j\right) = f(W_j)$ results in the stiffness matrix, and solving $\Sigma_i B(W_i, W_j)\alpha_i = f(W_j)$ gives the right hand side.

The left-hand side in (3.1) is exactly the equation that arises from Galerkin's method, while the right-hand side is not necessarily the right-hand side used in Galerkin's method. One problem is to characterize the basis functions $\{W_p(\xi)\}_{1\le p\le r}$ such that the Galerkin right-hand side is the energy conserving right-hand side. To do this, for energy conservation in the simulations, the quantity

$$\rho_j = \sum_{j=1}^{n_p} W_j(x_j)$$

is computed for all j, as is the density function

$$\rho(x) = \sum_{j=1}^{n_p} \rho_j W_j(x).$$

The k-th component of the right-hand side of our FEM is then computed by

$$\mathrm{rhs}_k = \int W_k(x)\rho(x)dx$$

where $k = 1, 2, ..., L$.

3.1. Momentum conservation. Momentum conservation is discussed extensively in [3] and [4]. However, the discussion of momentum-conserving discretizations in Section 5.3.3 of [3] will make more sense if one first understands Section 6.9 of [4].

In general, the ideas of momentum conservation come from a few assumptions about the problem. If, in a small volume V, the electric field E is constant and the approximate charge density has exact moments

$$\int_V \rho(x)dx = \sum_{x^i \in V} q^i, \tag{3.5}$$

then momentum is conserved within V.

Momentum conservation in [1] or [3] concerns certain geometries and boundary conditions in which p_{part} is constant and refers to discretizations for which the computed p_{part} is also constant.

For momentum conservation in our simulations, $\rho(x)$ is approximated by either a piecewise-constant or continuous piecewise-linear function. That is, $\rho(x)$ is a function based on the density (number of particles/element), interpolated in a piecewise-constant or continuous piecewise-linear fashion.

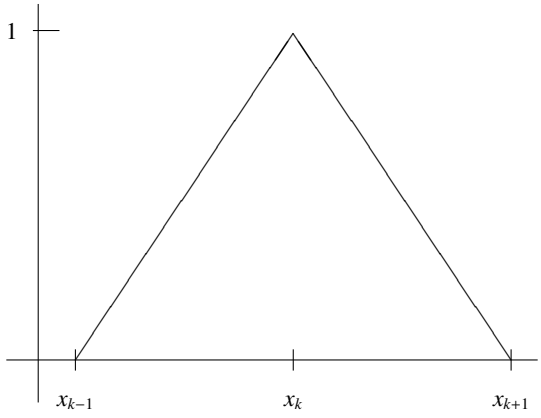


FIG. 4.1. *The k-th basis function,* $W_k(x)$

4. Results. In our computations and MATLAB simulations, we chose our basis functions $W_k(x)$ to be the hat function, as depicted in 4

As a great deal of the experiments involved code verification and correctness, spending a substantial amount of time working out bugs and errors, we have yet to implement valid code and collect accurate data for the momentum conserving scheme. While this paper is under review, we will collect this data and present it in its entirety for the final submission.

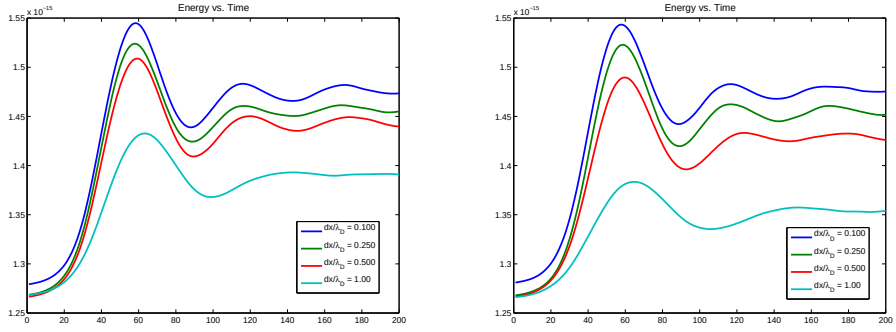
For a uniformly discretized domain, with domain length H, element width dx and L elements, we have that $H = Ldx$. In implementation, there are two methods of running the simulations. The first is holding the length of the entire domain H constant while changing the width dx (and number, L) of the individual elements. These experiments are discussed in Section 4.1. The second implementation involves keeping a constant number of elements while letting H and dx vary. These trials are demonstrated in Section 4.2.

4.1. Constant domain size, varying element width. The first set of numerical experiments ran the hot plasma simulation 1000 times, averaging the results over 200 time steps. All experiments simulated 3168 particles initially distributed uniformly in space ($H = 1.33\times10^{-4}$) with random velocities. Plots of the total energy of the system against time can be seen in Figure 4.2(a) for varying values of dx/λ_D using the energy conserving scheme. Figure 4.2(b) shows the total energy against time using a regular FEM for comparison. The percent change in energy (E) and momentum (P) were computed from the initial to final times and are shown in Table 4.1.

In these trials, the FEM conserved energy slightly better than the energy conserving method. It is also interesting to note that as the resolution becomes more coarse, energy is conserved more for both methods. The values for the percent change in momentum were included to demonstrate its stochastic nature (dependence on the random velocity). As a result, the percent change is not a good metric for momentum. Figures 4.3(a) and 4.3(b) show the stochastic evolution of momentum over time for the energy-conserving scheme and FEM, respectively ($dx/\lambda_D = 0.5$).

To see how the percent change in total energy behaves as a function of dx/λ_D while holding H constant for both the energy conserving scheme and FEM, see Figure 4.1

To see just how well energy is conserved for fewer and wider elements with a constant domain size, we tested values of dx/λ_D larger than one. Plots of the results can be seen in Figures 4.4(a) and 4.4(b) and percent changes in energy are in Table 4.2.



(a) Total energy against time using the energy conserving scheme.

(b) Total energy against time with the FEM.

FIG. 4.2. *Plots of the total energy vs time for varying dx/λ_D values, holding the total length of the domain constant letting the element size vary.* $H = 1.33 \times 10^{-4}$

TABLE 4.1
Percent change in energy and momentum for constant values of H.

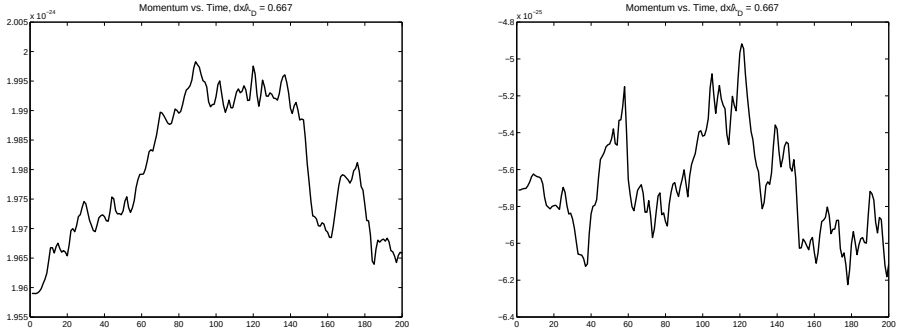
		Energy Conserving		FEM	
dx/λ_D	L	% Change E	% Change P	% Change E	% Change P
0.100	80	15.18	0.04	15.17	5.32
0.160	50	15.02	38.59	14.92	0.86
0.200	40	14.89	0.60	14.92	72.75
0.250	32	14.71	3.63	14.48	6.49
0.333	24	14.63	0.57	13.72	4.60
0.400	20	14.44	1.46	13.55	3.99
0.500	16	13.67	0.35	12.57	7.05
0.600	12	12.55	2.35	10.94	11.25
0.800	10	11.46	3.52	9.51	33.63
1.000	8	9.66	0.70	6.92	157.51

Again, the standard FEM conserves total energy slightly better than the energy conserving scheme, and as the element width increases, there is less change in total energy. This decreasing change in total energy is a result of the continuous in time and discrete in space integration. The fewer elements there are, the less errors accumulate over time, resulting in a better energy profile.

4.2. Varying domain size, constant element width. To further test the energy conserving scheme, the next set of experiments held the number of discretizations constant ($L = 10$), and the domain size H and element width (dx) were allowed to change with respect to the Debye length. Trails were run with 3168 particles for 200 times steps 1000 times and averaged over time.

This setup differs from that in Section 4.1 in that there are spatial differences in how the particles are distributed. A larger domain results in the same number of particles per element, but spread out farther, initially. Computationally, the density in the element is the same, but interactions between particles become smaller. Table 4.3 gives the percent change in energy and momentum for this set of experiments.

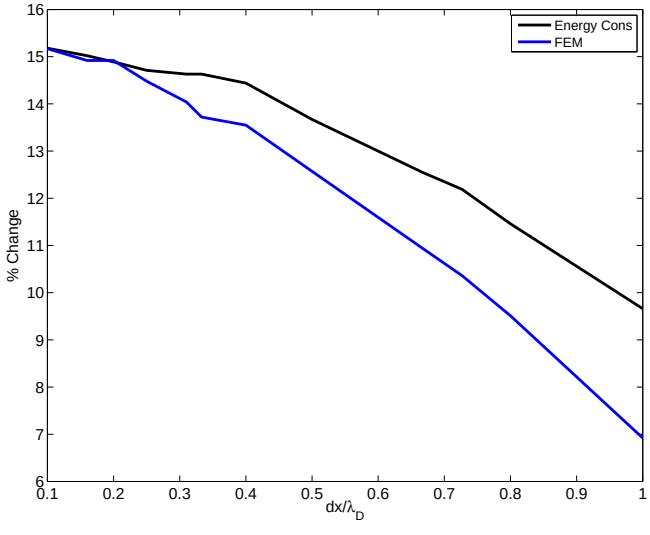
We note that for smaller values of dx/λ_D, energy is well conserved; as the element size



(a) Momentum against time with the energy conserving scheme.

(b) Momentum against time with the FEM.

FIG. 4.3. *Plots of momentum vs time for $dx/\lambda_D = 0.5$, holding the total length of the domain constant letting the element size vary. The stochastic nature of the momentum, at least initially, can be seen here.*



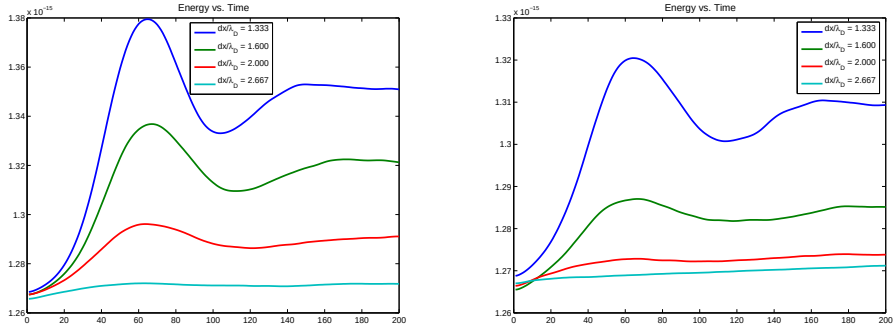
approaches the Debye length, there is a dramatic increase in energy. Comparisons of the total energy in time can be seen in Figures 4.5(a) and 4.5(b). It is also worth pointing out that the FEM again seems to conserve energy slightly better and that the percent change in momentum is again an inaccurate metric of conservation. To see the percent change, see Figure 4.2.

It is important to point out, again, that the system's momentum still behaves randomly, especially early on in the simulation. Here, we include plots of the momentum for $dx/\lambda_D = 0.5$ to illustrate this idea. Figure 4.6(a) uses the energy conserving method, while Figure 4.6(b) uses the regular FEM.

5. Conclusions. We have written, implemented, and tested numerical schemes for plasma simulations. Based off of previous work, we analyzed methods that, in theory, conserve energy and momentum. However, the nature of these discretizations and the behavior of plasmas made implementation difficult and often produced unrealistic results. The implemented

TABLE 4.2
Percent change in total energy for constant values of H, with dx/λ_D larger than 1.

		Energy Conserving	FEM
dx/λ_D	L	% Change E	% Change E
1.333	6	6.50	3.20
1.600	5	4.25	1.55
2.000	4	1.86	0.59
2.667	3	0.48	0.33



(a) Energy against time with the energy conserving scheme.

(b) Energy against time with the FEM.

FIG. 4.4. *Plots of total energy vs time for $dx/\lambda_D > 1$, holding the total length of the domain constant letting the element size exceed the Debye Length.*

energy-conserving method seemed to not work as well at conserving energy as the regular FEM did. We also saw very different energy profiles for how we interpreted the relationship among the domain, its elements, and the Debye length.

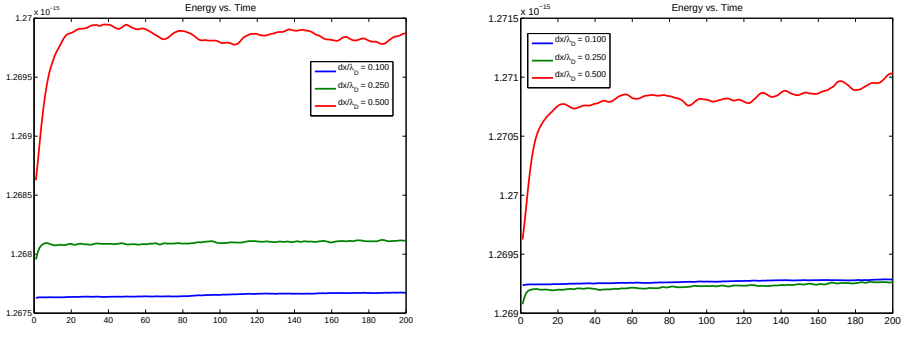
After several revisions and trials, we now have a more accurate and realistic code. While this paper was being drafted, several bugs and errors returned results that did a poor job at conserving energy and an even worse job at conserving momentum.

5.1. Future Work. We would like to stress that while this paper is under review, we will collect better, more accurate data. The momentum conserving scheme now works. However, due to time constraints, this data, and better energy conserving data, could not be included in this draft. The final revision will include these numerics.

In terms of actual future work, being able to expand these ideas and simulation into two or three dimensions would be beneficial. The simplifications and assumptions we made in the one-dimensional model would have to be worked out, however.

TABLE 4.3
Percent change in energy and momentum for a constant value of $L = 10$, H allowed to vary.

	Energy Conserving		FEM	
dx/λ_D	% Change E	% Change P	% Change E	% Change P
0.100	0.0037	0.0801	0.0040	0.1830
0.160	0.0065	0.5866	0.0058	1.3739
0.200	0.0077	2.4575	0.0096	0.5892
0.250	0.0125	0.0439	0.0148	1.878
0.333	0.0235	13.03	0.0307	0.1204
0.400	0.0432	1.752	0.0493	4.483
0.500	0.0983	0.691	0.1106	0.4843
0.667	0.9004	0.222	0.7374	0.1914
0.800	11.603	2.742	9.421	2.857
1.000	68.25	6.98	58.22	5.166



(a) Energy against time with the energy conserving scheme.

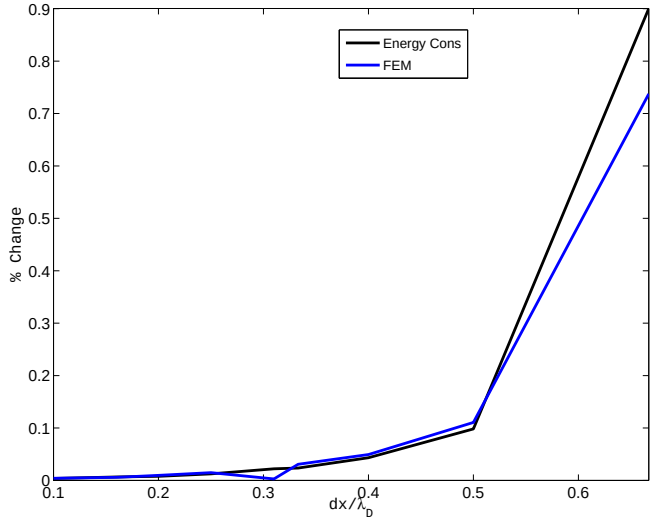
(b) Energy against time with the FEM.

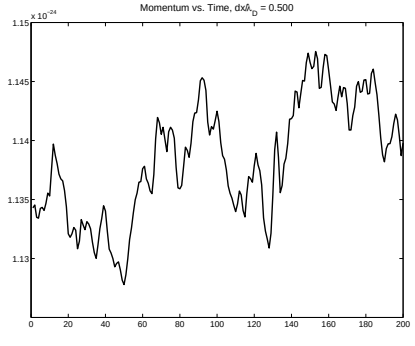
FIG. 4.5. *Plots of total energy in time with the energy conserving scheme and FEM, both holding a constant number of elements, letting the element width and domain size vary.*

6. Acknowledgements. The authors would like to thank Tom Hughes of Sandia National Laboratories for his help, insight, and recommendation of [5].

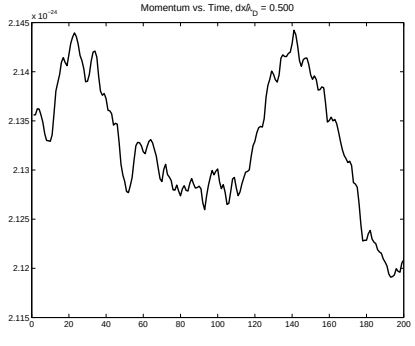
REFERENCES

[1] C. K. BIRDSALL AND A. B. LANGDON, *Plamsa Physics Via Computer Simulation*, McGraw-Hill, 1985.
[2] R. FITZPATRICK, *Introduction to plasma physics*. `http://farside.ph.utexas.edu/teaching/plasma/lectures/lectures.html`.
[3] R. W. HOCKNEY AND J. W. EASTWOOD, *Computer Simulation Using Particles*, Taylor & Francis Group, 1981.
[4] J. D. JACKSON, *Classical Electrodynamics*, John Wiley & Sons, Inc., 1963.
[5] H. OKUDA, *Nonphysical instabilities in plasma simulation due to small $\lambda_d/\delta x$*, Fourth Conference on the Numerical Simulation of Plasma, (1970), pp. 511 – 525.





(a) Momentum against time with the energy conserving scheme.



(b) Momentum against time with the FEM.

FIG. 4.6. *Plots of momentum vs. time for* $dx/\lambda_D = 0.5$*, holding the number of elements constant and letting the domain and element size vary.*

INTERFACE RECONSTRUCTION VERIFICATION IN ALEGRA

M.S. SWAN*, W.J. RIDER†, AND O.E. STRACK‡

Abstract. Lagrangian codes automatically track interfaces between materials but cannot handle large deformations. Arbitrary Lagrangian-Eulerian (ALE) codes are better able to simulate large deformation of materials, but do not automatically track interfaces[4]. During each timestep of an ALE simulation there is a Lagrangian step followed by a remap step that, once the nodal and elemental values have been advected, places these data back onto the Eulerian grid. When more than one material is present, it is imperative to accurately model the boundary of each material so as to properly simulate interactions between them. As can be expected, different methods perform with different orders of accuracies and require different amounts of computational effort. Interface reconstruction verification is used to establish what those orders of convergence are, to determine the acceptable behavior of each method, and to ensure that the code developers are informed when any test deviates from the accepted normal behavior.

1. Introduction. In simulations, interface reconstruction algorithms are responsible for tracking the boundaries between bodies as those bodies move through a mesh. As the simulation framework does the computations on the mesh, there needs to be a method to determine which nodes on that mesh contain which data and what those data are. The point of better interface reconstruction methods is to more accurately represent the movement of material through the mesh while maintaining the shape of the body[7].

The purpose of this research is to benchmark seven interface reconstruction algorithms: Simple Line Interface Calculation (SLIC)[5], Sandia Modified Youngs' Reconstruction Algorithm (SMYRA), new SMYRA[1], and Pattern Interface Reconstruction (PIR)[3]. PIR has four variants that are differentiated by the method of smoothing that the algorithm applies. The options for smoothing are: *off*, *linear*, *spherical*, and *on*. The smoothing option *on* uses a mixture of linear and spherical smoothing. For simplicity, this document only presents data for the options *off* and *on* of the four smoothing options for PIR. In the benchmarks, all options will be tested for convergence.

These algorithms are all included in the Sandia-developed solid dynamics code ALEGRA as options for interface reconstruction. The verification tests discussed here will be added to the ALEGRA benchmark repository. This is to ensure that the simulation framework behaves as expected and to detect changes that would affect the accuracy of the reconstruction algorithms. When a change is detected in the nightly builds, the developers are informed so the cause of the change can be determined. They can then decide whether to fix whatever caused the change or to rebaseline the verification tests to accept the new behavior as the new accepted standard.

Each of the above stated seven methods has a bias, a tendency to advect stronger along the coordinate axes, in the way in which it tracks the interface. The lower the order of convergence, the more bias the method will exhibit. These lower-order methods are diffusive in nature and can be used to dampen numerical oscillations that might occur while using a higher-order, local maxima- and minima-preserving method[6]. Even though the lower-order methods are less accurate they still fill a needed role in the simulation.

2. Method. The method chosen to verify the interface reconstruction algorithms involves making a simple test simulation that has an analytical or reference solution against which the computed solution can be compared. Typical simulations for verifying interface reconstruction methods usually include simple translations and solid body rotations of regular shapes[9]. The basis of verification tests is that upon mesh and timestep refinement the

*University of Utah, scot.swan@gmail.com
†Sandia National Laboratories, wjrider@sandia.gov
‡Sandia National Laboratories, oestrac@sandia.gov

solution will converge.

The primary variable used to determine the error is the volume fraction of each element. The volume fraction is a representation of how full the element is of a particular material. If an element is totally occupied by that material, the volume fraction would be unity, or if the element does not contain any of that material the volume fraction would be zero[1]. For each element in the simulation, the expected volume fraction for that element is subtracted from the simulated volume fraction. The error of a simulation, using the L_1 norm of the volume fraction, is determined by the sum of the absolute values of those differences in volume fractions with that sum being divided by the number of elements in the mesh. The following equation is the mathematical representation of how to determine the error in a simulation[8]:

$$error = \frac{\Sigma_{i=1}^{n}\Sigma_{j=1}^{m}\left|A_{ij} - A_{ij}^{*}\right|}{n \cdot m} \tag{2.1}$$

where A_{ij} is a $n \times m$ 2D array of volume fractions from the final timestep of the simulation and A_{ij}^{*} is a $n \times m$ 2D array of the reference solution. The above equation will only hold for regular cartesian coordinate systems. Once error data is collected from several simulations at varying mesh and timestep refinements, the error vs. mesh refinement data can be plotted in logarithmic space as a linear function. The slope of the linear function is the order of convergence for the method.

As the mesh is refined, a corresponding refinement in timestep is needed to ensure that the material cannot have a displacement that exceeds the width of the elements. The Courant number or Courant-Friedrichs-Lewy (CFL) condition defines the upper limit of the ratio of timestep to mesh refinement[2]. The Courant number (denoted as C) can be modeled in 1D by the following equation:

$$C > \frac{|u| \cdot \Delta t}{\Delta x} \tag{2.2}$$

Where u is the velocity, Δt is the timestep, and Δx is the element size. Adherence to this criteria ensures the mathematical stability of the simulation. The CFL condition is almost always less than unity and is frequently around one half. As the Courant number approaches zero the simulation becomes more taxing on the interface reconstruction algorithm due to the number of cycles and the accumulation of error. In order to check for convergence, the CFL condition must be the same for each simulation at every mesh resolution. For the simulations in this battery of tests, a Courant number of one half is used.

The error of each simulation is in small part linked to mesh geometry. For the final simulations that are to be added as regular benchmarks in ALEGRA the mesh correlates perfectly with the initial position of the square. This ensures that the initial volume fraction values of either unity or zero. For completeness, the simulations were run with irregular mesh refinements such that the initial condition had volume fractions that were not unity or zero to verify that each method has a general trend of convergence. As the mesh is gradually refined for every simulation, the local error is minimized when the mesh precisely lines up with the material being simulated. Thus, the error can be shown to have a periodic nature corresponding to mesh refinement (see figure 2.1).

3. Tests. In the process of testing the interface reconstruction methods, five tests were run to fully verify that the algorithms are well behaved. Two tests were developed for the purpose of thoroughly testing the different algorithms in as few simulations as possible, while the other three tests were used to verify that in the most simple cases the methods worked as expected.

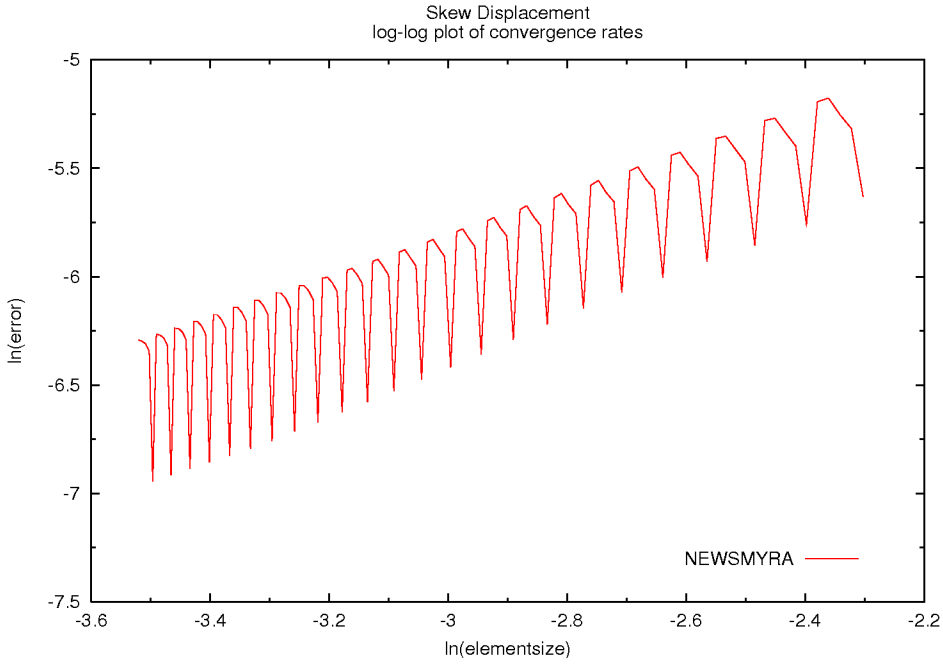


FIG. 2.1. *95 simulations at varying mesh refinements using NEW SMYRA to demonstrate periodicity of error associated with mesh refinement. The local minima occur when the mesh exactly lines up with the initial position of the material being simulated. Note that the amplitude of the error oscillation is constant in log-log space. This translates into exponential decay in the magnitude of error with mesh refinement.*

The five tests referred to above are vertical displacement, horizontal displacement, 45 degree displacement, skew ($\approx 26.5°$) displacement, and 90 degree rotation; However,he two tests to be checked into the repository are the skew displacement and 90 degree rotation simulations. Each of these five tests were carried out in the positive and negative directions to verify that the reconstruction algorithms would all behave as expected in every direction on the 2D plane. For example, the vertical displacement test was run with translation of a block in the positive Y-direction and in the negative Y-direction, and compared against each other to check for consistency.

Each of the two tests that will be added to the benchmark repository will be set up in such a way that the block will be translated (or rotated) so that the first timestep and the last timestep should be equal. This method was chosen so that the reference solution will be easily attainable for any simulation layout that could be desired, so long as the translated block ends where it started. This method, once the necessary files were made, was easily adapted to the other reconstruction algorithms and the other translation simulations.

Skew Displacement. The simulation that involved skew displacement was included in the verification suite because it reveals weaknesses that exist in rigid translation in the interface reconstruction algorithms. The non-ideal (no particular plane of symmetry) angle of displacement made this test an excellent choice to verify the greatest possible number of translational situations with the least number of tests and with the lowest requirement of computational effort.

This test, as depicted in figure 3.1, involves translation up and to the right along a trajectory of ≈ 26.5 degrees with a prescribed velocity field with a magnitude that sinusoidally

FIG. 3.1. *Images of the volume fraction from a skew translation simulation using SMYRA as the method for interface reconstruction. Notice that the error accumulates near the corners even under non-rotational displacement.*

varies with time. The simulation is run for one full cycle so that the block is translated up and to the right for the first half of the cycle and is returned along that same path during the second half of the cycle. This is done for simplicity in error calculations (as discussed in section 4).

90 Degree Rotation. Unlike the skew displacement test, the 90 degree rotation test has a prescribed velocity field that varies with position rather than time. This means that every node in the mesh will have a different velocity instead of the entire mesh having a constant velocity field that would advect the block uniformly in a given direction. This simulation will be more demanding on the interface reconstruction algorithm due to the varying flux values across the mesh.

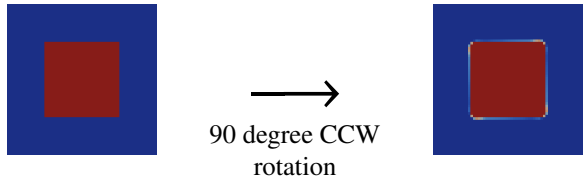


FIG. 3.2. *Images of the volume fraction from a rotational displacement simulation using SMYRA. Notice that the primary concentrations of error are on the leading edges near the corners.*

The rotation of the block is about the centroid with no lateral or vertical translation. The exclusion of rigid body translation from this rotation problem is justified by the previous skew displacement simulation that tests the convergence of the methods for strict rigid body translation.

Although this test does not apply a 90 degree counter-clockwise rotation and then apply a 90 degree clockwise rotation to return the material to the initial position, the ending timestep and the beginning timestep are comparable. This takes advantage of the planes of symmetry of the square that is being rotated.

4. Analysis. The output files from the simulations are the basis of the analysis to determine the order of convergence. The analysis for the included error plots(figures 4.2 and 4.3) were done using a python script that calculated the L_1 norm according to equation 2.1 and a linear regression python script that calculated the least-squares line fit for the calculated data. While these scripts were efficient at plotting and analyzing the data, they were difficult to use and time consuming to rebaseline as a benchmark test. For this reason, the benchmark tests use the analysis tools 'vcomp' and 'vdiff' that come bundled with ALEGRA.

After the simulations have been run to completion, vcomp (verification comparison tool) analyzes each computed solution against the associated reference solution for each simulation and determines an order of convergence for the set for the L_1, L_2, and L_∞ norms. The list of

computed solutions and reference solutions are all contained in the vcomp input file along with the timesteps at which to perform the analysis. The output of vcomp is contained in an output file, which holds all of the results generated by vcomp. This output file is then passed to vdiff.

The program vdiff (verification differencing tool) is used to compare two output files (or two different timesteps within the same output file). The variables to be compared are defined, along with the error tolerances for each variable, in a vdiff input file. For the reconstruction verification suite, we are interested in comparing the vcomp output against the baseline output to check for variations in convergence rates and error values.

One additional reason for using vcomp and vdiff is the ability to do a batch rebaseline of the tests when using ALEGRA's testing application *testAlegra*. The use of these programs also brings the verification suite into uniformity with the other benchmarks and verification tests that are already in use.

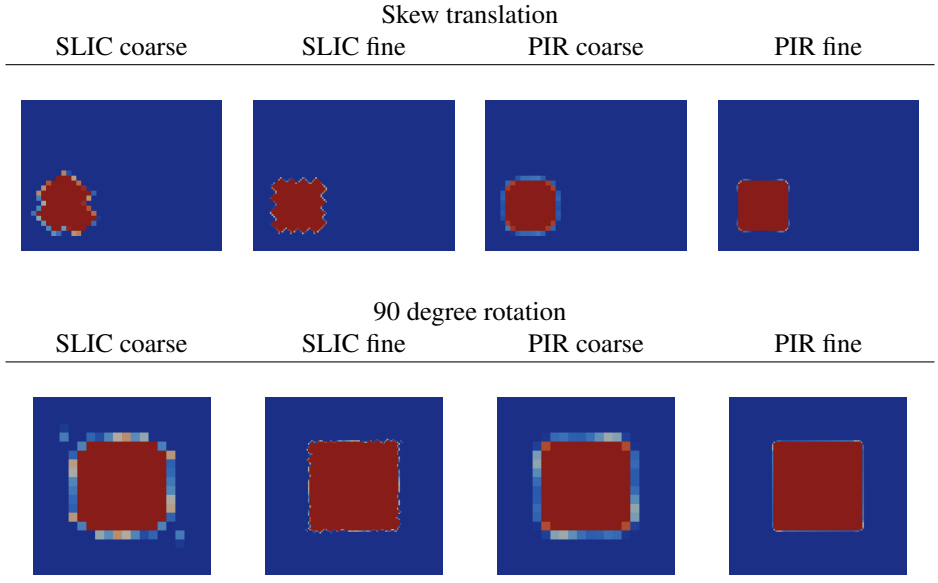


FIG. 4.1. *Images of Final Timesteps for Select Simulations*

A visual comparison of results for the diagonal displacement and 90 degree rotation between SLIC and PIR can be found in figure 4.1. The comparison clearly portrays the inability of SLIC to correctly track the boundaries of the box, particularly in the diagonal displacement simulation at either of the mesh refinements. It is interesting to note that SLIC had a higher convergence rate than the other six reconstruction methods when only lateral or vertical translation of the box was involved. It can readily be determined that while very effective at 1D interface tracking, SLIC's ability to properly model an interface in 2D is greatly lacking at any level of mesh refinement. Upon visual inspection it is apparent that PIR translates and rotates the block well and the block retains its shape with comparatively minimal rounding on corners. Currently, PIR is the highest order interface reconstruction algorithm available in ALEGRA.

While the plots in figure 4.2 and the data in table 4.1 are very insightful and in a general sense represent the expected trend, the higher order methods are converging at markedly lower rates than expected. It can readily be seen that the concentrations of error for these simulation are at the corners of the square for both the skew translation and the rotation tests.

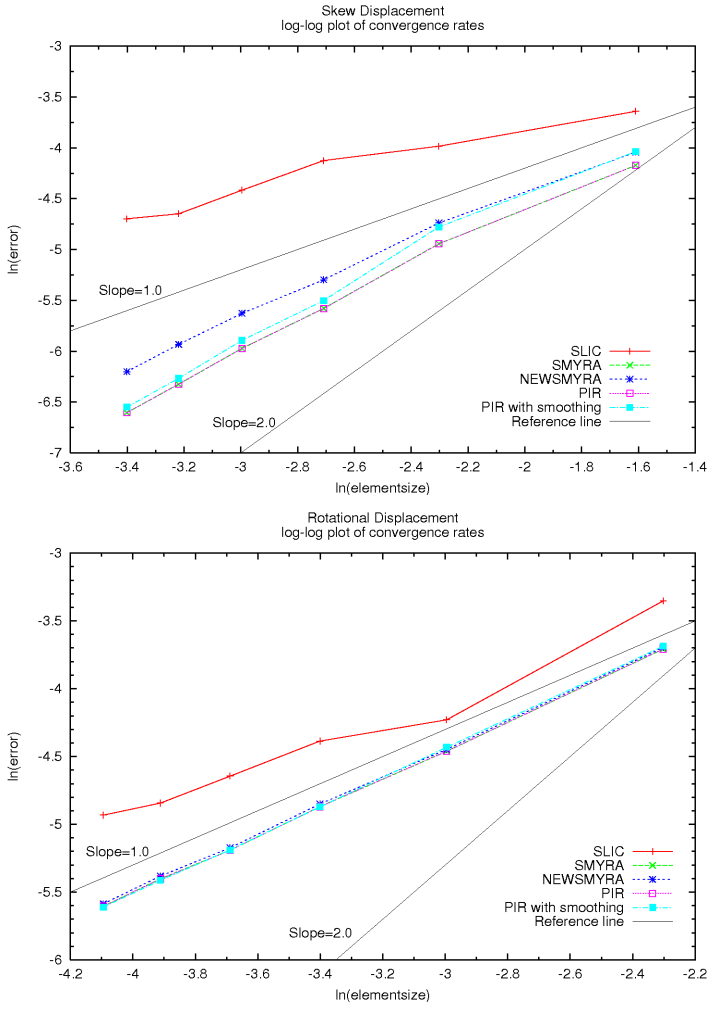


FIG. 4.2. *log-log convergence plots for skew and rotational displacement.*

In an attempt to attain the expected convergence rates the box will be replaced with a circle for the skew displacement test. This test yields results that are more in line with the expected convergence rates (see figure 4.3 and table 4.2).

The differences in the plots for the translation of the square and the circle demonstrates the strengths and weaknesses of the different methods. For the square translation (either rotation or skew translation) all by SLIC perform with convergence rates greater than one. This is noticeably different for those same methods when advecting a circle. It can be concluded that SMYRA, NEW SMYRA, and PIR without smoothing are better fit for tracking flat interfaces, while PIR with smoothing performs over an order of magnitude better in convergence rate than the others.

5. Conclusion. The interface reconstruction algorithms used by ALEGRA are an integral part of the simulation framework as they define spatial extent and shape of simulated materials. The position of the interface is also used in determining which contact algorithms should be used when one material comes into contact with another or when a boundary con-

Convergence Rates		
	SKEW	ROTATION
SLIC	0.52	0.91
SMYRA	1.30	1.06
NEW SMYRA	1.15	1.06
PIR smoothing = off	1.30	1.06
PIR smoothing = on	1.35	1.08

TABLE 4.1

Convergence rates for 5 of the 7 interface reconstruction methods for skew and rotational displacement of a square. Compare with table 4.2.

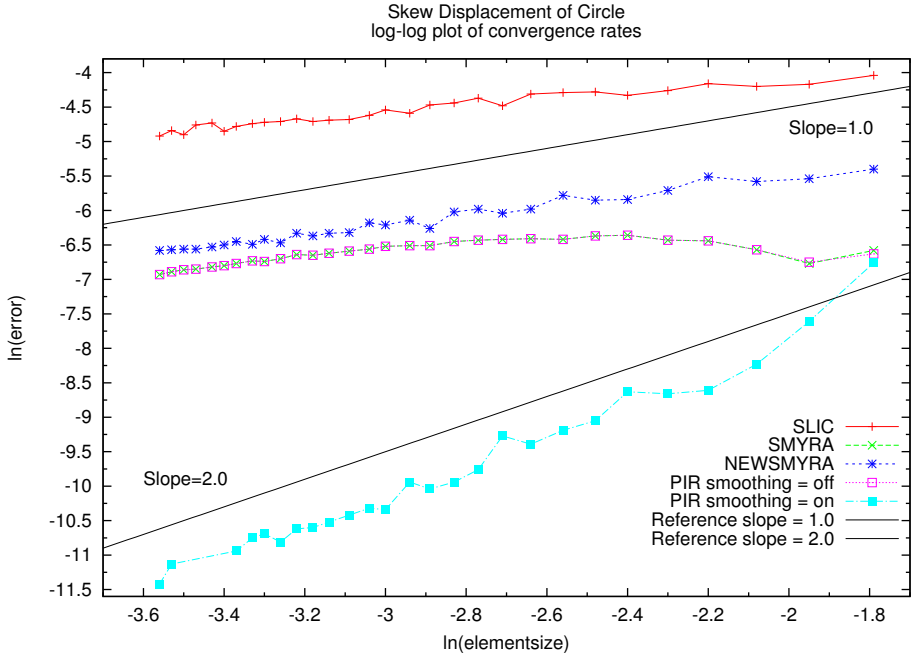


FIG. 4.3. *Convergence test for a circle advected in the skew displacement manner. Due to the lack of corners, this test yields a much greater spread in convergence rates for the different methods. While PIR with smoothing has orders of magnitude less error at fine mesh resolutions, it is less robust. At ln(elementsize)* ≈ -3.5*, several datapoints were not generated due to a lack of robustness in the smoothing algorithm (these issues are currently being addressed). For convergence rates, see table 4.2.*

dition prescribes a contact algorithm. As these methods are called for each timestep and for each element, even a slight increase in accuracy in the reconstruction methods will notably affect the accuracy of the simulation as a whole. This demonstrates the need to be sensitive to changes of the convergence rates, the limitations, and the abilities of each of the reconstruction methods. For this reason, the suite of interface reconstruction verification tests has been developed to ensure that this integral part of ALEGRA performs consistently and accurately across platforms and in different situations.

Convergence Rates	
SLIC	0.46
SMYRA	0.69
NEW SMYRA	0.68
PIR smoothing = off	0.69
PIR smoothing = on	1.92

TABLE 4.2

Convergence rates for 5 of the 7 interface reconstruction methods for skew displacement of a circle (see figure 4.3). Because the data from the coarser simulations was not in the asymptotic regime, the linear regression only utilized the lower half of the data points (for more consistent data) in a least-squares linear fit. The grouping of convergence rates around 0.70 was surprising as those methods generally perform with a convergence rate of at least one (see figure 4.2).

[1] R. BELL AND E. HERTEL, *An improved material interface reconstruction algorithm for Eulerian codes*, SAND92-1716C, (1992).

[2] R. COURANT, K. FRIEDRICHS, AND H.LEWY, *On the partial difference equations of mathematical physics*, IBM Journal, (1967), pp. 215–234.

[3] J. DOLBOW, S. MOSSO, J. ROBBINS, AND T. VOTH, *Coupling volume-of-fluid based interface reconstructions with the extended finite element method*, Computer Methods in Applied Mechanics and Engineering, 197 (2008), pp. 439–447.

[4] R. V. GARIMELLA, V. DYADECHKO, B. K. SWARTZ, AND M. J. SHASKOV, *Interface reconstruction in multi-fluid flow simulations*, Mathematical Modeling and Analysis, (2005).

[5] W. NOH AND P. WOODWARD, *SLIC:(Simple Line Interface Calculation)*, Lecture Notes in Physics, 59 (1976), pp. 330–340.

[6] A. OLIVEIRA AND A. B. FORTUNATO, *Toward an oscillation-free, mass conservative, Eulerian-Lagrangian transport model*, J. Comput. Phys., 183 (2002), pp. 142–164.

[7] W. J. RIDER AND D. B. KOTHE, *Reconstructing volume tracking*, Journal of Computational Physics, 141 (1997), pp. 112–152.

[8] M. RUDMAN, *Volume-tracking methods for interfacial flow calculations*, International Journal for Numerical Methods in Fluids, 2 (1997), pp. 671–691.

[9] R. SCARDOVELLI AND S. ZALESKI, *Interface reconstruction with least-square fit and split Eulerian-Lagrangian advection*, International Journal for Numerical Methods in Fluids, 41 (2003), pp. 251–274.

MODELING A RESONANT TUNNELING DIODE USING TRILINOS

ANNE S. COSTOLANSKI* AND ANDREW G. SALINGER†

Abstract. A method for modeling the performance of a resonant tunneling diode (RTD) using Sandia's Trilinos software is described. The equations to model the behavior of an RTD are given, along with their corresponding numerical approximations. An object-oriented structure using C++ classes is defined, and the method for incorporating Trilinos' nonlinear solver and continuation packages is detailed.

1. Introduction. Resonant tunneling diodes (RTDs) are nanoscale semiconductor devices which were first proposed by Tsu and Esaki in 1973[8]. They predicted a phenomenon known as negative differential resistance, in which the current through a tunneling barrier reaches a local maximum when the injected carriers achieve certain resonant energies. In 1974, Chang et al. fabricated the first RTD device that demonstrated evidence of negative differential resistance[2]. A decade later, in 1983, Sollner et al. improved upon previously achieved results[7], and research on RTDs increased.

Resonant tunneling diodes have two distinct features that make them stand out from other semi-conductor devices: their high speed operation and their ability to produce negative differential resistance. Since tunneling is a very fast phenomenon, the RTD is among the fastest devices ever made. It can be shown by theoretical analysis that the time taken to switch from its current peak to valley, or visa versa, can be less than 1 ps. RTDs have been demonstrated to detect radiation up to 2.4 THz, can generate 700 GHz signals, and have a maximum operational oscillation frequency projected to be over 1 THz [5]. Due to these features and the potential to be used in high speed devices, RTDs have been studied extensively since the mid 1980s [9, 10].

In this paper, we will propose a new method for simulating the performance of an RTD using C++ and Trilinos. The paper is organized as follows: Section 2 describes the basic structure of an RTD, section 3 details the equations that model an RTD, and section 4 specifies how these equations are discretized. Section 5 discusses how to implement the discretized equations, with section 6 listing the details of the C++ classes and section 7 the specific Trilinos implementation. Our conclusions are listed in section 8.

2. Resonant Tunneling Diodes. A typical resonant tunneling diode is composed of the following sections (see figure 2.1):

- Two heavily doped, narrow energy band gap material regions on either end of the device
- Several layers (barriers) of a larger energy band gap material on the interior of the device, separated from the doped regions by thin undoped spacer regions
- Quantum well region(s) separating the barrier regions from each other

The doped regions on either end of the device are generally large in size in comparison to the barrier, spacer, and well regions. In a typical RTD, the quantum well thickness is around 50Å, and the barrier layers can range from 15 to 50Å. The well, barrier layers, and the spacer regions (between the heavily doped narrow energy gap material and the barrier layer) have a significantly lower doping level than the heavily doped regions on each end of the device. These spacer regions ensure that dopants do not diffuse to the barrier layers. Bias is then applied across the device to induce current flow.

*North Carolina State University, ascostol@ncsu.edu.
†Sandia National Laboratories, agsalin@sandia.gov

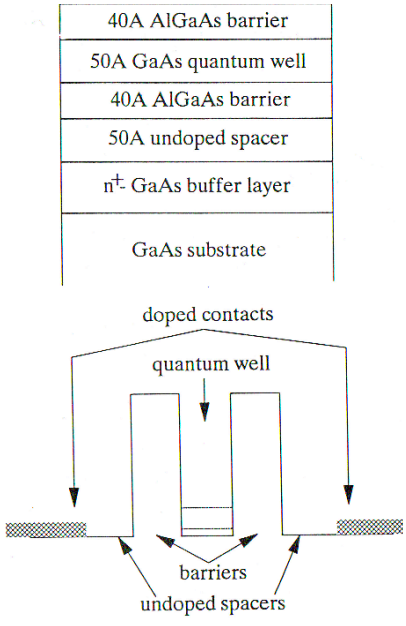


FIG. 2.1. *Structure of a typical resonant tunneling diode. The upper picture provides the physical layout of the materials used in the device, and the bottom picture is the energy diagram of the device. [5]*

3. The Wigner-Poisson Equations. Since an RTD is built on the nanoscale and is governed by quantum mechanics rather than classical physics, the standard drift-diffusion model no longer applies in calculating the current in the device. One of the primary models used for simulating the performance of a resonant tunneling diode is the Wigner-Poisson formulation, which couples the Wigner equation with Poisson's equation to predict the behavior of nanoscale semiconductor devices. For the complete derivation of the Wigner-Poisson formulation, see [1] for details.

The Wigner equation is defined as

$$\frac{\partial f(x,k,t)}{\partial t} = K(f) + P(f) + S(f) \tag{3.1}$$

where $f(x,k,t)$ is the Wigner distribution function that describes the distribution of electrons within the device. It is a function of x, the position along the device, which runs from 0 to L (the length of the device); k, the momentum variable, which runs from $-\infty$ to $+\infty$; and time t. We will focus on the steady state solution, so the time dependence will be dropped.

The first term on the right side of the equation, $K(f)$, is the kinetic energy term and corresponds to the effects due to kinetic energy on the distribution function f. It is given by

$$K(f) = -\frac{hk}{2\pi m^*}\frac{\partial f}{\partial x}. \tag{3.2}$$

Here h is Planck's constant and m^* is the effective mass of an electron.

The second term $P(f)$ is the potential energy term, and is defined as

$$P(f) = -\frac{4}{h}\int_{-\infty}^{\infty} f(x,k')T(x,k-k')dk' \tag{3.3}$$

with

$$T(x,z) = \int_0^{\infty} [U(x+y) - U(x-y)]\sin(2yz)dy \tag{3.4}$$

where $U(x)$ is the potential energy inside the device. $U(x)$ can be written as

$$U(x) = \Delta_c(x) + u_p(x) \tag{3.5}$$

where $\Delta_c(x)$ represents the energy band function defined by the barriers and wells within the device and $u_p(x)$ represents the solution to Poisson's equation,

$$\frac{d^2u_p(x)}{dx^2} = \frac{q^2}{\epsilon}\left[N_d(x) - n(x)\right]. \tag{3.6}$$

Here q is the charge on a electron, ϵ is the dielectric constant (which is material dependent), $N_d(x)$ is the doping profile of the device, and $n(x)$ is the electron density function, which is defined as

$$n(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x,k)dk. \tag{3.7}$$

The boundary conditions for Poisson's equation are

$$u_p(0) = V_0, \qquad u_p(L) = V_L \tag{3.8}$$

where V_0 is the initial voltage at the left side of the device, and V_L is the amount of bias applied across the device. Traditionally $V_0 = 0$ and $V_L = -V$ with $V \geq 0$.

The third term in the Wigner equation, the scattering term $S(f)$, accounts for collision interactions between electrons in the device, and is defined as

$$S(f) = \frac{1}{\tau}\left[\frac{f_0(x,k)}{\int_{-\infty}^{\infty} f_0(x,k)dk}\int_{-\infty}^{\infty} f(x,k)dk - f(x,k)\right] \tag{3.9}$$

where τ is the relaxation time of an electron in the device. $f_0(x,k)$ is the equilibrium Wigner distribution function, which is the solution to equation (3.1) with $S(f) = 0$ and no change in the bias voltage applied across the device; i.e., $V_L = V_0$.

Boundary conditions are imposed on the Wigner distribution function:

$$f(0,k) = \frac{4\pi m^* k_B T}{h^2}\ln\left\{1 + \exp\left[-\frac{1}{k_B T}\left(\frac{h^2k^2}{8\pi^2m^*} - \mu_0\right)\right]\right\}, k > 0 \tag{3.10}$$

$$f(L,k) = \frac{4\pi m^* k_B T}{h^2}\ln\left\{1 + \exp\left[-\frac{1}{k_B T}\left(\frac{h^2k^2}{8\pi^2m^*} - \mu_L\right)\right]\right\}, k < 0 \tag{3.11}$$

where k_B is Boltzmann's constant, T is the temperature, and μ_0 and μ_L are the Fermi energies at the corresponding ends of the device.

Finally, the current density in the device, the main quantity of interest, can be calculated by

$$j(x) = \frac{h}{2\pi m^*}\int_{-\infty}^{\infty} kf(x,k)dk. \tag{3.12}$$

4. Discretizing the Wigner-Poisson equations. Since the system of equations specified in the Wigner-Poisson formulation is too difficult to solve analytically, numerical techniques are used to approximate the solution. First, we divide the spatial domain into N_x equally spaced grid points, with the length in between grid points defined as $\Delta x = \frac{L}{N_x - 1}$. The grid points are given by

$$x_i = (i-1)\Delta x \qquad \text{for i} = 1, 2, ..., N_x. \tag{4.1}$$

To discretize the momentum domain, we must first truncate the domain from $(-\infty, \infty)$ to $(-K_{max}, K_{max})$ where K_{max} is chosen such that when $|k| > K_{max}$, $f(x,k) \approx 0$. We will use $K_{max} = 0.25$ inverse Angstroms as a best approximation [4]. Then the momentum domain can be divided into N_k equally spaced grid points, with the length between grid points given by $\Delta k = \frac{2K_{max}}{N_k}$, and the grid points defined as

$$k_j = \frac{(2j - N_k - 1)\Delta k}{2} \qquad \text{for j} = 1, 2, ..., N_k. \tag{4.2}$$

Defining the grid points this way allows us to avoid a grid point at $k_j = 0$ where there would be a singularity.

We will denote a solution of the Wigner distribution function f at a grid point (x_i, k_j) as f_{ij}.

To approximate the kinetic term $K(f)$, we will use a second-order upwind difference method since we have boundary conditions at one end of the device (which end of the device depends on the sign of k). So the approximation is

$$K(f_{ij}) \approx \begin{cases} -\frac{hk_j}{2\pi m^*}\left(\frac{-3f_{ij} + 4f_{i-1,j} - f_{i-2,j}}{2\Delta x}\right) & , \quad k_j > 0 \\ -\frac{hk_j}{2\pi m^*}\left(\frac{3f_{ij} - 4f_{i+1,j} + f_{i+2,j}}{2\Delta x}\right) & , \quad k_j < 0. \end{cases} \tag{4.3}$$

The potential $P(f)$ term is discretized using the composite trapezoidal rule

$$P(f_{ij}) \approx -\frac{4}{h}\sum_{j'=1}^{N_k} f_{ij'} T(x_i, k_j - k_{j'}) w_{j'} \tag{4.4}$$

where the $w_{j'}$ are the weights of the composite trapezoidal rule:

$$w_{j'} = \begin{cases} \Delta k & \text{for j}' = 2, 3, ..., N_k - 1, \\ \frac{\Delta k}{2} & \text{for j}' = 1, N_k \end{cases} \tag{4.5}$$

For the $T(x_i, k_j - k_{j'})$ term, we must make additional approximations to discretize this integral. Since the upper limit of the integrand in equation (3.4) is infinity, we must truncate the limit at a value $L_c \leq L$ called the correlation length. However, L_c may or may not correspond to a grid point in the spatial domain, and thus we must modify the weights for the discretization to take this into account.

Assume L_c falls between grid points N_c and $N_c + 1$. We can divide the integral into two pieces, an integral from 0 to x_{N_c} and another from x_{N_c} to L_c, and then use the composite trapezoidal rule to approximate the integral over $[0, x_{N_c}]$ and the trapezoid rule for the integral over $[x_{N_c}, L_c]$. For an arbitrary function $g(x)$, this would be

$$\int_0^{x_{N_c}} g(x)dx + \int_{x_{N_c}}^{L_c} g(x)dx \approx \sum_{m=1}^{N_c} g(x_m) w_m + \frac{(L_c - x_{N_c})}{2}\left[g(x_{N_c}) + g(L_c)\right] \tag{4.6}$$

where the composite trapezoidal weights w_m are as defined for the $P(f)$ term. Using Taylor expansions and setting $h_{N_c} = L_c - x_{N_c}$ and $h_{N_c+1} = x_{N_c+1} - L_c$, we have

$$g(x_{N_c}) = g(L_c) - h_{N_c} \cdot g'(L_c) + O(h_{N_c}^2) \tag{4.7}$$

$$g(x_{N_c+1}) = g(L_c) + h_{N_c+1} \cdot g'(L_c) + O(h_{N_c+1}^2). \tag{4.8}$$

Since $h_{N_c}, h_{N_c+1} \leq \Delta x$, we can use $O(\Delta x^2)$ in place of the error terms in each of the above equations. Eliminating the $g'(L_c)$ term yields an approximation for $g(L_c)$ that can be used to simplify the $T(x_i, k_j - k_{j'})$ term:

$$g(L_c) = \frac{h_{N_c} \cdot g(x_{N_c+1}) + h_{N_c+1} \cdot g(x_{N_c})}{\Delta x} + O(\Delta x^2) \tag{4.9}$$

Thus the $T(x_i, k_j - k_{j'})$ term can be approximated as

$$T(x_i, k_j - k_{j'}) \approx \sum_{i'=1}^{N_c+1} [U(x_i + x_{i'}) - U(x_i - x_{i'})] \sin(2x_{i'}(k_j - k_{j'}))w_{i'} \tag{4.10}$$

where the $w_{i'}$ are the modified trapezoidal weights

$$w_{i'} = \begin{cases} \frac{\Delta x}{2} & \text{for i}' = 1 \\ \Delta x & \text{for i}' = 2, 3, ..., N_c - 1 \\ \frac{\Delta x + h_{N_c}}{2} + \frac{h_{N_c} h_{N_c+1}}{2\Delta x} & \text{for i}' = N_c \\ \frac{h_{N_c}^2}{2\Delta x} & \text{for i}' = N_c + 1. \end{cases} \tag{4.11}$$

The scattering term $S(f)$ can be discretized using the composite trapezoidal rule as

$$S(f_{ij}) \approx \frac{1}{\tau} \left[\frac{f_0(x_i, k_j)}{\sum_{j'=1}^{N_k} f_0(x_i, k_{j'}) w_{j'}} \sum_{j'=1}^{N_k} f_{ij'} w_{j'} - f_{ij} \right] \tag{4.12}$$

where the $w_{j'}$ are the standard weights listed in equation (4.5) for the $P(f)$ term.

To discretize Poisson's equation, we use a center difference formula

$$\frac{u_p(x_{i-1}) - 2u_p(x_i) + u_p(x_{i+1})}{\Delta x^2} = \frac{q^2}{\epsilon} [N_d(x_i) - n(x_i)] \tag{4.13}$$

with $u_p(x_1) = V_0$ and $u_p(x_{N_x}) = V_L$.

The electron density $n(x)$ and the current density $j(x)$ are also approximated using the trapezoidal rule:

$$n(x_i) \approx \frac{1}{2\pi} \sum_{j=1}^{N_k} f_{ij} w_j \tag{4.14}$$

$$j(x_i) \approx \frac{h}{2\pi m^*} \sum_{j=1}^{N_k} k_j f_{ij} w_j \tag{4.15}$$

where the weights are again the standard trapezoidal weights listed in equation (4.5) for $P(f)$.

5. Numerical Implementation. To solve a standard nonlinear equation $W(x) = 0$, Newton's method can be used to find a root of the equation by solving the iterative equation $x_{i+1} = x_i - W'(x_i)^{-1}W(x_i)$, where $W'(x_i)$ is the Jacobian of W at x_i. When $\|x_{i+1} - x_i\| \leq \epsilon$ where ϵ is an error tolerance, then x_{i+1} is an approximate root.

However, for the discretized Wigner distribution function, equation (3.1), creating the iteration step $W'(x_i)^{-1}W(x_i)$ is not computationally efficient, particularly when the $W'(x_i)$ matrix is large (typical implementations have on the order of 125,000-1 million+ grid points, so the matrix could be approximately 1 million $\times$ 1 million). Instead, we will use an inexact Newton method to solve the system. (For more information on inexact Newton methods, see [3] and the references contained therein.) With inexact Newton methods, the nonlinear equation is approximated by

$$\|W'(x_i)s + W(x_i)\| \leq \eta_i \|W(x_i)\| \tag{5.1}$$

where η_i is a forcing term that controls the size of the relative residual. Newton's method is used to compute the x_i, and the step s is approximated by a linear iterative method. For Wigner-Poisson, we will use GMRES to solve for the steps, since GMRES is a Krylov subspace method for solving non symmetric systems that does not require the computation of an iteration matrix[3].

This method can be efficiently implemented using Sandia's Trilinos software, which has a variety of built in inexact Newton method solvers, including Newton-GMRES. In addition, Trilinos allows the option to use finite differences to approximate the Jacobian-vector product, since computing and discretizing the Jacobian of the Wigner-Poisson formulation would be extremely complicated.

6. C++ Class Descriptions. Since Trilinos is written in C++, we decided to write the main program code in C++ to take advantage of the object oriented structure. Ten classes were created, each representing a different piece of the Wigner-Poisson equations. Each class has a Compute function to calculate either the appropriate action on the Wigner function f or to compute terms that will be used later in calculating f. The two classes that involve finite difference approximations for the kinetic term and the poisson term each have an Operator function that sets up the sparse matrix associated with the derivative term in the equation.

Barrier The first class, the "Barrier" class, creates the barrier and doping profiles that are used in the solution of the potential term $P(f)$. The Compute function calculates the relative energy band profile, $\Delta_c(x)$, and doping profile $N_d(x)$ at each grid point along the length of the device, which are used in equations (4.10) and (4.13) respectively.

The next classes created were those to calculate the specific terms in the Wigner distribution function:

- a "kineticMethod" class for the kinetic term $K(f)$
- a "scatterMethod" class for the scattering term $S(f)$
- a "potentialMethod" class for the potential term $P(f)$

kineticMethod The kineticMethod class has an Operator function to create the sparse matrix associated with the upwind/downwind discretization of the derivative term in equation (4.3), and a Compute function which applies the matrix to the Wigner distribution function f.

scatterMethod The scatterMethod's Compute function calculates the result of equation (4.12) for a given f.

The potentialMethod has a Compute function to calculate equation (4.4) for a given f. However, the term $T(x, k - k')$ must be known for all values of k', so several more classes were created to break down the computation of $T(x, k - k')$ into more manageable pieces. These pieces must be computed before the Compute function in the potentialMethod class is executed. The classes are

- "SinMat" to calculate the $\sin(2xy)$ term
- "TcMethod" and "TpMethod" to calculate the T integral

SinMat The Compute function in the SinMat class calculates the values of $\sin(2xy)$ for $(x, y) \in [0, L] \times (-2K_{max}, 2K_{max})$ in equation (4.10).

TcMethod and TpMethod Since the potential energy inside the device, $U(x)$, is the sum of two functions per equation (3.5), with $\Delta_c(x)$ fixed and $u_p(x)$ the solution to the Poisson equation, we can break the T integral into two separate integrals and assign a C++ class to each. Rewriting the T integral as the sum of two terms, T_p and T_c, we have:

$$T(x_i, k_j - k_{j'}) = \sum_{i'=1}^{N_c+1} [U(x_i + x_{i'}) - U(x_i - x_{i'})] \sin(2x_{i'}(k_j - k_{j'})w_{i'} \tag{6.1}$$

$$= \sum_{i'=1}^{N_c+1} [u_p(x_i + x_{i'}) - u_p(x_i - x_{i'})] \sin(2x_{i'}(k_j - k_{j'})w_{i'} \tag{6.2}$$

$$+ \sum_{i'=1}^{N_c+1} [\Delta_c(x_i + x_{i'}) - \Delta_c(x_i - x_{i'})] \sin(2x_{i'}(k_j - k_{j'})w_{i'}$$

$$= T_p(x_i, k_j - k_{j'}) + T_c(x_i, k_j - k_{j'}). \tag{6.3}$$

Since both $\Delta_c(x)$ and $\sin(2xy)$ are fixed for given values of (x, y), T_c can be precomputed and used throughout the remaining calculations. This is accomplished through the Compute function in the TcMethod class. A similar Compute function in the TpMethod calculates the value of $T_p(x_i, k_j - k_{j'})$; however, the value of $T_p(x, k - k')$ at a given x value depends on the solution to Poisson's equation, $u_p(x)$, so another class called "poissonMethod" is defined to compute $u_p(x)$.

poissonMethod The poissonMethod class has an Operator function to create the sparse matrix associated with the second order central difference approximation in equation (4.13). The associated Compute function solves the linear system $Ax = b$ where A is the sparse matrix and b is the right hand side term that includes the doping profile $N_d(x)$ and the electron density $n(x)$.

EleDensMethod and CurrentMethod The last two classes, EleDensMethod to compute the electron density using equation (4.14) and CurrentMethod to compute the current density using equation (4.15), both have a Compute function to calculate the respective function at a particular value of x_i. The CurrentMethod class is called only once the Wigner function f has converged at a particular value of the bias voltage.

7. Solution Process using Trilinos. To solve the system in Trilinos, we start by creating a ProblemInterface class that implements the LOCA::Epetra::Interface::Required interface. The ProblemInterface constructor calls the other C++ class constructors to set up each class and compute any data that will be fixed for the life of the run, such as the Barrier profiles, the SinMat matrix, the T_c term, and both of the finite difference matrices for the kinetic term and the Poisson equation.

Next, the ComputeF function calls the Compute function for each remaining class do the appropriate computation. The kinetic term, potential term, and scattering term are then added together to form the Wigner function $W(f)$. See figure 7.1 for a flowchart of the process.

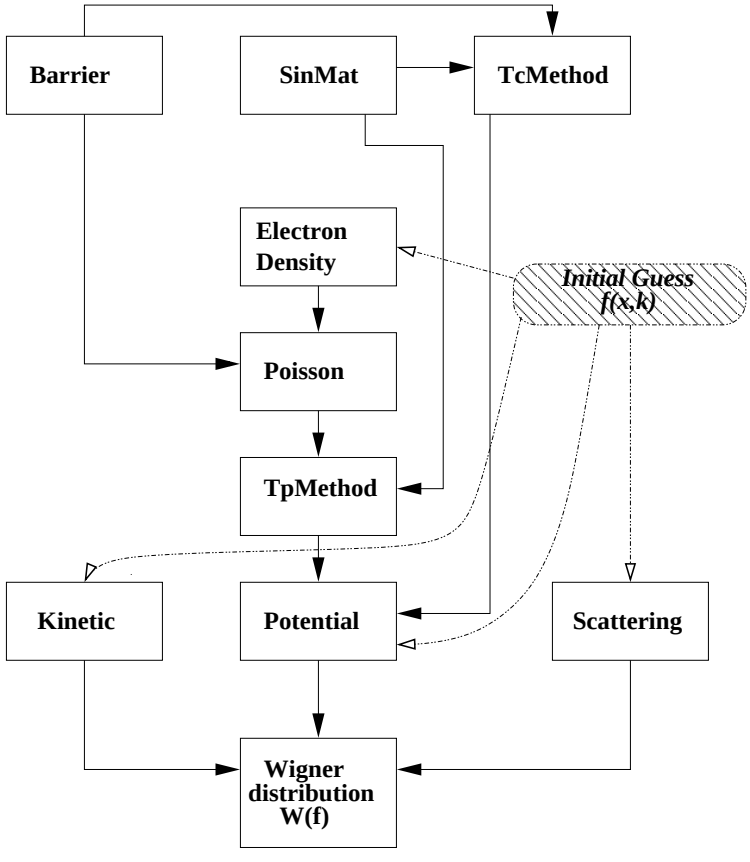


FIG. 7.1. *Flowchart of the ComputeF function.*

The system of equations will be solved as two homotopy problems. The first involves continuation on the barrier profile, starting from 0 and increasing up to $\Delta_c(x)$, in order to calculate $f_0(x,k)$; and the second is done on the bias applied to the device, starting at V_0 and increasing to V_L, to calculate the current density as a function of voltage.

To start the computation, an initial guess for f must be chosen. Then an initial distribution f is calculated using zero bias (that is, $V_0 = V_L$) and a barrier profile identically equal to 0. To calculate $f_0(x,k)$, which is used in equation (4.12), we set $S(f) \equiv 0$ and solve equation (3.1) by increasing the barrier profile from 0 to $\Delta_c(x)$ by multiplying $\Delta_c(x)$ by a constant and then increasing the constant incrementally from 0 to 1, solving for f at each incremental step. This is done *via* Trilinos by calling LOCA, Trilinos' continuation package. Once f_0 has been computed, continuation is performed again, this time by calculating $S(f)$ in equation (3.1) and solving for f using the ending bias value as the continuation parameter. Finally, the current density can be computed from f as the bias voltage is increased from V_0 to V_L.

8. Conclusions. Object oriented design gives flexibility for investigating new algorithms. For example, changing the discretization of the Poisson equation is a local change

in one class. Similarly, tradeoffs between memory usage and FLOPS, such as storing the entire SinMat, can be easily investigated with a local change to one class.

The C++/Trilinos implementation defined above provides more computational efficiency as well as greater flexibility in modeling a variety of devices than previous models do. Although other implementations exist in both Matlab and Fortran, Trilinos is written in C++ to handle memory allocation and computations more efficiently, which will allow finer meshes to be simulated in shorter runs times than those in previous work [4, 6]. In addition, Trilinos' ability to handle parallel computation will decrease run times even more.

Also, due to the flexibility inherent in the Barrier class structure, devices involving multiple barriers and nonsymmetric barrier and doping profiles can be analyzed more easily than in previous models, which were hard-coded to accept a fixed number of barriers (generally only two) and only symmetric barrier and doping profiles [4, 6]. Therefore, the C++/Trilinos implementation will increase the breadth of knowledge about RTDs and potentially other nanoscale devices that can be modeled using the Wigner-Poisson formulation.

9. Acknowledgements. This paper was partially supported by Army Research Office grants W911NF-07-1-0112 and NanoRTD LLC.

REFERENCES

[1] F. Buot and K. Jensen, *Lattice weyl-wigner formulation of exact many-body quantum-transport theory and applications to novel solid-state quantum-based devices*, Physical Review B, 42 (1990), pp. 9429–9457.
[2] L. Chang, L. Esaki, and R. Tsu, *Resonant tunneling in semiconductor double barriers*, Applied Physics Letters, 24 (1974), pp. 593–595.
[3] C. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, no. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1995.
[4] M. S. Lasater, *Numerical Methods for the Wigner-Poisson Equations*, PhD thesis, North Carolina State University, 2005.
[5] K. Ng, *Complete Guide to Semiconductor Devices*, Wiley-Interscience, New York, New York, 2nd ed., 2002.
[6] G. J. Recine, *Numerical Simulation of Quantum Electron Transport in Nanoscale Resonant Tunneling Structures*, PhD thesis, Stevens Institute of Technology, 2004.
[7] T. Sollner, W. Goodhue, P. Tannenwald, C. Parker, and D. Peck, *Resonant tunneling through quantum wells at frequencies up to 2.5thz*, Applied Physics Letters, 43 (1983), pp. 588–590.
[8] R. Tsu and L. Esaki, *Tunneling in a finite superlattice*, Applied Physics Letters, 22 (1973), pp. 562–564.
[9] P. Zhao, H. Cui, and D. Woolard, *Dynamical instabilities and i-v characteristics in resonant tunneling through double-barrier quantum well systems*, Physical Review B, 63 (2001), pp. 075302–1–075302–14.
[10] P. Zhao, H. Cui, D. Woolard, K. Jensen, and F. Buot, *Simulation of resonant tunneling structures: Origin of the i-v hysteresis and plateau-like structure*, Journal of Applied Physics, 87 (2000), pp. 1337–1349.

AB INITIO PATH INTEGRAL MOLECULAR DYNAMICS STUDY OF INTERMOLECULAR PROTON TRANSFER REACTIONS

ALEJANDRO PÉREZ*, MARK E. TUCKERMAN†, HAROLD P. HJALMARSON‡, AND O. ANATOLE VON LILIENFELD§

Abstract. Classical free energy profiles of intermolecular proton transfer for model systems of DNA base pairs, and two models for silica, H_2—$Si(OH)_3$, and H_2—$OSi(OH)_3$ were computed using *ab initio* molecular dynamics in gas phase at 300 K. For all of the tested exchange-correlation functionals the description of the energetics of the silicon reactions has been found to be insufficiently accurate. The quantum free energy profiles for the silicon reactions were estimated using path integral Monte Carlo calculations on a fitted potential. In addition, quantum free energy profiles were computed for a proposed model of the DNA base pair using *ab initio* path integral molecular dynamics. It is shown that quantum effects on nuclei lead to a near complete suppression of the reverse barrier for the DNA base pair models. Thus, these findings do not support the idea that neutral tautomeric forms of DNA base pairs can be implicated in mutagenesis. Our path integral results underscore the importance of nuclear quantum effects in proton transfer reactions even at room temperature.

1. Introduction. Free energy is a fundamental quantity in chemistry and biology that characterizes the probability of an event to happen in the canonical ensemble. Absolute free energy values are difficult to compute because it entails a full sampling of the phase space, which is clearly unfeasible for complex systems. More commonly, one deals with free energy differences, such as the free energies of solvation, or binding of a susbtrate to the active site of an enzyme, for which phase space is greatly reduced. Consequently, it is not surprising to find a vast list of methods in the literature on how to compute free energy differences, which includes thermodynamic integration, [1] umbrella sampling, [2] blue moon ensemble, [3, 4] metadynamics, [5] etc. In the case of proton transfer reactions, high energy barriers are frequent and these methods have proven to be useful in estimating free energy differences.

In nature there are plenty of cases where the quantum nature of the nuclei plays an important role that cannot be neglected. Solving the quantum dynamics of many-body systems involving light atoms remains one of the most challenging problems in computational physics and chemistry due to the unfavorable computer scaling with system size and time scale of numerically exact methods. Quantum equilibrium properties, however, are routinely investigated using the path integral (PI) formalism developed by Feynman. [6, 7, 8] The PI interpretation fostered a new understanding of the microscopic world and provided a deep insight into various complex quantum phenomena, such as superfluidity. [9] The PI method was successfully applied to a wide variety of model systems, [10] and it represents a promising avenue in condensed matter physics. Unfortunately, direct application of this formalism to real time dynamics faces a severe *sign problem* that necessitates approximation schemes and continues to be one of the unsolved in natural sciences.

With the advent of fast parallel machines, recent *ab initio* path integral molecular dynamics (AIPIMD) simulations [11, 12] of hydrogen bonded system have become possible [13, 14, 15, 16]. In AIPIMD, the nuclei are quantized following the Feynman's path integral (PI) formalism [6, 7, 8] and the many-body potential is obtained "on the fly" from an electronic structure calculations using the Car-Parrinello methodology [17]. Thus, in AIPIMD not only thermal fluctuations and the possibility for bond breaking and formation are allowed but also tunneling and zero point energy effects on nuclei are included in the simulations.

*Corresponding author: ap1484@nyu.edu. Department of Chemistry, New York University.
†Department of Chemistry and Courant Institute of Mathematical Sciences, New York University.
‡Sandia National Laboratories, Albuquerque, New Mexico.
§Sandia National Laboratories, Albuquerque, New Mexico.

Chemical reactions involving proton transfer are ubiquitous in nature. A relevant system for which proton transfer plays an important role is the DNA base pairs. In 1963, Löwdin proposed that rare tautomeric forms of the DNA Watson-Crick base pairs could be implicated in DNA point mutations [18]. These tautomers are generated by the antiparallel and concerted transfer of two hydrogen-bonded protons between DNA base pairs: adenine (A) and thymine (T); and cytosine (C) and guanine (G). Theoretical studies have later shown that the AT tautomeric equilibrium is unlikely to play any role in DNA mispairs, whereas the GC tautomers could potentially have an impact on mutational mechanisms. In this context, several factors need to be considered in DNA mutagenesis: (1) concentration and lifetimes of rare tautomers (2) barriers for the forward and backward double proton transfer (DPT) (3) characteristic time scale of the cell's DNA repairing mechanisms (4) physical conditions, such as hydration and charge state of base pairs.

Despite much research in the last 10 years, so far there is a lack of direct experimental evidence supporting the relevance of this rare forms in DNA mutagenesis. Thus, no conclusive proof has been reported so far and the issue of DNA mutations induced by intermolecular tautomerism of the base pairs remains an open question. To the best of our knowledge, full quantum treatment of the double proton transfer in DNA base pairs have not been reported using AIPIMD.

In this report, we have performed AIPIMD simulations for two model systems that mimic the hydrogen-bonding pattern of DNA Watson-Crick base pairs. The model systems investigated here are displayed in Fig. 4.4 and Fig. 4.7. The formamide-formamidine (FIFA) dimer has been previously used [19] to model the interaction between AT base pair, see Fig. 4.4. Recently, Leszczynski and coworkers performed a post-Hartree-Fock investigation of the FIFA complex in gas phase and aqueous solution. [19] Their theoretical study found strong evidence that in gas phase FIFA exhibits a concerted and asynchronous mechanism, whereas in solution the reaction proceeds in a stepwise fashion with no clear minimum on the product site. [19] The other model system (N-guanidiylformaldehyde–methanimidamide formate, or simply GFMF) is proposed here to model the intermolecular double proton transfer in the GC base pair, see Fig. 4.7. It is surprising, however, that nobody has attempted an exploration of these model systems by relaxing the restriction of classical nuclei. Thus, in the present paper we have carried out the first *ab initio* path integral molecular dynamics (PIMD) study of the FIFA and GFMF complexes in gas phase. We computed the free energy profile for the concerted double proton transfer both classically and quantum mechanically.

Hydrogen also plays an important role in metal oxide semiconductors (MOS) industry as responsible for the generation and also the removal of defects at SiO_2/Si interfaces. The energetics of hydrogen interacting with different silicon clusters has been investigated both theoretically and experimentally in order to elucidate the mechanism of reaction and shed some light into the atomistic details. Yet the dynamics is poorly understood and great disparity of results has been reported. [20] It is known, however, that atomic hydrogen can depassivate the SiO_2/Si interface by reaction with the hydrogen bonded to silicon sites and thus creating silicon dangling bonds. Also, it is well established that SiO dangling bonds can dissociate molecular hydrogen and create strong Si-OH bonds. In this article, we reexamine the energetics and free energy profiles of some model reactions to shed some light into these controversial results. We estimate the nuclear quantum effect on fitted potentials using a one dimensional path integral code.

This article is organized as follows: Sec. (2.1) reviews the equilibrium path integral molecular dynamics methodology of Ref. [21]. Sec. (2.2) presents a brief review on free energy computation of rare events with special emphasis on the umbrella sampling technique. [2]. In the accompanying Section (3), the model systems and relevant technical de-

tails are presented. The article continues in Sec. (4) with a discussion of the main results. Conclusions are drawn in Sec. (6).

2. Background and Theory.

2.1. Equilibrium path-integral molecular dynamics. In this section, we briefly review the methodology of path-integral molecular dynamics (PIMD). In the proceeding, $\hat{H}$ will denote the Hamiltonian operator of the system, $\beta = 1/k_{\mathrm{B}}T$ the inverse thermal energy, and $Z(\beta) = \mathrm{Tr}\left[\exp(-\beta\hat{H})\right]$ the canonical quantum partition function.

The discrete path-integral expression for the quantum canonical partition function for a single particle of mass m and Hamiltonian $\hat{H} = \hat{p}^2/2m + V(\hat{x})$ in one dimension is

$$Z_P(\beta) = \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{P/2} \int dx_1 \cdots dx_P \tag{2.1}$$
$$\exp\left\{-\beta \sum_{k=1}^{P}\left[\frac{1}{2}m\omega_P^2(x_k - x_{k+1})^2 + \frac{1}{P}V(x_k)\right]\right\},$$

where $\omega_P = \sqrt{P}/(\beta\hbar)$, and P is the Trotter number or number of imaginary time slices along the thermal path. The paths must satisfy the cyclic condition $x_{P+1} = x_1$ arising from the trace of the (unnormalized) canonical density matrix $\exp(-\beta\hat{H})$.

Without affecting any of the thermodynamic or equilibrium properties of the system, we can introduce a set of P uncoupled normalized Gaussian integrals into the previous expression

$$Z_P(\beta) = \mathcal{N} \int dp_1 \cdots dp_P \int dx_1 \cdots dx_P \tag{2.2}$$
$$\exp\left\{-\beta \sum_{k=1}^{P}\left[\frac{p_k^2}{2m'_k} + \frac{m}{2}\omega_P^2(x_k - x_{k+1})^2 + \frac{1}{P}V(x_k)\right]\right\},$$

where m'_k are some arbitrary fictitious mass parameters and $\mathcal{N}$ is an overall normalization constant. These fictitious masses m'_k are totally arbitrary as far as equilibrium properties are concerned. These parameters m'_k merely constitute a tool to navigate phase space and sample a rather complicated multidimensional integrand. Thus, the quantum canonical partition function could in principle be computed via molecular dynamics (MD) using the following classical Hamiltonian

$$H = \sum_{k=1}^{P}\left[\frac{p_k^2}{2m'_k} + \frac{1}{2}m\omega_P^2(x_k - x_{k+1})^2 + \frac{1}{P}V(x_k)\right], \tag{2.3}$$

which describes the motion of a cyclic polymer chain with harmonic nearest-neighbor interactions in an attenuated external classical potential $V(x)/P$. [8, 22] Because of the resemblance of the cyclic polymer to a necklace, the imaginary time points are colloquially referred to as "beads", and the variables $\mathbf{x} = x_1, ..., x_P$ are referred to as the "primitive" path-integral variables. The parameters m'_k determine the time scale on which the imaginary time points $x_1, ..., x_P$ are sampled. This computational approach to calculate the quantum partition function was termed path integral molecular dynamics (PIMD).

However, as pointed by Hall and Berne, [23] the efficiency of the primitive or naive PIMD algorithm is very poor due to the dominance of the harmonic forces from the quantum kinetic

energy. As one tries to converge the PI by increasing the Trotter number P, the harmonic springs become stiffer and, at the same time, the external potential gets more attenuated – the external potential $V(x)/P$ becomes a small perturbation to the periodic harmonic motion. Consequently, the system enters in the so-called Kolmogorov-Arnold-Moser (KAM) regime, and its behavior becomes highly non-ergodic. [24] Moreover, even if thermostats are coupled to each degree of freedom in the system, the wide frequency spectrum introduced by the harmonic coupling causes the MD time step to be limited by the fast modes, thereby leading to very poor sampling of the low-frequency modes.

A solution to all these problems was introduced by Tuckerman *et al.* [21] and consists of a combination of three elements: (1) the variables in Eq. (2.3) are transformed to a new set of coordinates that diagonalizes the harmonic coupling; (2) the fictitious masses m'_k are adjusted so that all modes move on the same time scale; (3) a thermostat is coupled to each mode degree of freedom in the system so as to effect rapid sampling, equipartitioning, and a proper canonical distribution.

The transformation from "primitive" to a new set of so-called "staging" modes [21] can be derived from similar transformations used in path-integral Monte Carlo. [25] In its simplest form, the staging modes $q_1, ..., q_P$ (denoted collectively by $\mathbf{Q}$) are given by

$$\begin{aligned} q_1 &= x_1 \\ q_k &= x_k - \frac{(k-1)x_{k+1} + x_1}{k}, \qquad k = 2, ..., P. \end{aligned} \tag{2.4}$$

Note that this transformation is a special case of a more general staging transformation discussed in Ref. [21]. When the change of variables given by Eq. (2.4) is introduced into Eq. (2.3), the discretized partition function becomes

$$\begin{aligned} Z_P(\beta) = \mathcal{N} \int dp_1 \cdots dp_P \int dq_1 \cdots dq_P \\ \exp\left\{-\beta \sum_{k=1}^{P} \left[\frac{p_k^2}{2m'_k} + \frac{1}{2} m_k \omega_P^2 q_k^2 + \frac{1}{P} V(x_k(\{\mathbf{Q}\})) \right] \right\}, \end{aligned} \tag{2.5}$$

where $x_k(\{\mathbf{Q}\})$ indicates the inverse transformation, and the masses m_k are defined to be

$$\begin{aligned} m_1 &= 0 \\ m_k &= \frac{k}{k-1} m, \qquad k = 2, ..., P. \end{aligned} \tag{2.6}$$

Note that, by this definition, the mode variable q_1 drops out of the quantum kinetic energy part so that its motion is solely governed by the external potential V. In order to ensure that all modes move on the same time scale, the fictitious masses m'_k are chosen according to $m'_1 = m$ and $m'_k = m_k$. Therefore, PIMD in staging modes is defined by the transformed Hamiltonian

$$H_{\text{stage}} = \sum_{k=1}^{P} \left[\frac{p_k^2}{2m'_k} + \frac{1}{2} m_k \omega_P^2 q_k^2 + \frac{1}{P} V(x_k(\{\mathbf{Q}\})) \right]. \tag{2.7}$$

In Eq. (2.7), the momenta p_k are treated as "conjugate" to the mode variables q_k, which means that the dynamics generated by Eqs. (2.7) and (2.3) are different because the transformation is not canonical. This freedom of choosing the fictitious masses is permitted as far as equilibrium thermodynamic quantities are concerned.

Finally, once the equations of motion are derived using Eq. (2.7), each mode variable is coupled to a separate thermostat, *e.g.*, a Nosé-Hoover chain thermostat. [26] In Ref. [21],

a more general staging type of approach was introduced by allowing staging "segments" of length j to be defined, thereby providing a natural cutoff between fast and slow modes, a generalization that was shown to possess certain advantages regarding the convergence of path integrals with large P.

The authors of Ref. [21] also suggested that the same scheme could be used with normal mode variables. This idea was subsequently implemented by Cao and Voth in the context of centroid molecular dynamics (see below), [27] by Tuckerman *et al.*, in *ab initio* path integrals methods, [28] by Marx *et al.* in *ab initio* centroid molecular dynamics algorithms, [29] and by Martyna *et al.* in the context of path integrals at constant pressure. [30] The transformation in this case takes the form

$$q_k = \frac{1}{\sqrt{P}} \sum_{i=1}^{P} U_{ki} x_i, \tag{2.8}$$

where the linear transformation U_{ki} diagonalizes the matrix arising from the quantum kinetic part: $A_{ij} = 2\delta_{ij} - \delta_{i,j-1} - \delta_{i,j+1}$ with $A_{i,P+1} = A_{i1}$ and $A_{i0} = A_{iP}$. Introducing this change of variables in Eq. (2.3) yields a partition function that has the same form as Eq. (2.6) but with

$$\begin{aligned} m_k &= m\lambda_k \quad \text{where} \\ \lambda_{2k-1} &= \lambda_{2k-2} = 2P\left\{1 - \cos\left[\frac{2\pi\,(k-1)}{P}\right]\right\} \end{aligned} \tag{2.9}$$

and $\lambda_1 = 0$, $\lambda_P = 4P$ (for even P). As with the staging transformation, the mode q_1 drops out of the quantum kinetic energy term. In fact, it is the centroid mode of the ring polymer,

$$q_1 = \frac{1}{P} \sum_{i=1}^{P} x_i. \tag{2.10}$$

In order to ensure that all modes move on the same time scale, the fictitious masses m'_k are chosen according to $m'_k = m_k$ and $m'_1 = m$, which is the optimal choice for the free particle. However, depending on the system, other choices for the fictitious kinetic masses may be more efficient. [31] As in the staging case, each normal-mode degree of freedom must be also coupled to its own thermostat.

The schemes reviewed in this section have proved highly useful in equilibrium PIMD. In *ab initio* PIMD the forces and total energies are derived from a modern electronic structure method. Most of the codes nowadays use density functional theory (DFT) to evaluate these quantities. [32] The methods assume Born-Oppenheimer approximation although extensions to excited states have recently been explored. *Ab initio* PIMD simulations are rather expensive since it requires P times electronic structure calculations than its classical counterpart. Note, however, that PIMD is embarrassingly parallelized as the replicas are propagated independently with very little communication among them. In addition, the electronic orbital coefficients are not obtained by explicit diagonalization of the electronic Hamiltonian each time step but rather are propagated from initially optimized coefficients using the Car-Parrinello scheme. [17] For more details on *ab initio* PIMD, see Refs. [33, 29, 34].

2.2. Free energy methods and rare events. Free energy is a fundamental quantity in Statistical Mechanics that measures the probability of a event happening in the canonical ensemble. [35] The free energy along a given reaction coordinate $\xi = \xi(\mathbf{x})$ (also known as potential of mean force) is defined mathematically by

$$F(\xi') = -\frac{C}{\beta} \frac{\int d\mathbf{x}\, e^{-\beta U(\mathbf{x})}\, \delta\left[\xi(\mathbf{x}) - \xi'\right]}{Q(\beta)}, \tag{2.11}$$

where $Q(\beta)$ is the canonical configuration partition function, $C = h^{-3N}/N!$, and $U(\mathbf{x})$ is the potential energy of the system.

If energy barriers in U are lower than or similar to the thermal energy $k_B T$ (≈ 0.6 kcal/mol at 300K) then the free energy along a reaction coordinate ξ can be computed by standard MD using a simple histogram technique $F(\xi) = -k_B T \ln P(\xi)$.

Rare events are defined as processes characterized by the presence of energy barriers significantly higher than the thermal energy. Direct sampling of these phenomena via ordinary MD is difficult and special techniques are required to enhance the sampling of inaccessible regions. In the literature, there are numerous approaches to compute free energies of infrequent events, such as thermodynamic integration, [1] blue moon ensemble, [3, 4] metadynamics, [5] umbrella sampling, [2], among others. This paper deals exclusively with reactions that have barriers well above the thermal energy and thus with rare events. The umbrella sampling method [2] was chosen here because of its easy implementation in existing codes.

In umbrella sampling, one restrains the position of a selected reaction coordinate ξ to a certain window i using typically an harmonic bias potential

$$U_i(\xi) = \frac{K}{2}(\xi - \xi_i)^2 . \tag{2.12}$$

The harmonic force constant K should be at least twice the negative value of the curvature of the potential energy surface (PES) at the transition state

$$K \approx -2\left(\frac{\partial^2 E}{\partial \xi^2}\right)^{\ddagger}, \tag{2.13}$$

but its value is otherwise arbitrary. Then, one divides the interval of interest in several windows and performs independent MD simulations on each window. The greater the force constant K is, the larger is the number of windows required. A period of equilibration is required for each window, which is then followed by a production run. From these simulations, one obtains a biased probability distribution $P_i^b(\xi)$ for each window i and using a direct histogram $F_i^b(\xi) = -k_B T \ln P_i^b(\xi)$, the biased free energy is obtained.

To reconstruct the original free energy profile from biased or non-Boltzmann MD simulations one uses the weighted histogram analysis method (WHAM). [36, 37] The unbiased (underlying or original) free energy $F = F^{ub}$ is recovered by

$$F_i^{ub}(\xi) = -\frac{1}{\beta}\ln P_i^b(\xi) - U_i(\xi) - C_i, \tag{2.14}$$

where C_i are some constants which differ for each window. These constants can be determined by solving iteratively the following pair of non-linear equations

$$\begin{aligned} P^{ub}(\xi) &= \frac{\sum_{i=1}^{N_w} N_i\, P_i^b(\xi)}{\sum_{i=1}^{N_w} N_i \exp\{\beta[C_i - U_i(\xi)]\}} \\ C_i &= -\frac{1}{\beta}\ln \int d\xi\; P^{ub}(\xi)\; \exp\{-\beta U_i(\xi)\}, \end{aligned} \tag{2.15}$$

where N_i is the number of configurations sampled in window i and N_w is the total number of umbrella windows. Once this calculation is converged, the unbiased free energy is easily obtained from $F^{ub}(\xi) = -k_B T \ln P^{ub}(\xi)$. The WHAM requires sufficient overlap between consecutive windows for successful convergency.

A suitable reaction coordinate to study a typical proton transfer reaction from one site S_1 to another site S_2, not necessarily belonging to the same molecule,

$$S_1 - \mathrm{H} \cdots S_2 \Longleftrightarrow S_1 \cdots \mathrm{H} - S_2 \tag{2.16}$$

is the relative difference between two relevant distances $\boldsymbol{\xi} = \mathbf{r}_{\mathrm{H}S_1} - \mathbf{r}_{\mathrm{H}S_2}$. In normal mode PIMD simulations, one applies the restrain potential on the centroid mode of the ring polymer, [22] see Eq. (2.10). Then the forces on the implicated centroids due to a harmonic umbrella potential in window i, Eq. (2.12), are

$$\mathbf{f}_{S_1} = K(\xi - \xi_i)\boldsymbol{\xi}/\xi \tag{2.17}$$

$$\mathbf{f}_{S_2} = \mathbf{f}_{S1} \tag{2.18}$$

$$\mathbf{f}_{\mathrm{H}} = -2K(\xi - \xi_i)\boldsymbol{\xi}/\xi \tag{2.19}$$

for S_1, S_2, and H atoms, respectively. These forces bias the sampling of the centroids so as the generalized reaction coordinate remains near the value given by ξ_i.

3. Computational details. In this section, we present the computational details for the model systems investigated. All *ab initio* MD simulations were performed at the Γ-point using Kohn-Sham density functional theory [32] within the generalized gradient approximation functional as implemented in the plane-wave pseudopotential code `CPMD`. [38] The umbrella bias potential was implemented in the routine `pi_md.F` of the `CPMD` program. The harmonic force constant K was set to 0.1 a.u. which is high enough to drive the proton transfer in all studied reactions. The time step used was carefully chosen to properly integrate all internal modes and ensure conservation of the total energy. Each umbrella window was equilibrated for about 1 ps using a time step 0.048 fs. Subsequent windows started with the last geometry of the previous window to facilitate the equilibration. Statistics were acquired for 3.6 ps using a time step of 0.072 fs. This small time step is necessary for the adiabatic propagation of electronic coefficients in the Car-Parrinello approach. [17]

For *ab initio* path integral calculations, 16 beads were used for the quantization of all the nuclei at these physical conditions. All the present calculations are in gas phase. Solvent effects were not included due to computational cost of path integral (PI) calculations. However, solvent is not expected to play a crucial in the intermolecular DPT of DNA base pairs because the reaction site is excluded from the solvent and the DNA interior has small dielectric constant [39, 40].

Canonical sampling was achieved via Nosé-Hoover chain thermostats [26] of length 4 and 7^{th} order Suzuki/Yoshida integrator. [41, 42] All atomic degrees of freedom were thermostated, which is crucial for efficient PIMD calculations. The CP mass was set to 400 a.u. and no thermostats on the electronic orbital coefficients were necessary. The fictitious electronic kinetic energy and total energy were carefully monitored and found stable during all the simulations. The initial velocities of the ions and electronic coefficients were set to zero at the beginning of the equilibration using the "QUENCH" option in `CPMD`. The molecular system was placed in a sufficiently large box and cluster (isolated molecule) periodic boundary conditions were adopted. The Tuckerman-Martyna Poisson solver was used to electrostatically decouple the periodic images. [43] All the atoms were placed such a way that they were found more than 3 Å from the walls. Moreover, the electronic density was plotted and checked that it did not touch nor cross any boundaries.

The PBE0 hybrid exchange-correlation functional was used for all silicon reactions. [44, 45, 46] The local-spin density approach was adopted in all open-shell systems. The silicon systems were difficult to treat at the DFT level and from the energetics analysis this hybrid functional turned out to be the best (see next paragraph). A plane wave kinetic energy cutoff

of 75 Ry and non-local Goedecker-Teter-Hutter (GTH) pseudopotentials were used to expand the valence electronic orbitals. [47] The plane wave cutoff for orbitals and density in the exact exchange part was set to 50 and 75 Ry, respectively, using the CPMD option "HFX CUTOFF". The system was placed in a box of $12 \times 10 \times 10$ Å^3 and cluster PBCs were adopted. The free energy profile was computed at the classical level using umbrella sampling. An *ab initio* path integral study was not further pursued for the silicon systems due to the computational cost of the hybrid functionals and poorly description offered by DFT. Rather, the quantum suppression in the free energy barrier was estimated using a path integral Monte Carlo [48] simulation on a numerical fit to the PES.

For the FIFA complex, the energetics is very well described by DFT and the BLYP [49] functional was used. The valence electronic orbitals were expanded using a plane wave kinetic energy cutoff of 100 Ry in a isolated box of $12.5 \times 10.5 \times 6.5$ Å^3. The core electronic orbitals were represented by non-local Goedecker-Teter-Hutter (GTH) pseudopotentials [47] as given by Krack. [50]. Classical and quantum free energy profiles were computed using umbrella sampling. A total of 14 umbrella windows were collected.

For the GFMF complex, a plane wave cutoff of 75 Ry was used. The molecule was placed in a box of $15 \times 15 \times 8$ Å^3. The core electronic orbitals were represented by the Troullier-Martins pseudopotentials [51]. A total of 11 umbrella windows were simulated to reconstruct the free energy profile.

All molecular visualization was carried out using the program VMD. [52] and the scripts found in Ref. [53].

4. Results and Discussion.

4.1. Energetics and the choice of the DFT functional. In order to assess the validity of the DFT parametrization employed here, we computed the energetics for each model system. Quantum chemical calculations were performed using the cluster code TURBOMOLE using the basis set "def-TZVPP". [54] This doubly polarized triple-zeta valence basis set is similar in quality to the standard Pople's $6-311G(2df)$ and large enough to provide sufficiently converged results (in the basis set limit sense) for various levels of the theory. The resolution of the identity featured in TURBOMOLE was not employed in any of the calculations for sake of accuracy. Note that these structures are not fully relaxed; thus, it is not surprising to find discrepancies with the free energy plots of the next section. Future work will include energy differences of more optimized structures. Preliminary results have already revealed the same trend (not shown).

Unrestricted (local spin density) DFT calculation of the reaction $H + HSi(OH)_3 \Rightarrow H_2 + Si(OH)_3$ were performed with TURBOMOLE and CPMD. The results are listed in Table 4.1. All the exchange-correlation functionals investigated seem to describe poorly the energetics of this reaction and severely underestimate the reaction barrier. PBE functional even predicts a negative value (-1.172) for the forward barrier. In view of Table 4.1, it seems that exact exchange plays a crucial to yield qualitatively correct results. Thus, we decided to carry out free energy calculation with the hybrid functional PBE0, which has been shown to yield reasonable values for a great number of organic compounds. [44] Full *ab initio* path integral calculations on this reaction were not pursued here. Instead, the suppression in the free energy barrier was be estimated via a one-dimensional Path Integral Monte Carlo code (PIMC) [48] calculation on the numerical fit of the PES.

The energetics for the reaction $H_2 + OSi(OH)_3 \Rightarrow H + Si(OH)_4$ is listed in Table (4.2). Not surprisingly, current DFT functionals also performs very poorly for this open-shell system and again only the hybrid exchange-correlation functionals (PBE0, B3LYP) manage to achieve qualitative results. As before, PBE0 was used to carried out the *ab initio* CPMD

TABLE 4.1

Relative energies (kcal/mol) of three representative structures for the silicon reaction $H + HSi(OH)_3 \Rightarrow H_2 + Si(OH)_3$. The structures correspond approximately to reactants (R), transition state (TS), and products (P). All energies were computed with TURBOMOLE *and "def-TZVPP" basis set except the last value, which was from a converged* CPMD *calculation.*

method	$\Delta E_{\text{R-P}}$	$\Delta E_{\text{TS-R}}$	$\Delta E_{\text{TS-P}}$
UHF	1.65	13.36	15.01
MP2	5.87	8.93	14.79
CC2	6.89	8.39	15.28
PBE0	9.89	3.00	12.89
PBE	12.93	0.74	13.89
B3LYP	12.79	1.60	14.39
BLYP	15.35	0.04	15.39
PBE0[a]	9.5	3.35	11.95

[a] values computed using CPMD.

TABLE 4.2

Relative energies (kcal/mol) of three representative structures for the silicon reaction $H_2 + OSi(OH)_3 \Rightarrow H + Si(OH)_4$. All energies were computed with TURBOMOLE *and "def-TZVPP" basis set.*

method	$\Delta E_{\text{R-P}}$	$\Delta E_{\text{TS-R}}$	$\Delta E_{\text{TS-P}}$
MP2	28.55	6.63	35.18
CC2	27.85	3.998	31.84
PBE0	18.72	1.62	20.34
PBE	17.06	-1.172	15.35
B3LYP	13.39	2.34	15.73
BLYP	10.97	0.59	11.56

simulations. [44, 45, 46] Only the classical simulation was performed and the path integral calculation was not pursued.

The reason of the failure of all current DFT functionals in these silicon reactions is not clear to us but it could be related to the need of a multiple Slater determinant description. In fact, TURBOMOLE T1-diagnostic criteria reported a too high value which indicates that the molecular system may not be well described by a single Slater determinant [54]. Moreover, it should be mentioned that previous DFT works reported a great variability for the energetics of these silicon reaction. [20] If our findings are correct, they would invalid many *ab initio* studies on similar silicon compounds.

The results for the FIFA complex are listed in Table (4.3). The table reveals that, unlike the silicon reactions, all electronic methods (including DFT) agree very well with each other. The hybrid exchange-correlation functionals Perdew-Burke-Ernzerhof zero (PBE0) [45, 46, 44] and the Becke 3-parameter Lee-Yang-Parr (B3LYP) [55] almost achieve chemical accuracy with respect the reference value coupled cluster doubles (CC2). The BLYP functional was chosen to perform the *ab initio* MD simulations as a good compromise between accuracy and computational cost. [49] The last line in Table (4.3) gives the value computed from converged CPMD (100 Ry, box size: $12.5 \times 10.5 \times 6.5$ Å^3) which is in excellent agreement with value predicted by TURBOMOLE (less than 0.6 kcal/mol difference).

Finally, Table (4.4) show the energetics for the GFMF dimer. Unlike the silicon systems, the energetics is very well described at the DFT level. The BLYP functional [49] was chosen as a good compromise between accuracy and computational cost.

TABLE 4.3

Relative energies (kcal/mol) of some representative structures of the FIFA complex. All energies were computed with TURBOMOLE *using the "def-TZVPP" basis set, except the last row which was obtained from a converged* CPMD *calculation. The structures correspond approximately to reactants (R), transition state (TS), and products (P).*

method	$\Delta E_{\text{P-R}}$	$\Delta E_{\text{TS-P}}$	$\Delta E_{\text{TS-R}}$
RHF	11.80	9.47	21.27
MP2	8.62	4.37	12.99
CC2	**9.24**	**3.08**	**12.32**
PBE0	8.93	2.81	11.74
PBE	8.56	1.19	9.75
B3LYP	9.47	3.52	12.99
BLYP	**9.30**	**2.44**	**11.74**
BLYP[a]	**9.57**	**2.18**	**11.75**

[a] values computed using CPMD.

TABLE 4.4

Relative energies (kcal/mol) of some representative structures of the GFMF complex. All energies were computed with TURBOMOLE *using the "def-TZVPP" basis. The structures correspond approximately to reactants (R), transition state (TS), and products (P).*

method	$\Delta E_{\text{P-R}}$	$\Delta E_{\text{TS-P}}$	$\Delta E_{\text{TS-R}}$
RHF	13.50	14.65	28.15
MP2	9.37	4.04	13.41
CC2	**10.24**	**2.67**	**12.91**
PBE0	10.95	3.00	13.95
PBE	10.63	0.55	11.18
B3LYP	11.39	4.78	16.17
BLYP	**11.16**	**3.15**	**14.31**

4.2. Depassivation of $HSi(OH)_3$ by a H radical. Fig. 4.1 shows the potential energy surface (PES) for the reaction between molecular hydrogen and a single occupied sp^3 silicon dangling bond (a simplified model of what is known as E' centers), [20]

$$H + HSi(OH)_3 \Rightarrow H_2 + Si(OH)_3. \tag{4.1}$$

This reaction features a very late transition state and, therefore, is highly exothermic ($\approx$ 14 kcal/mol) towards products in accord with Hammond's postulate. As mentioned before, there has been a great disparity of activation barriers in the literature and it seems that this reaction is a difficult case for current exchange-correlation functionals in DFT.

Fig. 4.2 shows the classical free energy profile of the reaction at 300 K. The data suggest the strong tendency of hydrogen radicals to depassivate $HSi(OH)_3$ at finite temperatures, which has already been confirmed by theoretical [56, 57, 20] and experimental studies. The suppression in the classical barrier was estimated using a path integral Monte Carlo (PIMC) code. The results are shown in Fig. 4.1. PIMC predicts a barrier suppression of approximately 1.5 kcal/mol.

Another possible depassivation reaction is

$$H^+ + HSi(OH)_3 \Rightarrow H_2 + Si(OH)_3^+. \tag{4.2}$$

However, the activation energy was estimated earlier to be very large ($\approx$ 30 kcal/mol) [58] and was not further investigated.

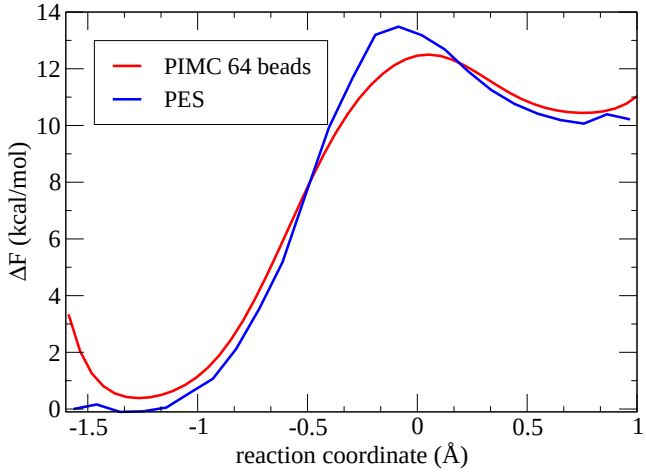


FIG. 4.1. *Potential energy surface scan (blue) for the reaction in Eq. 4.1. In red, it is shown the estimated free energy profile from a PIMC calculation. The left side of PES corresponds to products H_2+$Si(OH)_3$, and the right side to reactants, H + $HSi(OH)_3$.*

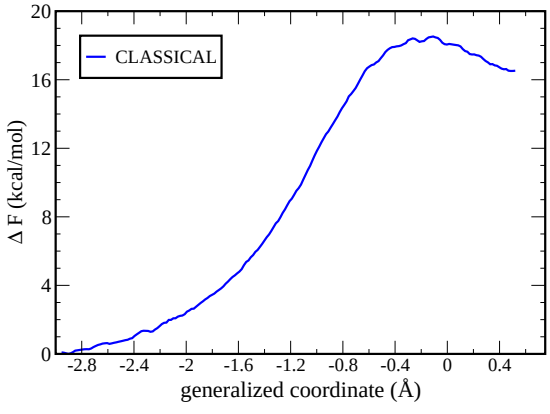


FIG. 4.2. *Classical free energy profile of reaction in Eq. 4.1 at* 300*K. Left side: products; right side: reactants.*

4.3. Cracking of H_2 at $OSi(OH)_3$ radical. In this section, we study the mechanism of hydrogen cracking at $OSi(OH)_3$ sites (sometimes called non-bridging oxygen or *NBO* centers).

Fig. 4.3 displays the classical free energy profile for the reaction

$$H_2 + OSi(OH)_3 \Rightarrow H + Si(OH)_4. \tag{4.3}$$

Fig. 4.3 shows the classical free energy profile at 300 K. The reaction is highly exothermic towards the formation of SiO-H bond ($\approx$ 12 kcal/mol) and our data effectively supports the fact that oxygen dangling bond in Si acts as strong molecular cracking sites for hydrogen. This finding is corroborated by previous experimental and theoretical studies. [20]

4.4. Formamidine-Formamide complex in gas phase. In this section, we present the results of the formamidine-formamide (FIFA) complex in gas phase at 300 K. The molecular structure for the normal and "rare" tautomers are displayed in Fig. 4.4.

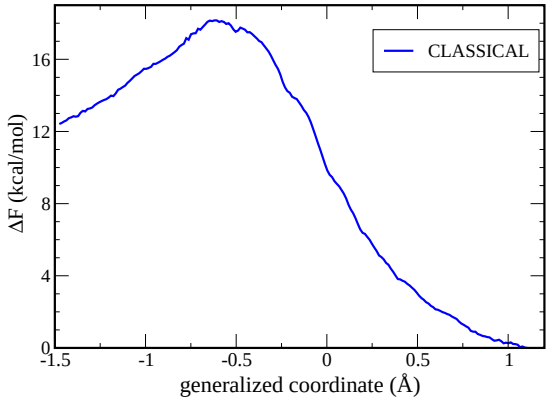


FIG. 4.3. *Classical free energy profile of reaction Eq. 4.3 at* 300*K. The left side corresponds to reactants,* H_2+$OSi(OH)_3$*, and the right side to products,* $H + Si(OH)_4$.

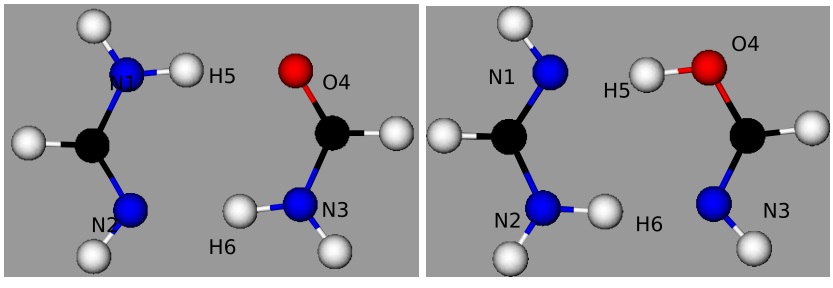


FIG. 4.4. *TOP: Molecular structure of the optimized formamidine-formamide (FIFA) complex. BOTTOM: Molecular structure of the double proton transfer form of FIFA dimer ("rare" tautomer).*

According to Ref. [19], the system features a double proton transfer (DPT) via a concerted and asynchronous mechanism in gas phase. The DPT form of FIFA is displayed in Fig. 4.4. The choice of a single reaction coordinate is a non trivial task for this system and in principle a complete characterization of this process would require a full two-dimensional map, which is clearly prohibitive. Here, the relative distance to the less acidic proton (H_5 in Fig. 4.4) was chosen here as a reaction coordinate, *i.e.*, $\boldsymbol{\xi} = \mathbf{r}_{H_5N_1} - \mathbf{r}_{H_5O_4}$. According to the aforementioned study, [19] the complete transfer of H_5 from nitrogen N_1 to oxygen O_4 marks the appearance of products. By the time H_5 migrates to O_4, the other proton, H_6, has already been spontaneously transferred to N_2 . Thus, this coordinate is appropriate to follow the double proton transfer reaction as it approximates well the determining step of the reaction.

Fig. 4.5 shows the free energy profile for double proton transfer in FIFA at 300 K. The classical profile features a very asymmetric barrier with a late transition state. The reaction is endothermic ($\approx$ 10 kcal/mol) towards products in accord with Hammond's postulate. Classical simulations predict a barrier of approximately 2.5 kcal/mol from rare to normal tautomers, with a well defined equilibrium between stable geometries. The classical free energy barrier at 300 K agrees with the barrier computed at 0 K from adiabatic calculations. The relative energies at 0 K of FIFA system agrees very well with the value in Ref. [19, 59, 60] computed at the MP2/6-31+G** level. Our FIFA value for forward barrier is also in good agreement with the energy profile reported in Ref. [61] computed at the B3LYP/6-311++G** level.

Nuclear quantum effects, on the other hand, have a dramatic impact on the reverse barrier in FIFA, which is now completely eliminated (red line in Fig. 4.5). Thus, according to the

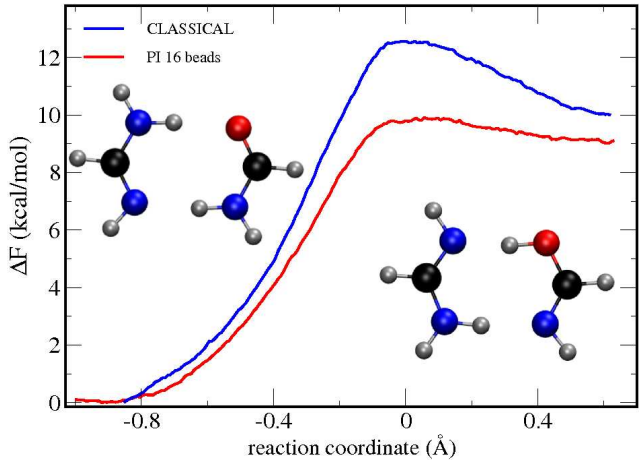


FIG. 4.5. *Classical (blue) and quantum (red) free energy profile of FIFA at 300 K computed at the BLYP level. Inset shows the corresponding structures. Left: normal tautomer; right: "rare" tautomer. Statistical errors (not shown) are too small to be visible.*

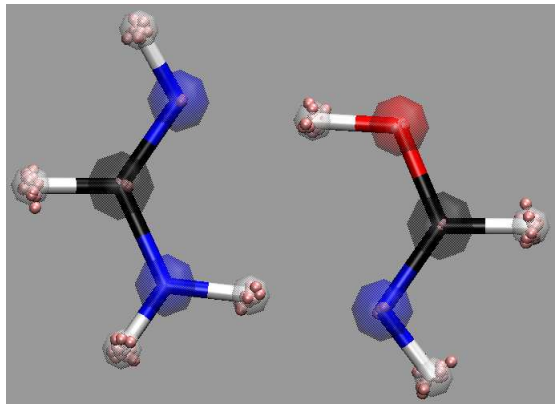


FIG. 4.6. *Path integral representation of double proton transfer form (or "rare" tautomer) of FIFA.*

path integral result, there is no stable structure corresponding to the double proton transfer form of FIFA. As reported in Ref. [13], not only the transferring proton species, but also the heavy-atom skeleton must be quantized as the latter has a non-trivial effect on the proton tunneling reaction. Our findings highlight the importance of nuclear quantum effects on protons in biomolecules even at room temperature. The path integral representation of the "rare" tautomer of FIFA is given in Fig. 4.6. For a path integral dynamical animation of FIFA dimer, see links in Ref. [62].

4.5. GC model system in gas phase. Fig. 4.7 shows the chemical structure of our proposed model system (guanidylformaldehyde-methanimidamidyil formide or hereafter GFMF) for the GC base pair. To our knowledge, this is the first time that this system is used to model the intermolecular DPT in GC base pair.

Fig. 4.8 shows the free energy profile for double proton transfer in GFMF at 300 K. As in FIFA dimer, the relative distance to the less acidic proton (H in Fig. 4.7) was chosen here as a reaction coordinate, *i.e.*, $\boldsymbol{\xi} = \mathbf{r}_{\mathrm{HN}} - \mathbf{r}_{\mathrm{HO}}$. This model system features similar energetics

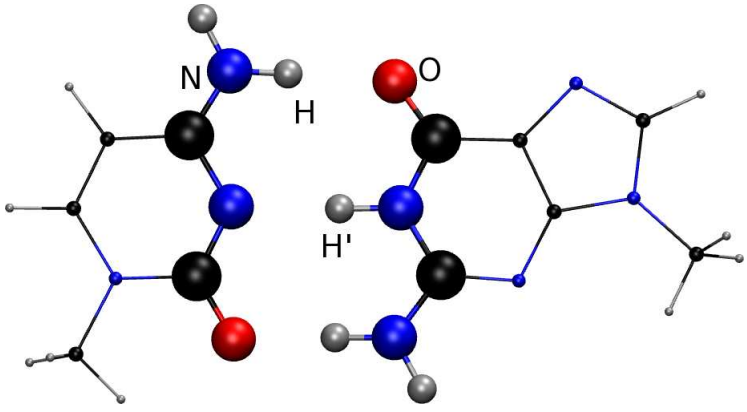


FIG. 4.7. *Scheme of model GFMF (big spheres) superimposed to the true Watson-Crick GC base pair (small spheres). LEFT shows normal tautomers; RIGHT shows "rare" tautomers. Hydrogen atoms have been added to both models in order to satisfy valency. Color code: carbon (black), oxygen (red), hydrogen (grey), nitrogen (blue). The most relevant atoms in the double proton transfer have been labeled.*

as the FIFA dimer, although the shape of the classical barrier is sharper. Classically, it is predicted a reverse barrier of approximately 4 kcal/mol separating well defined equilibrium geometries. The energetics at 0 K of our GC model system agrees very well with the value for true GC in Ref. [63] computed at the MP2/6-31G** level. This agreement shows that our model system provides a decent description of the DPT in the real system despite of lacking aromaticity. Once again, upon quantization of nuclei, the barrier entirely disappears and the system shows no clear minimum structure on the products side (right in Fig. 4.8). Thus, nuclear quantum effects on proton have a major role in the stability of canonical DNA base pairs and are ultimately responsible of the fidelity of DNA replication against neutral intermolecular tautomeric forms. For a path integral animation of the transition state of the DPT in GFMF, see links in Ref. [62].

Thus, neutral tautomers have little role in DNA mutagenesis. Obviously, this does not exclude the possibility that charged or radical DNA base pairs play a relevant role in point mutations. In fact, recent theoretical studies [64, 59, 61] have shown that charged protonated base pairs display smaller activation barriers, which makes proton transfer facile. In addition, hydration effects [65], coordination to a metal cation [60], and molecular environment will also have an undoubtedly effect but too complicated to be modeled with path integrals.

5. Future Work. The entropic contribution for these gas phase reactions is small owing to the small molecular reorganization accompanying the DPT. This is not surprising, as the thermal fluctuations in a rigid heavy-skeleton play a minor role in the proton hopping mechanism. In solution, however, this may not be the case, and the solvent molecules reorganize to stabilize the transition state, leading to a reduction in entropy. However, a competing effect also exists of favorable enthalpic interactions with the solvent. Thus, future work will include studying the same reactions in the presence of explicit solvent (perhaps a few water molecules). In addition, an unbiased MD simulation will be performed starting from products to histogram the centroid position of hydrogens H_5 and H_6 (see Fig. 4.4). Experimental data on FIFA system will be searched to possibly establish a comparison with our numerical results. The umbrella sampling will be repeated using different initial conditions with the goal of further reducing the statistical error. Alchemical transmutation of the transferring protons to deuteriums are currently underway to estimate the isotope effect in FIFA. Finally, free energies will also be estimated (within the harmonic approximation for the vibrational degrees

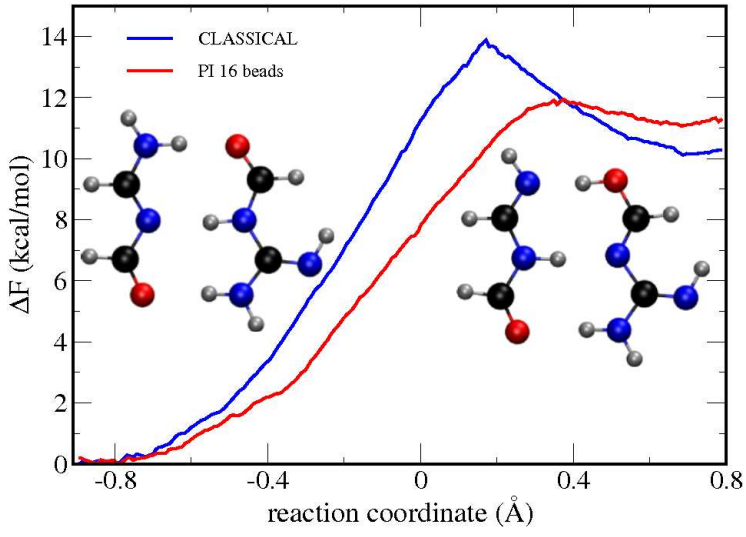


FIG. 4.8. *Free energy profile for the double proton transfer in GFMF at 300 K. Full quantum: red; Classical nuclei: blue. Inset shows the corresponding structures. Left: normal tautomer; right: "rare" tautomer. Statistical errors (not shown) are too small to be visible.*

of freedom) using a thermochemistry analysis.

6. Conclusions. In the first part of this paper, we studied the interaction of hydrogen molecule with various silicon clusters in gas phase to gain some atomistic insight into the mechanism of the reaction of hydrogen with silica. Classical free energy profiles were computed at 300 K using density functional theory with biased molecular dynamics. For the silicon reaction studied here it is shown that the E' centers acts as depassivating sites whereas the *NBO* centers get strongly passivated by homolytically dissociating H_2. Quantum effects as estimated by PIMC show a significant suppression in the barrier for these reactions.

Quantum effects on nuclei were also included to study the double proton transfer reaction in models of DNA base pairs in gas phase. Using the Feynman's path integral formalism, a complete suppression in the reverse barrier was found which suggest that "rare" tautomers are not implicated in DNA point mutations.

In summary, it is shown that nuclear quantum effects play a crucial role in the double proton transfer reaction of model systems resembling the hydrogen bonding pattern of DNA base pairs. Inclusion of nuclear quantum effects leads to a complete suppression of the reverse barrier, which implies that these rare tautomers do not play any significant role in DNA mutations. Based on energetic arguments, these results are expected to carry over the true Watson-Crick base pairs unless external factors help stabilizing "rare" tautomers.

In a broader context, it was shown that proton tunneling can qualitatively affect the outcome in complex biological systems, such as DNA or enzyme active sites. Our results represent a fundamental change of paradigm with respect to previous studies on DNA base pair mutations. This work hopes to influence the way simulations on biomolecules are carried out in the future.

7. Acknowledgments. The authors would like to thank Dr. Ann E. Mattsson for critical reading of this manuscript. Special thanks go to Peter A. Schultz for many discussions. A. P. and H. P. H. gratefully acknowledge financial support from LDRD Project No. 117866/02. O.A.vL. acknowledges support from SNL Truman Program LDRD Project No. 120209. San-

dia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Co., for the United States Department of Energys National Nuclear Security Administration under Contract No. DE-AC04-94AL85000.

REFERENCES

[1] J. G. Kirkwood. Statistical Mechanics of Fluid Mixtures. *J. Chem. Phys.*, 3:300, 1935.
[2] G. M. Torrie and J. P. Valleau. Monte Carlo free energy estimates using non-Boltzmann sampling: Application to the sub-critical Lennard-Jones fluid. *Chem. Phys. Letts.*, 28:578–581, 1974.
[3] E. A. Carter, G. Ciccotti, J. T. Hynes, and R. Karpal. Constrained reaction coordinate dynamics for the simulation of rare events. *Chem. Phys. Lett.*, 156:472–477, 1989.
[4] M. Sprik and G. Ciccotti. Free energy from constrained molecular dynamics. *J. Chem. Phys.*, 109:7737–7744, 1998.
[5] A. Laio and M. Parrinello. Escaping free-energy minima. *Proc. Natl. Acad. Sci. USA*, 99:12562–12566, 2002.
[6] R. P. Feynman (ed. L. M. Brown). *Feynman's thesis: A new approach to Quantum Theory.* World Scientific, Singapore., 2005.
[7] R. P. Feynman. Space-Time Approach to Non-Relativistic Quantum Mechanics. *Rev. Mod. Phys.*, 20:367–387, 1948.
[8] R. P. Feynman and A. R. Hibbs. *Quantum Mechanics and Path Integrals, McGraw-Hill, New York.*, 1965.
[9] D. M. Ceperley. Path integrals in the theory of condensed helium. *Rev. Mod. Phys.*, 67:279–355, 1995.
[10] J. Cao and G. A. Voth. The formulation of quantum statistical mechanics based on the Feynman path centroid density. II. Dynamical properties. *J. Chem. Phys.*, 100:5106–5117, 1994.
[11] D. Marx and M. Parrinello. Ab initio path integral molecular dynamics: Basic ideas. *J. Chem. Phys.*, 104:4077–4082, 1996.
[12] M. E. Tuckerman, D. Marx, M. L. Klein, and M. Parrinello. Efficient and general algorithms for path integral Car-Parrinello molecular dynamics. *J. Chem. Phys.*, 104:5579, 1996.
[13] M. E. Tuckerman and D. Marx. Heavy-Atom Skeleton Quantization and Proton Tunneling in Intermediate-Barrier Hydrogen Bonds. *Phys. Rev. Lett.*, 86:4946, 2001.
[14] S. Miura, M. E. Tuckerman, and M. L. Klein. An ab initio path integral molecular dynamics study of double proton transfer in the formic acid dimer. *J. Chem. Phys.*, 109:5290, 1998.
[15] J. A. Morrone and R. Car. Nuclear Quantum Effects in Water. *Phys. Rev. Lett.*, 101:017801, 2008.
[16] M. E. Tuckerman, D. Marx, M. L. Klein, and M. Parrinello. On the Quantum Nature of the Shared Proton in Hydrogen Bonds. *Science*, 275:817–820, 1997.
[17] R. Car and M. Parrinello. Unified Approach for Molecular Dynamics and Density-Functional Theory. *Phys. Rev. Lett.*, 55:2471–2474, 1985.
[18] P. O. Löwdin. Proton Tunneling in DNA and its Biological Implications. *Rev. Mod. Phys.*, 35:724–732, 1963.
[19] Y. Podolyan, L. Gorb, and J. Leszczynski. Double-Proton Transfer in the Formamidine-Formamide Dimer. Post-HartreeFock Gas-Phase and Aqueous Solution Study. *J. Phys. Chem. A*, 106:12103, 2002.
[20] M. Vitiello, N. Lopez, F. Illas, and G. Pacchioni. H2 Cracking at SiO2 Defect Centers. *J. Phys. Chem. A*, 104:4674–4684, 2000.
[21] M. E. Tuckerman, B. J. Berne, G. J. Martyna, and M. L. Klein. Efficient molecular dynamics and hybrid Monte Carlo algorithms for path integrals. *J. Chem. Phys.*, 99:2796–2808, 1993.
[22] D. Chandler and P. G. Wolynes. Exploiting the isomorphism between quantum theory and classical statistical mechanics of polyatomic fluids. *J. Chem. Phys.*, 74:4078, 1981.
[23] R. W. Hall and B. J. Berne. Nonergodicity in path integral molecular dynamics. *J. Chem. Phys.*, 81:3641, 1984.
[24] V. I. Arnold. Proof of a theorem of A. N. Kolmogorov on the invariance of quasi-periodic motions under small perturbations of the Hamiltonian. *Russ. Math. Surv.*, 18:9–36, 1963.
[25] E. L. Pollock and D. M. Ceperley. Simulation of quantum many-body systems by path-integral methods. *Phys. Rev. B*, 30:2555–2568, 1984.
[26] G. J. Martyna, M. L. Klein, and M. E. Tuckerman. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *J. Chem. Phys.*, 97:2635–2643, 1992.
[27] J. S. Cao and G. A. Voth. The formulation of quantum statistical mechanics based on the Feynman path centroid density. IV. Algorithms for centroid molecular dynamics. *J. Chem. Phys.*, 101:6168, 1994.
[28] M. E. Tuckerman, D. Marx, M. L. Klein, and M. Parrinello. Efficient and general algorithms for path integral Car-Parrinello molecular dynamics. *J. Chem. Phys.*, 104:5579, 1996.
[29] D. Marx, M. E. Tuckerman, and G. J. Martyna. Quantum dynamics via adiabatic ab initio centroid molecular dynamics. *Comput. Phys. Comm.*, 118:166–184, 1998.
[30] G. J. Martyna, A. Hughes, and M. E. Tuckerman. Molecular dynamics algorithms for path integrals at constant pressure. *J. Chem. Phys.*, 110:3275–3290, 1999.
[31] M. H. Musër. On new efficient algorithms for PIMC and PIMD. *Comput. Phys. Comm.*, 147:83–86, 2002.

[32] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.*, 140:A1133–A1138, 1965.

[33] D. Marx and M. Parrinello. Ab initio path integral molecular dynamics: Basic ideas. *J. Chem. Phys.*, 104:4077–4082, 1996.

[34] J. Cao and G. J. Martyna. Adiabatic path integral molecular dynamics methods. II. Algorithms. *J. Chem. Phys.*, 104:2028–2035, 1996.

[35] L. D. Landau and E. M. Lifshitz. *Statistical Physics*. Pergamon Press, New York, 1968.

[36] A. M. Ferrendberg and R. H Swendsen. New Monte Carlo technique for studying phase transitions. *Phys. Rev. Lett.*, 61:2635–2638, 1988.

[37] A. M. Ferrendberg and R. H Swendsen. Optimized Monte Carlo data analysis. *Phys. Rev. Lett.*, 63:1195–1198, 1989.

[38] CPMD version 3.13.2, Copyright IBM Corp 1990-2008, Copyright MPI für Festkörperforschung Stuttgart., `http://www.cpmd.org/`, 1997-2001.

[39] A. O. Colson, B. Besler, and M. D. Sevilla. Ab initio molecular orbital calculations on DNA base pair radical ions: effect of base pairing on proton-transfer energies, electron affinities, and ionization potentials. *J. Phys. Chem.*, 96:9787–9794, 1992.

[40] A. O. Colson, B. Besler, and M. D. Sevilla. Ab initio molecular orbital calculations of DNA bases and their radical ions in various protonation states: evidence for proton transfer in GC base pair radical anions. *J. Phys. Chem.*, 96:661–668, 1992.

[41] M. Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *J. Math. Phys.*, 32:400, 1991.

[42] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, 150:262–268, 1990.

[43] G. J. Martyna and M. E. Tuckerman. A reciprocal space based method for treating long range interactions in ab initio and force-field-based calculations in clusters. *J. Chem. Phys.*, 110:2810, 1999.

[44] C. Adamo and V. Barone. Toward reliable density functional methods without adjustable parameters: The PBE0 model. *J. Chem. Phys.*, 110:6158, 1999.

[45] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.*, 77:3865, 1996.

[46] J. P. Perdew, Ernzerhof, and K. Burke. Rationale for mixing exact exchange with density functional approximations. *J. Chem. Phys.*, 105:9982, 1996.

[47] S. Goedecker, M. Teter, and J. Hutter. Separable dual-space Gaussian pseudopotentials. *Phys. Rev. B*, 54:1703–1710, 1996.

[48] V. M. Zamalin and G. E. Norman. The Monte-Carlo method in Feynman's formulation of quantum statistics. *USSR Comp. Math. and Math. Phys.*, 13:408–420, 1973.

[49] C. Lee, W. Yang, and R. G. Parr. *J. Chem. Phys.*, 37:785, 1988.

[50] M. Krack. Pseudopotentials for H to Kr optimized for gradient-corrected exchange-correlation functionals. *Theor. Chem. Acc.*, 114:145–152, 2005.

[51] N. Troullier and J. L. Martins. Efficient pseudopotentials for plane-wave calculations. *Phys. Rev. B*, 43:1993, 1991.

[52] A. Dalke W. Humphrey and K. Schulten. VMD version 1.8.6. *J. Molec. Graphics*, 14:33–38, 1996.

[53] `http://www.theochem.ruhr-uni-bochum.de/ axel.kohlmeyer/cpmd-vmd/part6.html`.

[54] R. Ahlrichs, M. Bär, M. Häser, H. Horn, and C. Kölmel. Electronic structure calculations on workstation computers: The program system turbomole. *Chem. Phys. Letters*, 162:165–169, 1989.

[55] A. D. Becke. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.*, 98:5648, 1993.

[56] H. A. Kurtz and S. P. Karna. Hydrogen Cracking in SiO2: Kinetics for H2 Dissociation at Silicon Dangling Bonds. *J. Phys. Chem. A*, 104:4780–4784, 2000.

[57] L. D. Crosby and H. A. Kurtz. Application of electronic structure and transition state theory: Reaction of hydrogen with silicon radicals. *Int. J. Q. Chem.*, 106:3149–3159, 2006.

[58] L. Tsetseris and S. T. Pantelides. Migration, incorporation, and passivation reactions of molecular hydrogen at the Si-SiO2 interface. *Phys. Rev. B*, 70:245320, 2004.

[59] M. Noguera, M. Sodupe, and J. Bertran. Effects of protonation on proton transfer processes in Guanine-Cytosine Watson-Crick base pair. *Theor. Chem. Acc.*, 112:318–326, 2004.

[60] L. Gorb, Y. Podolyan, P. Dziekonski, W. A. Sokalski, and J. Leszczynski. Double-Proton Transfer in Adenine-Thymine and Guanine-Cytosine Base Pairs. A Post-Hartree-Fock ab Initio Study. *J. Am. Chem. Soc.*, 126:10119–10129, 2004.

[61] M. Noguera, M. Sodupe, and J. Bertran. Effects of protonation on proton transfer processes in Adenine-Thymine Watson-Crick base pair. *Theor. Chem. Acc.*, 118:113–121, 2007.

[62] For a path integral animation of the double proton transfer reaction in FIFA and GFMF, see links at `http://www.cs.sandia.gov/ oavonli/`. The movie does not show any real dynamics but only the sampling of different microstates of the molecule accessible at that temperature.

[63] J. Florián and J. Leszczynski. Spontaneous DNA Mutations Induced by Proton Transfer in the Guanine-

Cytosine Base Pairs: An Energetic Perspective. *J. Am. Chem. Soc.*, 118:3010–3017, 1996.

[64] J. P. Cerón-Carrasco, A. Requena, C. Michaux, E. A. Perpéte, and D. Jacquemin. Effects of ions on the Proton Transfer Mechanism in DNA. *J. Phys. Chem. A*, 120:8159–8167, 2009.

[65] J. P. Cerón-Carrasco, A. Requena, C. Michaux, E. A. Perpéte, and D. Jacquemin. Effects of Hydration on the Proton Transfer Mechanism in the Adenine-Thymine Base Pair. *J. Phys. Chem. A*, 113:7892–7898, 2009.

A TWO-TEMPERATURE MODEL OF RADIATION DAMAGE IN α-QUARTZ

CAROLYN L. PHILLIPS*, PAUL S. CROZIER†, AND RUDOLPH J. MAGYAR‡

Abstract. Two-temperature models are used to represent the interaction between atoms and electrons during thermal transients such as radiation damage, laser heating, and cascade simulations. We introduce a two-temperature model applied to a wide-gap semiconductor, α-quartz, to model a heat deposition in a SiO_2 lattice. Our model of the SiO_2 electronic subsystem is based on quantum simulations of the electronic response in a SiO_2 repeat cell. We observe how the parameterization of the electronic subsystem impacts the degree of permanent amorphization of the lattice, especially compared to a metallic-like electronic subsystem. The parameterization of the semiconductor electronic subsystem has a significant effect on the amount of residual defects in the crystal after 10 picoseconds.

1. Introduction. Two-temperature models (TTM) attempt to capture the interplay between electrons and ions in a material by modeling the electrons and the ions as two separate systems, with two separate temperatures, that are able to exchange energy through frictional forces applied to the ions [12, 32]. These models are used to capture high-energy events like laser heating [7, 6, 11, 22, 33, 13], sputtering [28], shock-induced melting[16], heterogeneous melting[14], and cascade simulations [5, 9, 10]. Electron interaction with fast moving ions, electron-phonon coupling, and heat transfer through the electronic subsystem provide additional heat transport mechanisms to the system.

Several models have been proposed to capture the energy exchange between the electronic and atomic subsystems. In 1989, Caro and Victoria [4] proposed that the inelastic scattering between electrons and high velocity ions could be modeled as a drag force, while electron-phonon interactions could be modeled as a Langevin thermostat connecting the atomic subsystem to the electronic subsystem. In 2007, Duffy and Rutherford [8, 27] introduced a TTM based on the work of Caro and Victoria [4]. Their version of TTM can account for the electronic stopping, electron-phonon coupling, and spatial and temporal evolution of the electronic subsystem. In [25], an energy conserving version of this inhomogeneous Langevin thermostat TTM was introduced. The material is modeled as heavy atoms, evolving under MD equations of motion, coupled to a continuum finite heat reservoir, which represents the electrons. Electronic stopping is modeled as a drag force that is only applied to atoms whose velocity exceeds a threshold velocity. A stochastic force term and a second drag force term on each atom, that is, a Langevin thermostat, is introduced to represent the energy interchange between the atomic subsystem and the excited electrons. The electronic subsystem is modeled as a continuum with energy evolved by numerical integration of the heat diffusion equation. This model permits the energy stored in the electronic subsystem to vary both spatially and temporally.

Algorithmically, at each MD time step, energy is exchanged between the atomic subsystem and electronic subsystem. The interaction with the atomic subsystem is a source term in the electronic subsystem heat diffusion equation, while the Langevin thermostat temperature is the local electronic temperature.

The force, due to the electronic subsystem, applied to each atom takes the form,

$$F_{Langevin} = -\gamma_i \mathbf{v}_i + \tilde{F}(t)$$

*The University of Michigan, phillicl@umich.edu
†Sandia National Laboratories, pscrozi@sandia.gov
‡Sandia National Laboratories, rjmagya@sandia.gov

where

$$\gamma_i = \gamma_p + \gamma_s \text{ for } v_i > v_0$$
$$\gamma_i = \gamma_p \text{ for } v_i \leq v_0$$

where γ_p and γ_s represent the friction coefficients due to electron-ion interaction and electron-stopping, respectively, and v_0 is the threshold velocity for the electron-stopping interaction. In regimes where electronic stopping does not apply, this force satisfies the fluctuation-dissipation theorem,

$$\langle \tilde{F}(t) \rangle = 0$$

$$\langle \tilde{F}(t') \cdot \tilde{F}(t) \rangle = 2k_b T_e \gamma_p \delta(t' - t)$$

and the atomic and electronic system will equilibrate to a shared temperature.

The electronic temperature, T_e, is calculated by solving the equation for heat diffusion,

$$C_e \frac{\partial T_e}{\partial t} = \nabla (\kappa_e \nabla T_e) - \Delta E_{\tilde{f}} / \Delta V \tag{1.1}$$

where, $\Delta E_{\tilde{f},j}$ is the energy transfered to the atomic system during the time step. Using this model, we consider the implications of a TTM for heat deposition in α-quartz.

2. Model.

2.1. Modeling an electronic subsystem for a semiconductor. In reference [15], it was proposed that a two temperature model is inappropriate for a system of covalent bonds because the structural response of the material to heating is different from metals and the model electron-phonon coupling is less valid. Instead, a molecular dynamic simulations performed on the basis of time-dependent, many-body potential energy surfaces derived from tight-binding Hamiltonians was used to model small ($\approx$ 60 atom) carbon and germanium systems. This quantum tight-binding method limits the size, energy range, and timstep of the system that can be modeled. While the specific formulation of the TTM may be less rigorous, by a careful parameterization of the TTM for a semiconductor system, we seek to coarse-grain model the energy transfer between the electrons and ions in a system large enough to study large heat deposition events. We could, in principle, capture the effects of temperature on the force-field to reproduce some of the bond creation and breaking, but temperature dependent force-fields are not available.

In order to obtain parameters for our TTM, we appeal to basic physics and density functional molecular dynamics simulations as implemented in VASP [19, 20, 17, 18, 3, 21, 24]. The version of the TTM [27] that we use requires that several parameters be judiciously gleaned from literature, theory, and computational results. These parameters are: C_e, electronic specific heat; κ_e, electronic thermal conductivity; g_p, energy exchange rate between electrons and lattice; and γ_p, electron-ion interaction rate.

The basic unit of the quantum simulations is 9 atom α-quartz (SiO_2) hexagonal supercell. In our calculations, we employed projected-augmented wave pseudopotentials, a 400 eV plane-wave cut off, and 9×9×9 points in the reduced brilluon zone. Many electron effects are considered at finite temperatures with the Mermin local density approximation to the exchange-correlation energy.

The heat capacity of the electrons was extracted by comparing the internal energy change of the free electrons at room temperature and 10,000 K while keeping the nuclei fixed. Since

α-quartz is a large gap system ($\approx$ 9 eV), the heat capacity of the electrons should be a highly non-linear function of temperature and could, in practice, be extracted as an interpolation by sampling many temperature points. The heat capacity prediction based on internal energy calculations compares to about a factor of two to what would have been extracted using the density of states to calculate this quantity according to the method Celli et al. [22].

The electron-phonon scattering rate is not directly accessible in ground-state calculations, and a number schemes have been put forth to obtain this quantity numerically. Celli et al. proposed a scheme that would extract this quantity from a calculated density of states but requires empirical data to describe certain phonon energies. An alternate scheme relies on real-time propagation of the many-body system within a time-dependent density functional framework [23]. The investigators simulated dynamics of excited carriers in a (3,3) carbon nanotube identifying a 230 fs time scale when coupling to ionic motion starts to dominate. We are currently developing the capacity to perform analogous simulations. We estimate the SiO_2 coupling time should be shorter than what was found in the TDDF-MD simulations of the carbon nanotubes since the α-quartz system has a larger gap. On the other hand, the electron-phonon time scale is unlikely to be shorter than 20 fs since this is the time-scale of electron-electron interactions. Thus, we can bracket the probably electron-phonon time scales between 20-200 fs.

The thermal conductivity of the electrons in α-quartz is expected to be exceedingly small at zero temperature and to vary drastically with the number of carriers at higher temperature. An estimate for this value can be found by appealing to the Wiedemann-Franz rule relating thermal to electronic conductivities. Since the experimentally known electronic conductivity at room temperature is exceedingly small, the thermal conductivity of the electrons is also small. Note that application of the Wiedemann-Franz rule is valid since we are considering only the electronic subsystem which near the Fermi-surface will experience nearly elastic collisions. A more rigorous treatment of conductivity would require the calculation of the temperature dependent carrier density, but owing to the large gap in α-quartz, is likely to remain well below the metallic and semi-conductor values.

2.2. Molecular Dynamics Model. Using the LAMMPS code [26], a simulation cell containing 19×19×20 repeat cells (3 SiO_2) of α-quartz (21,660 silicon atoms) was initialized at zero pressure. The silicon and oxygen atoms interacted via BKS interaction potential [30], which has been used to model quartz in the crystal and amorphous states [31] and transition between the two [2]. The system was equilibrated for 10,000 times steps to $T = 300K$ as an NPT ensemble, using a time step of 1.6 femtoseconds. A Nose/Hoover pressure barostat was used with a damping parameter of 1000.0 fs and a Langevin thermostat was used with a coupling parameter $\gamma = m/D$, where D is a damping parameter set to 33.333 fs. Next, for 1000 time steps, heat was pumped into a spherical region encompassing 567 silicon atoms (1700 total atoms) at a rate of 205.76 Kcal/(mole-fs). At the same time, at a distance of 84.26 Å, 410 silicon atoms (1230 total atoms) were thermally coupled to an infinite heat reservoir at $T = 300K$ by a Langevin thermostat with the damping parameter set to 33.333 fs. These atoms effectively become a heat sink. After 1000 time steps, the system was still 94% crystalline (per the definition provided in Section 2.2), but with a liquid spot at the center of the simulation box at a peak temperature of approximately $T = 11,100K$, and a descending thermal gradient to the heat sink at a temperature of approximately $T = 300K$. The heat source was then turned off. An electronic subsystem was then coupled to the system. A coarse grid of 6×6×6 electronic cells was used, with approximately 300.9 atoms per cell (100.3 silicon atoms). We did not vary the number of electronic cells used to be consistent with reference [25]. At this grid cell decomposition, the initial damage spot used for the bulk of the simulations contains $\approx$ 13 grid cells. The electronic temperature of each cell was

TABLE 2.1
Electronic Subsystems

Type	Label	Shared	Parameters	Initialized
Semiconductor	SCI	$\kappa_e = 0$	$\tau = 20$	local atomic temp
	SCII	$C_e = 0.00144$	$\tau = 200$	local atomic temp
	SCIII	$\rho_e = 0.781$	$\tau = 20$	300K
	SCIV		$\tau = 200$	300K
Metallic	MI	$\kappa_e = 0.011$	$C_e = 0.0004$	local atomic temp
	MII	$\tau = 1000$	C_e=$F(T_e)$	same energy as SCI
	MIII	$\rho_e = 2.16$	C_e=$F(T_e)$	300K
Weak	WI	C_e = 1/100 SCI		local atomic temp
	WII	τ = 100× SCI		300K

initialized. Over the course of the simulation, the residual heat from the liquid spot flowed to the heat sink until, after a long time, the system has uniformly cooled to $T = 300K$.

In reference [25], model systems for a material that heals from a heat deposition (Lennard Jones FCC crystal) and a material that partially heals but sustains permanent damage (a glass-forming Binary Lennard-Jones crystal) were considered with model electronic subsystems. The annealing behavior of the α-quartz system over several different spot sizes was considered with both "no electrons" and "Quench". The heated spot was scaled in both energy and size, for scale values of s = 0.89,1.0, 1.1, 1.2, 1.56, and 2.0 relative to the reference spot size.

Multiple electronic subsystems were attached to the α-quartz. The details of the electronic subsytems are contained in Table 2.1. Thermal conductivity, κ_e has units Kcal/(fs-K-Å-electron). The electronic specific heat has units Kcal/(mole-electron-K). The relaxation time, τ has units of femtoseconds and is related to the electron-ion coupling by $\gamma_p = \frac{m}{\tau}$, where m was chosen to be the mass of a silicon atom. Electronic density, ρ_e has units electron/Å^3. The metallic heat capacity function is $F(T_e) = 0.00595 \cdot \tanh(0.0002 \cdot T_e)$ as used in reference [27].

The electronic subsystems SCI-IV were chosen to represent a "semiconductor" electronic subsystem with parameters selected by the method described in Section 2.1. The metallic subsystems MI-III were chosen to represent a metallic electronic subsystem based on the α-Fe electronic subsystem of reference [27]. This value was chosen by using the Wiedermann-Franz law based on the electronic conductivity of α-Fe at 300 K. Based on the density of a FCC α-Fe, an electronic density of 2.16 electrons/A^3 was used. The weak subsystems WI,II were chosen to have one-hundredth of the thermal capacity and one-hundredth the electron-ion coupling the SCI semiconductor system. For SCI, SCII, MI, and WI, the electronic temperature of each cell was initialized to the local temperature of the atoms in the grid cell. For MII, the electronic temperature was initialized so that each grid cell had the same electronic thermal energy as SCI. (Note that this is true for MI, as well, due to the choice of heat capacity.) For SCIII, SCIV, MIII, and WII, the electronic temperature of each cell was initialized to 300K.

For comparison, two more systems were considered, a system with no electronic system, ("No electrons") and a system where the heat sink Langevin Thermostat is coupled to every atom in the system ("Quench") with a damping parameter of 33.33 fs.

2.3. Characterizing Defect vs Crystal Atoms. To characterize the effect of the different electronic subsystems coupled to the atomic subsystem, we needed to distinguish between regions of the material that are damaged and regions that are crystalline. We used a variation on the local bond order analysis method [1, 29]. We compute the correlation function of the

normalized local bond order parameter q_8 vectors, or $\alpha_j = q_8(i) \cdot q_8(j)$, for each silicon atom j in the coordination shell of a silicon atom i. If more than a fraction f of the atoms in the coordination shell of an atom i are such that α_j exceeds a threshold value α_{thresh}, then that atom is designated a crystalline atom. Otherwise the atom may be in a liquid region or may be an interstitial defect, but is generically designated a "defect" atom. We define the coordination shell to be all silicon atoms within 3.6 Å of each silicon atom (≈ 4 atoms), corresponding to the first peak of the radial distribution function, and use $\alpha_{thresh} = 0.8$ and $f = 0.75$. In practice, we found that the defect atoms remained clustered at the center of the simulation cell in a localized liquid/amorphous region.

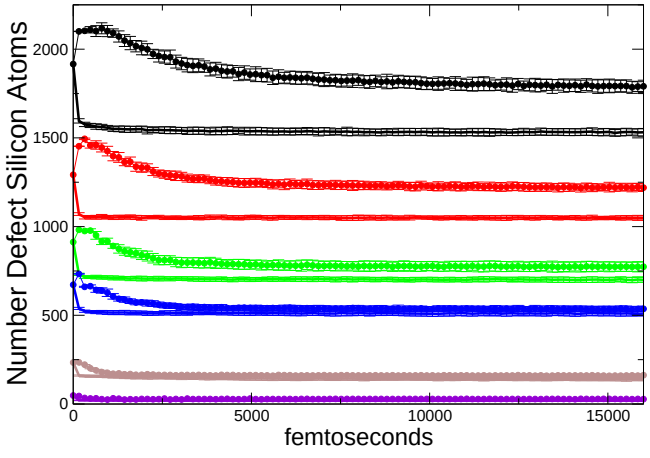
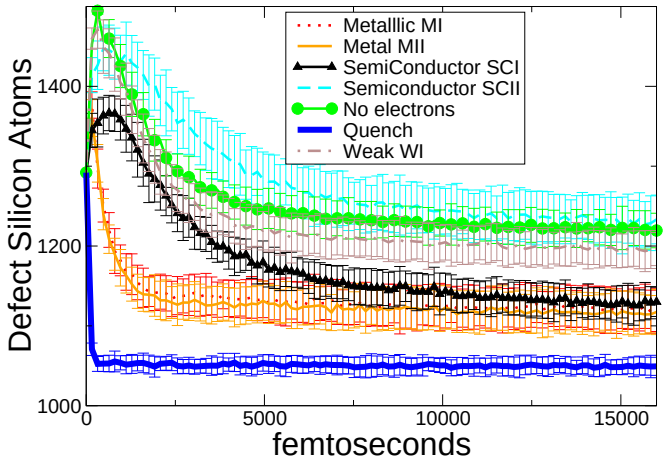


FIG. 3.1. *Annealing response in α-quartz to different degrees of initial damage. Circles represent no electron subsystem, straight line represents quenching the system.*

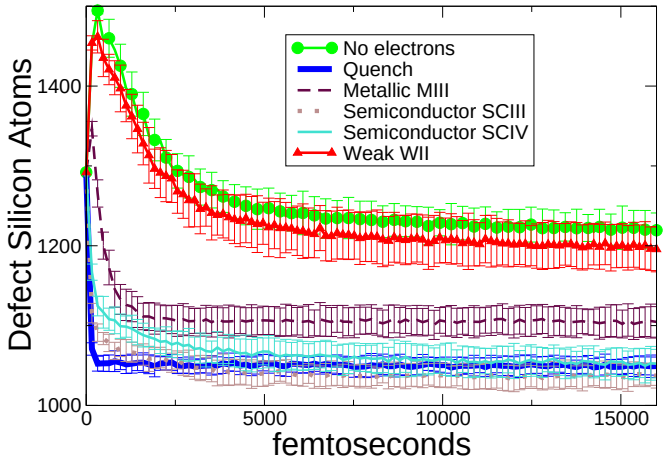
3. Results. In Figure 3, we can see the result of different initial damage in the system. Like the glass-forming binary LJ system of reference [25], permanent damage was sustained in the α-quartz after annealing, and the degree of damage was less over the entire simulation duration if heat was removed faster (i.e., "Quench"). Unlike the glassing-forming binary LJ system, even the smallest damage tested (48 defect silicon atoms) did not anneal completely (19 permanent defect atoms for a "Quench" system, 28 permanent defect atoms for a "No electron system").

In Figure 3.2(a), we see the impact on annealing of damage in the α-quartz of the different electronic subsystems. It is apparent that the presence of an electronic subsystem does have a significant impact on the healing of the system.

The primary way the electronic subsystem affects the annealing of damage appears to be in the added local specific heat. As the electronic subsystems in Figure 3.2(a) were initialized with a thermal energy above the equilibrium temperature, the "no electron" case, comparatively, has the least amount of energy to dissipate at initiation of the simulation. And yet, we observe that the "no electron" case has among the largest amount of damage for the α-quartz. The increased initial and permanent damage for "no electron-subsystem" can be seen in the LJ and BLJ systems of [25] as well. The SCII electronic subsystem has the same additional local heat capacity as the SCI electronic subsystem, but, also has a lower electron-ion cou-



(a) Different Electronic Subsystems with energy deposited in the subsystem



(b) Initially ambient electronic susbsystems

FIG. 3.2. *Influence of Electronic subsystems on annealing of α-quartz damage*

pling, making its additional heat capacity inaccessible to the lattice over the time of the heat flow evolution.

To illustrate this further, we consider the difference in the atomic temperature of the grid cells between the system with a "Weak" electronic subsystem, and a system with semiconductor SCI electronic subsystem in Figure 3. While the average atomic temperature of all the grid cells is lower for the "Weak" system versus the "Semiconductor" system, at increasing distance from the damage spot, the grid cells are hotter in the "Weak" system. These hotter

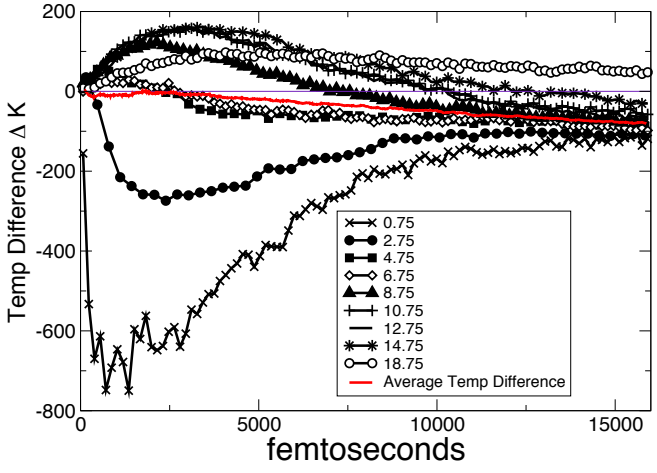


FIG. 3.3. *Difference in the atomic temperature between grid cells (labeled by squared distance in grid cell lengths from the center of the heat deposit) between the Weak W1 and Semiconductor SC1 system. Although the temperature of W1 is less on average, a substantial fraction of the cells are warmer in the system with less deposited energy due the smaller per cell net heat capacity.*

cells represent the part of the lattice where new damage is occurring over the simulation, or local reversible damage is becoming irreversible. In reference [25], we found that permanent damage in a glass forming material was correlated with significant displacement from the original lattice position. The added heat capacity of the SCI electronic subsystem removes kinetic energy from the atoms and lowers the amount of "irreversibly damaged" atoms. In fact, even the "Weak" electronic subsystem had reduced permanent damage relative to the "No electrons" case.

For MII, the specific heat per volume of the metallic electronic subsystem is approximately linear over the temperature range considered. It is half of that of the semiconductor at room temperature and twice that of the semiconductor at 1100K. This functionality is considered to be a more accurate model for a metallic electronic heat capacity. However, the functional details of the heat capacity for the two metallic electronic subsystems MI and MII appears to have little impact on the annealing of the system. The metallic electron subsystem also has a 1/50th the electron-ion coupling of SCI and 1/5th electron-ion coupling of SCII. It is apparent that substantial electronic conductivity of the metallic electronic subsystem compensates for the reduced electron-ion coupling and helps anneal the damage faster.

In Figure 3.2(b), electronic subsystems were attached to the α-quartz that were initialized to 300K (i.e. no energy deposited in the electronic subsystem). The electronic subsystem now acts to absorb the kinetic energy from the damaged region and results in the lowest amount of permanent damage. The degree of permanent damage in the α-quartz is now directly inversely correlated with the electron-ion coupling. The role of the heat capacity in the annealing of the damage suggests the details of the "heat sink" at the corner of the simulation may not be important.

4. Conclusions. A TTM has been applied to a model of damage in a SiO_2 α-quartz system, parameterized based on a VASP quantum simulation of electrons in an α-quartz repeat

cell. We find that the thermal conductivity of an electronic subsystem for this SiO_2 small enough to be discounted, however, that the relaxation time of the electronic system is 5-50× shorter than a metallic system. The heat capacity of the semiconductor electron subsystem is comparable to the that of the metallic system.

The influence of an electronic subsystem on local damage is proportional to the temperature difference between the electronic and atomic subsystems, the electron-ion coupling strength, and the heat capacity of the electronic subsystem. For example, the metallic electronic subsystem has a weaker electron-ion coupling, but due to a significant thermal conductivity (rapidly cooling the local hot electronic regions), results in the same amount of permanent damage as semiconductor subsystem with a tighter coupling and no thermal conductivity.

A cold electronic subsystem with even a weak coupling can significantly reduce the permanent damage. This is not because the electronic subsystem provides a "short circuit" to a heat sink or the thermal sink of the rest of the lattice, but because the added local heat capacity lowers the local peak temperature which reduces the net diffusion of the local atoms and reduces the net irreversible damage to the lattice.

For the systems where the electronic subsystem was initialized to the local atomic temperature, the amount of permanent damage for the semiconductor electronic subsystem depended strongly on the strength of the electron-ion coupling.

This coarse-grained model of energy transfer between electronic and atomic subsystems in α-quartz shows that the electronic subsystem plays a significant role in mediating the amount of damage that results from a heat deposition event and needs to be included in models of radiation damage in SiO_2 to understand resultant transient and residual damage.

REFERENCES

[1] S. Auer and D. Frenkel, *Numerical prediction of absolute crystallization rates in hard-sphere colloids*, The Journal of Chemical Physics, 120 (2004), pp. 3015–3029.

[2] J. Badro, P. Gillet, and J.-L. Barrat, *Melting and pressure-induced amorphization of quartz*, EPL (Europhysics Letters), 42 (1998), pp. 643–648.

[3] P. E. Blochl, *Projector augmented-wave method*, Phys. Rev. B, 50 (1994), p. 17953.

[4] A. Caro and M. Victoria, *Ion-electron interaction in molecular-dynamics cascades*, Phys. Rev. A, 40 (1989), pp. 2287–2291.

[5] A. Caro, M. Victoria, T. Diaz de la Rubia, and M. W. Guinan, *The effect of electronic energy loss on the dynamics of thermal spikes in Cu*, Journal of Materials Research, 6 (1991), pp. 483–491.

[6] J. Chen, D. Tzou, and J. Beraun, *A semiclassical two-temperature model for ultrafast laser heating*, International Journal of Heat and Mass Transfer, 49 (2006), pp. 307 – 316.

[7] J. K. Chen, J. E. Beraun, L. E. Grimes, and D. Y. Tzou, *Modeling of femtosecond laser-induced non-equilibrium deformation in metal films*, International Journal of Solids and Structures, 39 (2002), pp. 3199 – 3216.

[8] D. M. Duffy and A. M. Rutherford, *Including the effects of electronic stopping and electron–ion interactions in radiation damage simulations*, Journal of Physics: Condensed Matter, 19 (2007), p. 016207 (11pp).

[9] M. W. Finnis, P. P. Agnew, and A. J. E. Foreman, *Thermal excitation of electrons in energetic displacement cascades*, Phys. Rev. B, 44 (1991), pp. 567–574.

[10] F. Gao, D. J. Bacon, P. E. J. Flewitt, and T. A. Lewis, *The effects of electron-phonon coupling on defect production by displacement cascades in α-iron*, Modelling and Simulation in Materials Science and Engineering, 6 (1998), pp. 543–556.

[11] H. Hakkinen and U. Landman, *Superheating, melting, and annealing of copper surfaces*, Phys. Rev. Lett., 71 (1993), pp. 1023–1026.

[12] M. Head-Gordon and J. C. Tully, *Molecular dynamics with electronic frictions*, The Journal of Chemical Physics, 103 (1995), pp. 10137–10145.

[13] D. S. Ivanov and L. V. Zhigilei, *Combined atomistic-continuum modeling of short-pulse laser melting and disintegration of metal films*, Phys. Rev. B, 68 (2003), p. 064114.

[14] D. S. Ivanov and L. V. Zhigilei, *Kinetic limit of heterogeneous melting in metals*, Physical Review Letters, 98 (2007), p. 195701.

[15] H. O. Jeschke, M. S. Diakhate, and M. E. Garcia, *Molecular dynamics simulations of laser-induced damage of nanostructures and solids*, Applied Physics A: Materials Science & Processing, (2009).

[16] L. Koči, E. M. Bringa, D. S. Ivanov, J. Hawreliak, J. McNaney, A. Higginbotham, L. V. Zhigilei, A. B. Belonoshko, B. A. Remington, and R. Ahuja, *Simulation of shock-induced melting of Ni using molecular dynamics coupled to a two-temperature model*, Physical Review B (Condensed Matter and Materials Physics), 74 (2006), p. 012101.

[17] G. Kresse and J. Furthmller, *Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set*, Comput. Mat. Sci, 6 (1996), p. 15.

[18] ———, *Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set*, Phys. Rev. B, 54 (1996), p. 11169.

[19] G. Kresse and J. Hafner, *Ab initio molecular dynamics for liquid metals*, Phys. Rev. B, 47 (1993), p. 558.

[20] ———, *Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium*, Phys. Rev. B, 49 (1994), p. 14251.

[21] G. Kresse and D. Joubert, *From ultrasoft pseudopotentials to the projector augmented-wave method*, Phys. Rev. B, 50 (1999), p. 1758.

[22] Z. Lin, L. V. Zhigilei, and V. Celli, *Electron-phonon coupling and electron heat capacity of metals under conditions of strong electron-phonon nonequilibrium*, Physical Review B (Condensed Matter and Materials Physics), 77 (2008), p. 075133.

[23] Y. Miyamoto, A. Rubio, and D. Tomanek, *Real-time ab initio simulations of excited carrier dynamics in carbon nanotubes*, Phys. Rev. Lett., (2006).

[24] J. P. Perdew and A. Zunger, *Self-interaction correction to density-functional approximations for many-electron systems*, Phys. Rev. B, 23 (1981), p. 5048.

[25] C. L. Phillips and P. S. Crozier, *An energy-conserving two-temperature model of radiation damage in single-component and binary lennard-jones crystals*, Journal of Chemical Physics, submitted, (2009).

[26] S. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, Journal of Computational Physics, 117 (1995), pp. 1 – 19.

[27] A. M. Rutherford and D. M. Duffy, *The effect of electron–ion interactions on radiation damage simulations*, Journal of Physics: Condensed Matter, 19 (2007), p. 496201 (9pp).

[28] L. Sandoval and H. M. Urbassek, *Influence of electronic stopping on sputtering induced by cluster impact on metallic targets*, Physical Review B (Condensed Matter and Materials Physics), 79 (2009), p. 144115.

[29] P. R. ten Wolde, M. J. Ruiz-Montero, and D. Frenkel, *Numerical calculation of the rate of crystal nucleation in a Lennard-Jones system at moderate undercooling*, The Journal of Chemical Physics, 104 (1996), pp. 9932–9947.

[30] B. W. H. van Beest, G. J. Kramer, and R. A. van Santen, *Force fields for silicas and aluminophosphates based on ab initio calculations*, Phys. Rev. Lett., 64 (1990), pp. 1955–1958.

[31] K. Vollmayr, W. Kob, and K. Binder, *Cooling-rate effects in amorphous silica: A computer-simulation study*, Phys. Rev. B, 54 (1996), pp. 15808–15827.

[32] Y. Wang and L. Kantorovich, *Nonequilibrium statistical mechanics of classical nuclei interacting with the quantum electron gas*, Physical Review B (Condensed Matter and Materials Physics), 76 (2007), p. 144304.

[33] L. V. Zhigilei and D. S. Ivanov, *Channels of energy redistribution in short-pulse laser interactions with metal targets*, Applied Surface Science, 248 (2005), pp. 433 – 439. 4th International Conference on Photo-Excited Processes and Applications.