SANDIA REPORT

SAND2011-8757
Unlimited Release
Printed November 2011

Efficient uncertainty quantification methodologies for high-dimensional climate land models

Khachik Sargsyan, Cosmin Safta, Robert Berry, Jaideep Ray, Bert Debusschere, and Habib Najm

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

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

Printed in the United States of America. This report has been reproduced directly from the best available copy.

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.osti.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/help/ordermethods.asp?loc=7-4-0#online

SAND2011-8757
Unlimited Release
Printed November 2011

Efficient uncertainty quantification methodologies for high-dimensional climate land models

Khachik Sargsyan, Cosmin Safta, Robert Berry,
Jaideep Ray, Bert Debusschere, Habib Najm
Sandia National Laboratories, Livermore, CA
{ksargsy,csafta,rdberry,jairay,bjdebus,hnnajm}@sandia.gov

Abstract

In this report, we proposed, examined and implemented approaches for performing efficient uncertainty quantification (UQ) in climate land models. Specifically, we applied Bayesian compressive sensing framework to a polynomial chaos spectral expansions, enhanced it with an iterative algorithm of basis reduction, and investigated the results on test models as well as on the community land model (CLM). Furthermore, we discussed construction of efficient quadrature rules for forward propagation of uncertainties from high-dimensional, constrained input space to output quantities of interest. The work lays grounds for efficient forward UQ for high-dimensional, strongly non-linear and computationally costly climate models. Moreover, to investigate parameter inference approaches, we have applied two variants of the Markov chain Monte Carlo (MCMC) method to a soil moisture dynamics submodel of the CLM. The evaluation of these algorithms gave us a good foundation for further building out the Bayesian calibration framework towards the goal of robust component-wise calibration.

Acknowledgment

This work was supported by the US Department of Energy, Office of Science, under the project "Climate Science for a Sustainable Energy Future", funded by the Biological and Environmental Research (BER) program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Contents

List of Figures

List of Tables

Part I

Crosscutting UQ

This page intentionally left blank.

Chapter 1

Polynomial Surrogate Construction and Dimensionality Reduction

We propose and implement a methodology for surrogate model construction that approximates the input-output relationship in a computationally intensive forward model. The surrogate model will provide an inexpensive alternative to the complex model in both forward uncertainty quantification studies and in inverse problems where many forward model runs are required to infer reliable estimates of input parameters. The methodology is then illustrated for test problems. It has also been applied to global sensitivity studies in the Community Land Model (CLM).

1.1 Polynomial Surrogate Construction

The key to performing both forward and inverse UQ on computationally costly models is the construction of a *surrogate* model that mimicks the input-output relationship. The surrogate can be evaluated quickly, thus allowing both efficient global sensitivity analysis strategies and parameter calibration studies without having to run the complex forward model itself impractically many times.

1.1.1 Polynomial Chaos

Polynomial Chaos (PC) spectral expansions are employed to represent the dependence of a Quantity of Interest (QoI) y on an input parameter vector $\boldsymbol{\lambda}$ [7, 26], and to serve as a surrogate to a computationally intensive forward model. In particular, it allows the input parameters to follow arbitrary distributions. Indeed, we can view each of the input parameters as random variables, and consequently, the output QoI is a random variable as well. This output random variable is expanded in terms of an orthogonal set of polynomials of a standard random variable with respect to the density of the latter. Here, we employ Legendre-Uniform (LU) PC expansions for simplicity, as well as for an interpretation of the expansion simply as a response surface or a polynomial fit. The Legendre polynomial with a multi-index $\boldsymbol{p} = (p_1, p_2, \ldots, p_d)$ is a multivariate polynomial function of d variables $(\eta_1, \eta_2, \ldots, \eta_d) = \boldsymbol{\eta}$ defined by

$$\Psi_{\boldsymbol{p}}(\boldsymbol{\eta}) = \psi_{p_1}(\eta_1)\psi_{p_2}(\eta_2)\cdots\psi_{p_d}(\eta_d), \tag{1.1}$$

where $\psi_{p_i}(\eta)$ is the standard one-dimensional Legendre polynomial of degree p_i, for $i = 1, 2, \ldots, d$. The sum of all degrees $p_1 + p_2 + \cdots + p_d$ is called the order of the multidimensional Legendre polynomial (1.1).

Furthermore, the QoI y is represented with a polynomial expansion

$$y \approx y_{\boldsymbol{c}}(\boldsymbol{\eta}) \equiv \sum_{k=0}^{K} c_k \Psi_k(\boldsymbol{\eta}), \tag{1.2}$$

where the scalar subscript k typically corresponds to the graded lexicographic ordering of the multiindices $\boldsymbol{p}$ [4]. The expansion (1.2) retains only polynomials of order up to l, i.e. $p_1 + p_2 + \cdots + p_d \leq l$, leading to a total of $K + 1 = (d + l)!/(d!l!)$ number of terms in the truncated series (1.2). More generally, for a set of multi-indices $\mathcal{S}$, one can write

$$y \approx y_{\boldsymbol{c}}(\boldsymbol{\eta}) \equiv \sum_{\boldsymbol{p} \in \mathcal{S}} c_{\boldsymbol{p}} \Psi_{\boldsymbol{p}}(\boldsymbol{\eta}) = \sum_{k=0}^{K} c_k \Psi_k(\boldsymbol{\eta}), \tag{1.3}$$

where the single index k corresponds to *some* ordering of the multi-indices and $K + 1 = |\mathcal{S}|$.

Typically, the input parameter vector $\boldsymbol{\lambda}$ and the random variable vector $\boldsymbol{\eta} \sim \text{Uniform}[-1, 1]$ are related via the cumulative distribution function (CDF) $F_{\lambda_i}(\cdot)$ of each input parameter, assuming they are independent,

$$\eta_i = 2F_{\lambda_i}(\lambda_i) - 1, \qquad \text{for } i = 1, 2, \ldots, d. \tag{1.4}$$

For example, when λ_i are assumed uniform on their respective intervals $[a_i, b_i]$, one obtains

$$\eta_i = \frac{2}{b_i - a_i} \left(\lambda_i - \frac{a_i + b_i}{2} \right). \tag{1.5}$$

Given simple maps (1.4) or (1.5), and without loss of generality, one can identify $\boldsymbol{\lambda}$ with $\boldsymbol{\eta}$. In the case of dependent input parameters, or in presence of additional constraints on input parameters, one can generalize the CDF transformation (1.4) and utilize the Rosenblatt transformation [18] to obtain a set of independent uniform random variables as an input.

The problem of building the response surface that represents the input-output relationship now reads as follows: given training model runs $\mathcal{D} = \{(\boldsymbol{\eta}_i, y_i)\}_{i=1}^{N}$, build a polynomial representation (1.3), *i.e.* find PC mode vector $\boldsymbol{c}$.

1.1.2 Bayesian Compressive Sensing

Bayesian methods are well suited to deal with incomplete, sparse information [20]. Typically, the outcome of a Bayesian approach consists of a posterior probability distribution, describing our

knowledge of the quantities under study. Bayes formula in the context of inferring a PC expansion for the quantities of interest, based on available data $\mathcal{D}$, can be written as

$$q(\boldsymbol{c}) \propto L_{\mathcal{D}}(\boldsymbol{c})p(\boldsymbol{c}) \tag{1.6}$$

Here the likelihood $L_{\mathcal{D}}(\boldsymbol{c})$ is a measure of a goodness-of-fit of the polynomial representation to the data. We will assume a gaussian noise model with standard deviation σ to write

$$L_{\mathcal{D}}(\boldsymbol{c}) = (2\pi\sigma^2)^{-N/2} \exp\left(-\sum_{i=1}^{N} \frac{(y_i - y_{\boldsymbol{c}}(\boldsymbol{\eta}_i))^2}{2\sigma^2}\right) \tag{1.7}$$

The prior distribution $p(\boldsymbol{c})$ incorporates any prior information on the object of inference, i.e. the PC mode vector $\boldsymbol{c}$. The posterior distribution $q(\boldsymbol{c})$ is the main outcome of the inference process, and it corresponds to the current knowledge about the inferred values of $\boldsymbol{c}$ *given* the data set $\mathcal{D}$.

While in principle, the Bayesian procedure outlined above could be used to determine the full vector of coefficients $\boldsymbol{c}$ of all basis functions, this is in practice not always feasible. If the QoI y depends on many parameters, then its PC expansion (1.3) will be very high dimensional, and if the forward model is computationally expensive to evaluate, then the number of samples required to determine all terms in this expansion would be prohibitively expensive. Instead, quite often one is given a fixed number of samples at random locations in the parameter space, and the task becomes to determine the best possible PC representation given the available data. To this goal, we rely on Bayesian Compressive Sensing (BCS) to determine a sparse set of basis functions that is best supported by the data, as outlined below.

The key in inferring a sparse set of PC modes is the usage of *sparsity* priors that "encourage" the modes to have nearly vanishing values, unless there is strong support in the data for those PC modes. This leads to a sparse set of basis functions. A common sparsity prior is the Laplace prior of a form

$$p(\boldsymbol{c}) = (\lambda/2)^{K+1} \exp\left(-\lambda \sum_{k=0}^{K} |c_k|\right). \tag{1.8}$$

In this case the *maximum a posteriori* (MAP) estimate of the object of inference $\boldsymbol{c}$, i.e. the vector $\boldsymbol{c}$ that maximizes the posterior $q(\boldsymbol{c})$, coincides with the solution of the optimization problem

$$\arg\max_{\boldsymbol{c}} \left(\log L_{\mathcal{D}}(\boldsymbol{c}) - \lambda ||\boldsymbol{c}||_1\right) \tag{1.9}$$

Clearly, the prior distribution corresponds to the l_1 regularization term. The optimization problem (1.9) corresponds to the classical compressive sensing algorithm that is extensively used in the signal processing community [**?**]. The positive parameter λ controls the relative importance of the penalty with respect to the goodness-of-fit. It is typically fixed at a user-defined value. In a hierarchical Bayesian setting, however, it can be endowed with a prior distribution and marginalized over in the posterior distribution.

However, the Laplace prior distribution (1.8) does not allow the computation of the posterior distribution in a closed form. Instead, following [**?**], we use a gaussian prior distribution of the form

$$p(\boldsymbol{c}) \propto \prod_{k=0}^{K} \exp\left(-\frac{c_k^2}{2s_k^2}\right). \tag{1.10}$$

Together with the likelihood (1.7), this choice of prior leads to a gaussian posterior distribution with mean and variance, respectively,

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Psi}^T\boldsymbol{y} \qquad \text{and} \qquad \boldsymbol{\Sigma} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi} + \boldsymbol{S})^{-1}, \tag{1.11}$$

where $\boldsymbol{\Psi}$ is a $N \times (K+1)$ matrix with entries $\boldsymbol{\Psi}_{ik} = \Psi_k(\boldsymbol{\eta}_i)$ and $\boldsymbol{S} = \text{diag}(\sigma^2/s_0^2, \ldots, \sigma^2/s_K^2)$.

The likelihood variance σ^2 and the prior variances $(s_0^2, \ldots, s_K^2)$ together form a vector of *hyperparameters*. Out of convenience, we will use a formal notation $\boldsymbol{s}^2$ for the vector of prior variances. In principle, one can construct a hierarchical Bayesian formulation with appropriate conjugate priors for σ^2 and $\boldsymbol{s}^2$ to obtain a closed form for the posterior distribution of $\boldsymbol{c}$. Here, however, a less complicated approach will be taken. Namely, the hyperparameters will be fixed at the values that maximize the *evidence* or the integrated likelihood

$$E(\sigma^2, \boldsymbol{s}^2) = \int_{\mathbb{R}^{K+1}} L_{\mathcal{D}}(\boldsymbol{c}; \sigma^2) p(\boldsymbol{c}; \boldsymbol{s}^2) d\boldsymbol{c} \quad \propto \quad \sigma^{-1}|\boldsymbol{C}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{y}^T\boldsymbol{C}^{-1}\boldsymbol{y}\right), \tag{1.12}$$

where $\boldsymbol{C} = \boldsymbol{I} + \boldsymbol{\Psi}\boldsymbol{S}^{-1}\boldsymbol{\Psi}^T$.

The maximization procedure for $E(\sigma^2, \boldsymbol{s}^2)$ essentially links Bayesian regression with the Relevance Vector Machine (RVM) technique. An iterative procedure for maximizing $E(\sigma^2, \boldsymbol{s}^2)$ [?] leads to very small values for s_k^2 for some k's, indicating that the evidence is maximized when the prior for the corresponding coefficients becomes a delta function around 0, i.e. the corresponding coefficients should be set to 0. The corresponding basis polynomials $\Psi_k(\cdot)$ are then dropped from the basis set. Therefore, the procedure automatically detects and retains the most important or relevant basis terms. While the BCS tolerance parameter allows detecting small s_k^2 values, in practice, an additional down-selection is needed by retaining the first K_b terms from the list of basis terms selected by BCS. This allows direct control on the number of basis terms retained, in case one needs to avoid overfitting or needs to have a bound on the basis set to meet computational budget constraints.

1.1.3 Iterative procedure with BCS

With a total order truncation and in the presence of a large number of dimensions, one can not afford to build an initial PC basis of order greater than two. Here we propose an iterative procedure that allows increasing the order for the relevant basis terms while maintaining the dimensionality reduction. Namely, given a multi-index set $\mathcal{S}$ corresponding to the current basis, we add a basis term only if it is *admissible*, i.e. if by subtracting an order from each non-zero dimension one never obtains a multi-index outside the set $\mathcal{S}$. In other words,

$$\boldsymbol{p} = (p_1, \ldots, p_d) \text{ is added to } \mathcal{S}, \text{ if } \boldsymbol{p} - \boldsymbol{e}_i \in \mathcal{S}, \text{ for all } i = 1, \ldots, d, \tag{1.13}$$

where $\boldsymbol{e}_i = (0, \ldots, 1, \ldots, 0)$ with 1 in the i-th position. The full algorithm then reads as follows:

- Step 0. Let $\tilde{\mathcal{S}}$ be a set of multi-indices with a total order $\leq l_0$, where l_0 is the initial PC order,

- Step 1. Run the BCS algorithm to reduce the current basis set $\tilde{\mathcal{S}} \to \mathcal{S}$. If needed, retain only first K_b terms,
- Step 2. Enrich the current basis by all admissible basis terms and call the new basis multi-index set $\tilde{\mathcal{S}}$. Repeat from Step 1 until the maximal order l is reached.

The resulting representation reads as follows:

$$\hat{y}(\boldsymbol{\eta}) = \sum_{\boldsymbol{p}\in\mathcal{S}} c_{\boldsymbol{p}} \Psi_{\boldsymbol{p}}(\boldsymbol{\eta}), \tag{1.14}$$

where the PC modes $\boldsymbol{c_p}$ are described by a multivariate gaussian posterior of a dimensionality that is equal to the cardinality of $\mathcal{S}$.

1.1.4 Error measure

We will consider two error measures, the goodness-of-fit of the resulting representation at the $N = N_t$ training points or at randomly chosen, N_v *validation* points. In particular, we will rely on a relative L_2 error. More precisely, the validation error will be

$$E_v(\boldsymbol{c}) = \sqrt{\frac{\sum_{i=1}^{N_v} \left(y(\boldsymbol{\eta}_i) - y_{\boldsymbol{c}}(\boldsymbol{\eta}_i)\right)^2}{\sum_{i=1}^{N_v} y(\boldsymbol{\eta}_i)^2}}, \tag{1.15}$$

where the posterior mean PC mode vector is taken $\boldsymbol{c} = \{c_{\boldsymbol{p}\in\mathcal{S}}\}$ with some ordering of the multi-indices $\boldsymbol{p} \in \mathcal{S}$. The error at the training points $E_t(\boldsymbol{c})$ is defined similarly. Note that our construction leads to an *uncertain* response surface, since the polynomial representation coefficients $\boldsymbol{c}$ are associated with a posterior probability distribution.

1.2 Test Results

1.2.1 Test 1: Analytically tractable case

Below we describe a very simple test case, where the reduced basis can be seen a priori based on the forward function. Namely, consider a 3-dimensional function

$$f(x, y, z) = x^2 + xy^2 + z^3 \tag{1.16}$$

where the input vector is denoted by $\boldsymbol{\eta} = (x, y, z)$ for clarity of presentation. The function (1.16) can be represented *exactly* in Legendre polynomial basis with 6 terms only. Namely,

$$f(x, y, z) = \frac{1}{3}\psi_0 + \frac{1}{3}\psi_1(x) + \frac{2}{3}\psi_2(x) + \frac{3}{5}\psi_1(z) + \frac{2}{5}\psi_3(z) + \frac{2}{3}\psi_1(x)\psi_2(y), \tag{1.17}$$

where $\psi_i(\cdot)$ is the univariate Legendre polynomial of order i, or, in terms of a 'dummy' variable t,

$$\psi_0 = 1, \qquad \psi_1(t) = t, \qquad \psi_2(t) = \frac{1}{2}(3t^2 - 1), \qquad \psi_3(t) = \frac{1}{2}(5t^3 - 3t). \tag{1.18}$$

We run the BCS algorithm with an initial order $l_0 = 3$ and without any additional iterations. The 3-rd order, 3-dimensional polynomial basis corresponds to 20 basis terms. The algorithm correctly detects the only relevant terms. Namely, it detects and retains the same 6 basis terms that appear in the exact expansion (1.17).

1.2.2 Test 2: Number of training runs necessary for reliable results

Consider a test function

$$y = \exp\left(\sum_{i=1}^{d} a_i \eta_i\right), \tag{1.19}$$

where positive constants a_i are picked to correspond to the 'importance' of the i-th dimension. In other words, it is expected that the larger the value of a_i the more relevant basis terms along the i-th dimension are.

The *sparseness* of available data is an essential issue and in principle challenges the BCS procedure. The following experiment will help determine how many samples are required to reliably detect a lower dimensional structure in a high-dimensional data set. Let us take first d_{imp} dimensions to be more important by an order of magnitude than the remaining $d - d_{\text{imp}}$ dimensions. Namely, we take

$$a_i = \begin{cases} 1 & \text{if } 1 \le i \le d_{\text{imp}}, \\ 0.1 & \text{if } d_{\text{imp}} < i \le d, \end{cases} \tag{1.20}$$

and run the BCS procedure with $l_0 = 1$, i.e. first order polynomial basis and a very generous tolerance. The goal of this simple test is to check whether the BCS algorithm picks the already-known important dimensions before the rest of the dimensions. For additional robustness, we generate 10 different sample sets each with N_t points, and declare success only if all 10 replica tests have detected the first d_{imp} important dimensions correctly.

Table 1.1. Number of training runs needed to reliably detect a lower-dimensional structure.

d_{imp}	d	N_t
1	any	20
2	any	50
5	any	130
10	any	980

In Table 1.1, we report the estimated number of training samples that are necessary for a reliable detection (i.e. no failures in 10 replica runs) of the important dimensions for different values of the number of important dimensions d_{imp}. The total number of dimensions d, as it turns out, is not important. Clearly, the number of training runs necessary to detect low-dimensional structures in the forward model depends on the dimensionality of these low-dimensional structures only, and is independent of the total number of dimensions.

1.2.3 Test 3: Convergence for various model sparsities

Consider now a somewhat more realistic example, where the dimensional importances a_i in (1.19) change in a more gradual manner. Let us take

$$a_i = \left(\frac{i}{d}\right)^M . \tag{1.21}$$

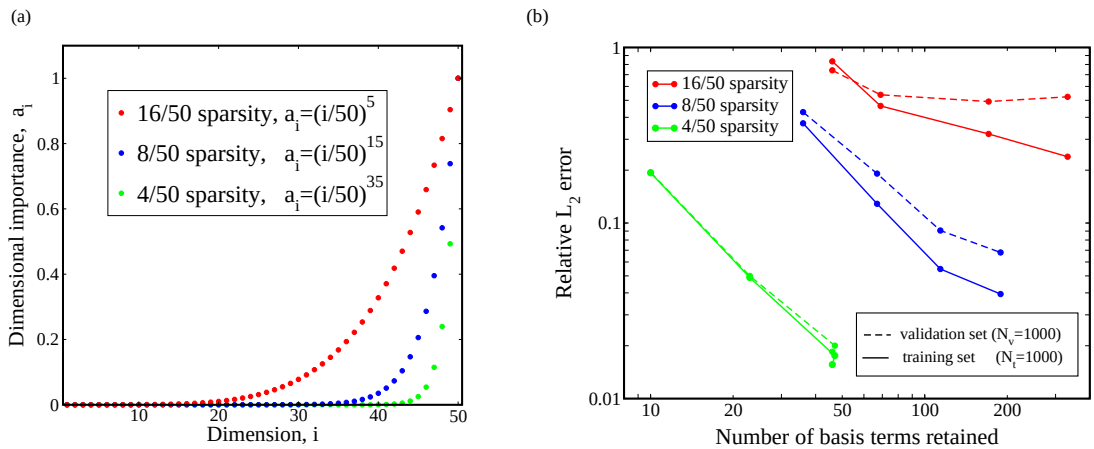


Figure 1.1. (a) Dimensional importances chosen for three test problems with varying degree of model sparsity. (b) The relative L_2 errors as functions of the basis cardinality for all three test problems computed at the training points and at the validation points. At every iteration, the basis is enriched by adding all the admissible basis terms of one order higher, up to fourth order. Note the logarithmic scale on both axes.

We fix the total dimensionality for this experiment at $d = 50$ and vary M. In particular, we picked three different values for $M = 5, 15, 35$, as illustrated in Figure 1.1(a). Clearly, for larger M, more relative importance is given to a fewer number of dimensions, i.e. the effective lower-dimensional space is smaller. Let us introduce a model *sparsity* measure as the number of dimensions that contribute to $\sim 90\%$ of the total sum of a_i's. It can be checked that the $M = 5, 15, 35$ cases

correspond to sparsities of $16, 8$ and 4, respectively. For example, for $M = 35$,

$$\frac{\sum_{i>d-4}(i/d)^M}{\sum_{i=1}^{d}(i/d)^M} \approx 0.9. \tag{1.22}$$

As Figure 1.1(b) illustrates, a model with better sparsity (i.e., with a smaller number of important dimensions) is represented better with the same number of training runs. The relative L_2 error on a validation set is a stricter test for the PC representation and it often indicates overfitting. That is, the relative error on the training points tends to decrease when more basis terms (degrees of freedom) are included, while the validation error $E_v(\boldsymbol{c})$ could be increasing when more than necessary terms are used to represent the current set of data.

1.2.4 Test 4: Dimensionality sorting study

Let us focus on the $M = 15$ case and illustrate further how the BCS algorithm detects the important dimensions. The dialed-in dimensional importances 1.21 are randomly shuffled for the purpose of illustration. The BCS algorithm is run only up to first order, with a very generous tolerance. Therefore, eventually, all the dimensions are picked as relevant. However, the RVM optimization procedure clearly shows the order of importance of the dimensions: the faster the iterative procedure for s_k^2 converges, the more important the k-th basis term is. Figure 1.2 shows the values of dimensional importances, as well as the order by which the algorithm picks them. With a stricter threshold, the BCS algorithm would stop earlier according to the red line shown in the figure. This proof-of-concept demonstrates that the BCS algorithm detects the dimensional importances in the expected sequence.

1.2.5 Test 5: Study of the dependence on the total dimensionality

The following test case is meant to study the dependence of the accuracy of the final representation on the dimensionality d of the input space. In particular, we used the test function (1.19) with dimensional importances dialed-in according to 1.21 with $M = 10$ and varying d. Once again this demonstrates that while 1000 training points is sufficient to represent a 10-dimensional problem with a reasonable low-dimensional structure well, it is somewhat satisfactory for a 30-dimensional problem, and fails to a certain degree, i.e. overfits, when trying to build a representation for a 50-dimensional problem.

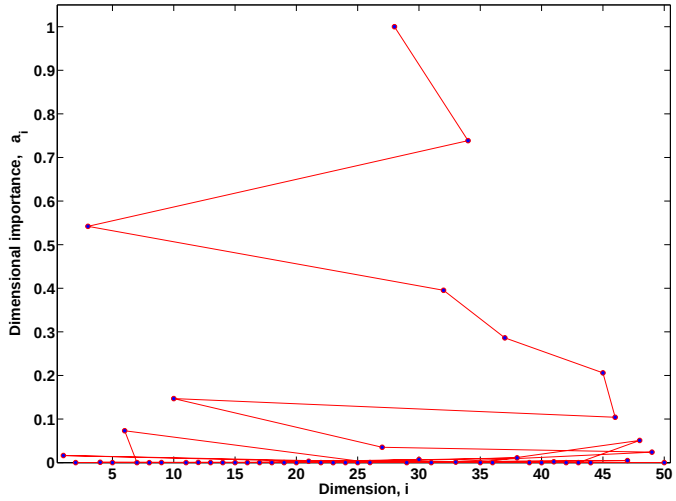


Figure 1.2. Results of a first order BCS procedure with a 50-dimensional test function and $N_t = 1000$ training samples. The dimensional importances are shown with blue dots, while the sequence in which the procedure picks the important dimensions is highlighted by the red lines joining the dots, starting from the top.

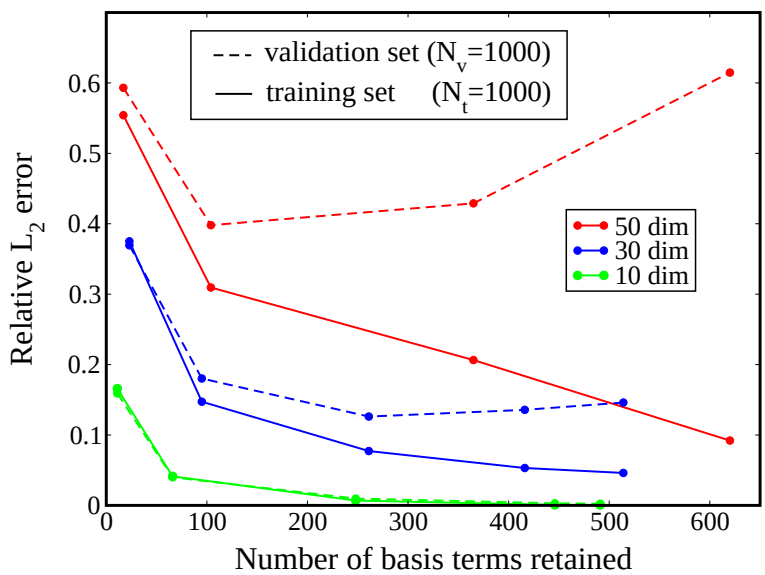


Figure 1.3. Relative L_2 errors as functions of the basis cardinality for a test problem computed at the training points and at the validation points. At every iteration, the basis is enriched by adding all the admissible basis terms of one order higher, up to fifth order.

This page intentionally left blank.

Chapter 2

Efficient Sampling in Irregular Domains

2.1 Grids in Regular Domains

Building a PC surrogate for the CLM model requires numerical approximation of numerous projection integrals, and utilizes multidimensional quadrature rules. Such multidimensional numerical quadrature is often built up from one-dimensional quadrature rules. A direct application of iterated integration leads to a multidimensional tensor-product grid. More sophisticated combinations of the univariate grids can lead to sparse grids which improve the efficiency of the multivariate quadrature for a given accuracy. This section will begin with a discussion of one-dimensional grids and conclude with a discussion on the Smolyak construction of grids tailored for projecting onto a prescribed basis.

2.1.1 One-dimensional Gauss-Patterson grids

The multidimensional rectangular grid sparse quadrature is based on a series of one-dimensional Gauss-Patterson [15] quadrature rules. These formulae are nested, and it was shown that addition of $n+1$ points to an n-point formulae yeilds an accuracy of degree approximately $3n$. The methodology for computing the recursively nested Gauss formulae is described below.

Let an n-point quadrature formula be augmented with p additional points, and let G_{n+p} by the polynomial whose roots are the $n+p$ abscissae of the new quadrature rule. A general polynomial of degree $n+2p-1$ can be expressed as

$$F_{n+2p-1}(x) = Q_{n+p-1}(x) + G_{n+p}(x) \sum_{k=0}^{p-1} c_k P_k(x) \tag{2.1}$$

where Q_{n+p-1} is a general polynomial of degree $n+p-1$ and P_k is the Legendre polynomial of order k. Since Q_{n+p-1} can always be integrated exactly by a $n+p$-point quadrature, if

$$\int G_{n+p}(x) P_k(x) dx = 0, \;\; k = 0, 1, \ldots, p-1 \tag{2.2}$$

then all polynomial of degrees $n+2p-1$ can be integrated exactly by the $n+p$-point quadrature.

We start deriving the $n+p$ formulae by first expanding G_{n+p} as

$$G_{n+p}(x) = \sum_{k=0}^{n+p} c_k P_k(x) \tag{2.3}$$

This expression is subsituted into eq. 2.2 leading to

$$\sum_{i=0}^{n+p} c_i \int P_i(x) P_k(x) dx, \;\; k = 0, 1, \ldots, p-1 \tag{2.4}$$

This implies, due to orthogonality of Legendre polynomials that $c_k = 0$ for $k = 0, 1, \ldots, p-1$. Thus G_{n+p} can be rewritten as

$$G_{n+p}(x) = \sum_{k=1}^{[n/2]+1} c_k P_{2k-2+p+q}(x) \tag{2.5}$$

where $q = n - 2[n/2]$. Since the original abscissae of the n-point formula, x_j, $j = 1, 2, \ldots, n$ are also the roots of G_{n+p}, a linear system can be assembled to compute the first $[n/2]$ c_k coefficients while the last coefficient $c_{[n/2]+1}$ can be arbitrarily set to 1.

$$\sum_{k=1}^{[n/2]} c_k P_{2k-2+p+q}(x_j) = -P_{n+p}(x_j), \;\; j = 1, 2, \ldots, [n/2] \tag{2.6}$$

Once the coefficients c_k are computed, the quadrature points are the roots of G_{n+p} given by expression (2.5).

The weights of the associated with the new $n+p$-point rule can be computed as [**?**]

$$w_j \propto \int L_j(x) dx, \;\; j = 0, 1, \ldots, n+p \tag{2.7}$$

where L_j is Lagrange interpolating polynomial of order $n+p-1$, $L_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n+p} \frac{x - x_i}{x_j - x_i}$.

The procedure above can be applied recursivelly, starting, for example, with an n-point Gauss-Legendre rule and adding $p = n+1$ abscissae at every iteration. If $n \to 2n+1$, then the resulting quadrature rule is of approximately $3n/2$ degree. The table below shows the abscissa and weights for Gauss-Patterson rules obtained starting from a 3-point Gauss-Legendre rule. Since the points are symmetric with respect to 0, only the positive abscissae and weights are shown. Please note that, for each recursion, the quadrature weights are computed for all abscissae.

Gauss-Legendre 3-point rule

$x:$	0	0.77460	
$w:$	0.88889	0.55556	

$+4$ points $\Rightarrow$ Gauss-Patterson 7-point rule

$x:$	0	0.43424	0.77460	0.96049
$w:$	0.45092	0.40140	0.26849	0.10466

$+8$ points $\Rightarrow$ Gauss-Patterson 15-point rule

$x:$	0	0.22339	0.43424	0.62110	0.77460	0.88846	0.96049	0.99383
$w:$	0.22551	0.21916	0.20063	0.17151	0.13442	0.092927	0.051603	0.017002

2.1.2 Multivariate Quadrature and the Smolyak Construction

One approach to multivariate quadrature is to use one-dimensional rules iteratively over the dimensions of the domain. This is equivalent to selecting a multivariate grid which is a tensor product of one-dimensional rules. If n points are used in each of d dimensions, then the integrand needs to be evaluated at n^d points. This is often many more than is needed for a given accuracy when the dimensionality d is high. Indeed, Smolyak introduced sparse grids which are based on a sparse tensor-product construction.

Sparse Grids

Let $\{(G_n, W_n)\}_{n\in I}$ be a sequence of one-dimensional grids (indexed by I) where grid G_k contains k points, with quadrature weights $W_k = (w_k^1, w_k^2, \ldots, w_k^k)$. Let α_k be the accuracy of (G_k, W_k), that is, quadrature on (G_k, W_k) has no error for polynomials of order α_k (or less).

A full-tensor product grid $T = \bigotimes_{i=1}^d (G_{k_i}, W_{k_i}) = \left(G^T, W^T\right)$ has precision $\alpha = (\alpha_{k_1}, \ldots, \alpha_{k_d})$, meaning that quadrature on T for each $[x_1^{a_1} x_2^{a_2} \cdots x_d^{a_d}]$ with all $\alpha_{k_i} \leq a_i$ has no error. In order for T to have accuracy α_n, then each k_i must be at least n. In particular, a full-tensor grid with accuracy α_n will have at least n^d points. For example, a full-tensor grid in 10 dimensions with accuracy 5 will require at least $3^{10} = 59,049$ points!

Smolyak demonstrated the construction of quadrature grids with much fewer points than full-tensor grids, each with the same accuracy. Consider two full-tensor grids $A = \left(G^A, W^A\right)$ and $B = \left(G^B, W^B\right)$, with precision α and β respectively. Let $\gamma = \min(\alpha, \beta)$ (the component-wise minimum), and let $C = (G_C, W_C) = \bigotimes_{i=1}^d (G_{c_i}, W_{c_i})$ be the full-tensor grid with precision γ. Then A, B, C can be combined into a quadrature rule $S = \left(G^S, W^S\right)$ with

$$G^S = G^A \cup G^B \cup G^C, \tag{2.8}$$

and the weights W^S on these points such that

$$Q_S = Q_A + Q_B - Q_C \tag{2.9}$$

and precision set

$$\sigma = \alpha \cup \beta = \{(p_1, \ldots, p_d) : \text{ each } p_i \leq \alpha_i \text{ or each } p_i \leq \beta_i\}. \tag{2.10}$$

If the G_n are nested (e.g. Gauss-Patterson or Clenshaw-Curtis) then G^S will have fewer points than the full-tensor product of precision $\max(\alpha, \beta)$.

This construction can be generalized to a sequence of full-tensor grids $\{A_j\}$. In particular, a sparse grid of accuracy α is the minimal union of tensor grids of accuracy at least α.

Polynomial Projection

A sparse tensor-product construction for nonintrusive spectral projection follows similarly. Each integral in the spectral projection

$$f = \sum_{\alpha} f_\alpha \, \varphi_\alpha \qquad f_\alpha = \langle f \, \varphi_\alpha \rangle \tag{2.11}$$

may utilize a distinct quadrature rule. In order for the projection to be exact when f is a polynomial of (multi-index) order α, the quadrature must be exact on all polynomials of order up to 2α. In particular, if it is known *a priori* that f can be represented on a prescribed basis in multi-indices $\{\alpha_i : i = 1 \ldots N\}$ then the projection can be computed efficiently on a (sparse) grid which is exact on indices

$$\sigma = \bigcup_i \alpha_i \tag{2.12}$$

using the notation in (2.10). This approach provides a grid for projection with many fewer points (in general) than using a single grid suitable for all projections simultaneously.

2.2 Grids in Irregular Domains

One of the primary assumptions in the above constructions of multivariate quadrature grids is that the domain of the integrand is regular, that is, it can be written as a tensor product of (one-dimensional) intervals. In some models, such as CLM, the domain is known to be irregular. One approach we have investigated is mapping the irregular domain to a regular domain via a Rosenblatt transformation. Another approach is to employ efficient quadrature in the irregular domain directly.

The idea is to find a quadrature rule (G, W) with grid $G = (x_1, \ldots, x_N\}$ (each $x_i \in \mathbb{R}^d$) and weights $W = (w_1, \ldots, w_N)$ which is exact on the span of some basis $\{\varphi_j : j = 1, \ldots, P\}$ on the

domain Ω,

$$\begin{aligned}
\sum_{i=1}^{N} \varphi_1(x_i)\, w_i &= \langle \varphi_1 \rangle \\
\sum_{i=1}^{N} \varphi_2(x_i)\, w_i &= \langle \varphi_2 \rangle \\
&\vdots \\
\sum_{i=1}^{N} \varphi_P(x_i)\, w_i &= \langle \varphi_P \rangle
\end{aligned} \tag{2.13}$$

such that $x_i \in \Omega$.

The exactness constraint provides a system of P equations and $N(d+1)$ unknowns. Informally, the quadrature (G, W) is said to be *efficient* if $P \approx N(d+1)$. Any additional constraints, such as requiring the weights w_i be positive, may lead to grids G which are less efficient (contain more points). While sparse grids are easy to construct, they are generally contain more grid points than efficient quadratures. For instance, the popular Clenshaw-Curtis sparse quadrature has $P = N$ for every d.

The general conditions under which an efficient quadrature exists is not known. Furthermore, when the dimensionality d is large, the nonlinear system (2.13) is difficult to solve, even for a regular domain. Finding efficient quadrature rules for high dimensional domains (both regular and irregular) is the subject of future work.

2.2.1 Efficient Quadratures in Smolyak Construction

The domain of CLM is irregular, but can be written as the tensor-product of a regular slice and a sequence of irregular slices

$$\Omega = \Omega_R \otimes \Omega_{I_1} \otimes \cdots \otimes \Omega_{I_K}. \tag{2.14}$$

This is due to constraints on some parameters. In particular, there are some triplets of parameters which are required to be positive and sum to one, which requires that triplet to be bounded by a triangular region in parameter space. In this case a quadrature grid can be constructed for the full domain Ω via a sparse tensor-product of efficient grids on each slice.

This approach may benefit from utilizing nested sequences of quadratures in the irregular domain slices. We would expect, however, that a nestedness constraint will come at the cost of efficiency (e.g. Gaussian quadrature grids are efficient in an interval, but the nested Gauss-Patterson grids need roughly twice as many points for the same order of accuracy). Investigation of the relative benefit of using nested quadratures is the subject of future work.

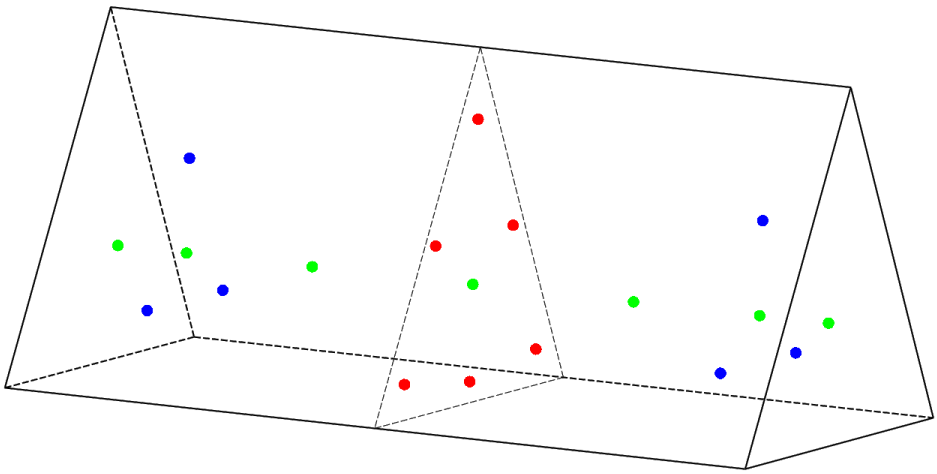


Figure 2.1. Sparse quadrature of total order five on a triangular prism $T \otimes I$ using efficient quadratures in the triangle slice T and Gauss-Patterson quadrature in the interval I. Each green point is a first-order triangle quadrature. Each set of three blue dots, together with the central green point, are third order grids. The six red points, together with the central green point, constitute a fifth-order grid in the triangle.

Chapter 3

Models Calibration Using Multichain Differential Evolution Monte Carlo Methods

3.1 Introduction

The aim of this chapter is to investigate multichain sampling methods as a means of fitting models to data, to estimate the models' parameters. Multichain methods for inverse problems, of which parameter estimation is a classical example, are a relatively new development. These methods allow the estimation of model parameters as distributions, which, in turn, allows the calculation of the parameters' uncertainties e.g., in the form of a mean value and a standard deviation. The multichain nature of the technique allows a conceptually simple path to parallelization, allowing their use with computationally expensive models (in relation to single-chain methods) e.g., coupled sets of submodels currently implemented in the Community Land Model version 4 (CLM4) [5]. However, multichain methods have not been tested with the kind of nonlinear models that are usually incorporated into Earth System Models, nor have their performance been compared with more established methods. Doing so forms the main thrust of this chapter. A secondary objective is to develop methods for model selection, e.g., given two competing models which are fit to the same data, which should one choose for further use (i.e., which one fits the data best)?

Calibration, or the fitting of models to data / observations / measurements to estimate model parameters, is part of any scientific activity. Typically, one estimates a single numerical value (also known as a point estimate) for the model parameters, usually by minimizing an L_2 norm of the discrepancy between model predictions (for a proposed set of model parameter values) and observations. However, within the context of uncertainty quantification, point estimates are rather irrelevant – the estimates usually do not provide any quantification of the uncertainty (e.g., in the form of "error bars") directly and consequently, are useless in a study of the parametric uncertainty of the model. Simply putting bounds on the variation of a model's parameters does not help; the independent treatment of parametric uncertainties lead to parameter combinations which are aphysical and for which the model may not be defined. Thus a probabilistic treatment of the model parameters, where parameters estimates are defined as probability density distributions (rather than point values) is desired. A distribution also allows "error bars" to be calculated easily; further, a joint distribution of parameters specifies the combinations of parameter values which are physi-

cally meaningful.

A Bayesian formulation of an inverse problem [13] is a common approach when model parameters are desired as distributions. A Bayesian inverse formulation combines a likelihood function (the probability $\mathcal{L}(\mathbf{y}^{obs}|\mathbf{\Theta})$ of observing the data $\mathbf{y}^{obs}$, given a certain value of model parameters, $\mathbf{\Theta}$) with prior beliefs regarding the value of the parameters, $\pi(\mathbf{\Theta})$, to obtain a posterior distribution of the model parameters, conditioned on the data

$$P(\mathbf{\Theta}|\mathbf{y}^{obs}) \propto \mathcal{L}(\mathbf{y}^{obs}|\mathbf{\Theta})\pi(\mathbf{\Theta}) \tag{3.1}$$

The posterior distribution $P(\Theta|\mathbf{y}^{obs})$ is constructed by drawing samples from it. Since $P(\Theta|\mathbf{y}^{obs})$ is usually an arbitrary distribution (as opposed to a well-known one like a Gaussian), specialized samplers are required. Markov chain Monte Carlo (MCMC) methods [8] are typically used for this purpose. Since each sample requires a model evaluation (which can be rather expensive in case of physics models), efficiency of sampling becomes a foremost concern. Adaptive MCMC methods, e.g., the Delayed Rejection Adaptive Metropolis (DRAM, [11]), are being increasingly used in parameter estimation, especially when the model is moderately intensive computationally (e.g., 1D and small 2D partial differential equations (PDEs) [14, 16]). However, DRAM draws its samples sequentially i.e., it is a *single-chain* algorithm, and the question arises whether the computational expense can be divided among multiple chains, which can subsequently be put on separate processors. Note that there is no restriction on the use of a *parallelized model* with DRAM; our focus, instead, is on partitioning the samples among processors.

Multichain methods: Multichain methods for solving Bayesian inverse problems are a class of global, stochastic optimization methods. In this study, we will restrict ourselves to the DrEAM algorithm ("Differential Evolution Adaptive Metropolis", [25]), which is a generalization of DE-MC ("Differential Evolution Markov Chain", [21]). Both DE-MC and DrEAM are related to a class of Population Monte Carlo methods (see the literature review in [21]), which have been successfully used in hydrology research [22–24]. In DrEAM, one simultaneously releases Markov chains from an over-dispersed set of points in the parameter space. These chains march in a manner similar to traditional MCMC methods; the difference lies in the manner in which the proposals are created. In order to construct a proposal for a chain, the remaining chains (or a subset thereof) are gathered into pairs (without replacement) and the difference vector between the states in each chain pair calculated. These difference vectors are combined (in different manners, giving rise to variants of the DE-MC and DrEAM algorithms) and scaled to calculate a proposal. The combination of states from various chains provides the correct orientation of the proposal distribution; the DrEAM algorithm determines the correct scaling. The proposal is accepted/rejected using an acceptance probability, derived in much the same manner as conventional MCMC methods.

The generation of proposals by combining the current states of multiple, concurrent chains is quite different from DRAM, which generates them from a multivariate Gaussian distribution. DRAM keeps a running history of the sample chain, and the proposal distribution's covariance matrix is periodically updated using it. Thus the history of the chain is used to orient the proposal distribution; its scaling follows arguments which are very similar to DrEAM. The use of a multivariate Gaussian proposal makes DRAM very efficient when constructing posteriors which are similar to Gaussians; however, in case of fat-tailed or multimodal distributions, mixing can be a problem. In principle, this shortcoming coming can be addressed by DrEAM, but this pre-supposes

an adequacy of concurrent chains from which an efficient posterior can be fashioned. Generally, in DrEAM [25], the practice has been to have as many chains as the number of parameters being estimated.

This brief discussion of DrEAM and DRAM lead to two hypotheses

1. For inverse problems where few (e.g., less than five) parameters are to be estimated i.e., low-dimensional inverse problems, DRAM may be more efficient than DrEAM. This is because it is unlikely that an efficient proposal distribution can be fashioned from a combination of 2 or 3 pairs of concurrent chains. This problem can be ameliorated by having many DrEAM chains, even for low-dimensional problems, but this comes at the cost of computational efficiency vis-à-vis a single-chained DRAM approach.

2. As the dimensionality of the inverse problem increases, DrEAM may become more competitive, especially for fat-tailed posterior distributions.

Verifying these hypotheses forms the bulk of the study.

Model selection: Selecting between competing models fitted to the same data is a straightforward task when parameters are estimated as point values; the L_2 norm of the discrepancy between data and model prediction forms a convenient metric. The problem is more involved when parameters are evaluated as distributions. In [9, 10], Gneiting *et al.* derive metrics, predicated on posterior predictive tests, that can be used to rank fitted models conditioned on their ability to reproduce observations. Basically, the model is evaluated using parameter samples from the posterior distributions to create an ensemble of model predictions corresponding to measurements (i.e., we conduct a posterior predictive test). One of the metrics that can be simply calculated is the mean absolute error (MAE) between the predictions and the data. For the cumulative rank probability score (CRPS), one calculates the difference between the CDFs (cumulative distribution function) of the model predictions and the measurement (denoted as a step function). The interval score (IS), obtained using the interquartile range (IQR), forms yet another metric to gauge the (predictive) skill of the posterior predictive test.

Note that all three metrics select between models based on the predictive skill of the fitted models. They do not distinguish between the complexity of the model and cannot detect over-fitting. The conventional statistical tactic of proposing nested models of increasing complexity and using information theoretic criteria e.g., Akaike Information Criterion is rarely of much use in scientific settings since few physically-based models are nested; rather, competing models reflect hypotheses regarding the underlying physical processes governing the observations. In such a setting, predictive ability is often a sound basis for model selection.

We structure the study as follows. We will investigate DrEAM and DRAM first within the context of a linear problem which has an analytical solution. This will allow us to gauge the difference between the two "numerical' solutions of the inverse problems (as obtained from DRAM and DrEAM) as well as their difference from the true solution. We will then proceed to a test with a nonlinear model of soil hydrology, to estimate the distribution of clay, as a function of depth, using simple (2 or 3-parameter) models of the clay profile. This will be followed by a test where a

higher-dimensional model (a 10-parameter Markov random field) is used for the clay profile. We expect that DRAM will outperform DrEAM in the low-dimensional problem, but a 10D problem may be large enough for DRAM and DrEAM to be comparable.

The chapter is structured as follows. In Sec. 3.2 we derive the Bayesian inverse problem and specify the error models. In Sec. 3.3 we describe the forward problems. In Sec. 3.4, and its subsections, we describe the modeling required to reduce the dimensionality of the inverse problem and the method used to generate the synthetic data on which the two methods were tested. We also demonstrate the use of MAE, CRPS and IS to select between models. In Sec. 3.5, we draw our conclusions.

3.2 Formulating the inverse problem

Let $M(\mathbf{\Theta})$ be a model, with the parameters $\mathbf{\Theta}$ which have to be estimated from a set of data $\mathbf{y}^{obs}$. Both $\mathbf{y}^{obs}$ and $\mathbf{\Theta}$ are vectors, and $\mathbf{y}^{obs}$ can be time-dependent. The model may not reproduce the data exactly (there are errors arising from measurements and the model's shortcomings) and we model the errors as i.i.d. Gaussians.

$$\mathbf{y}^{obs} = M(\mathbf{\Theta}) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma) \tag{3.2}$$

Under these conditions, the probability $\mathcal{L}(\mathbf{y}^{obs}|\mathbf{\Theta})$ of observing the data $\mathbf{y}^{obs}$, given a given value of model parameters, $\mathbf{\Theta}$) is given by

$$\mathcal{L}(\mathbf{y}^{obs}|\mathbf{\Theta}) \propto \prod_{i=1}^{N} \exp\left(-\frac{(y_i - M(\mathbf{\Theta}))^2}{\sigma^2}\right), \tag{3.3}$$

where $\{y_i\} = \mathbf{y}^{obs}$ are the elements in the data vector, which is of size N. σ is assumed known.

The prior beliefs regarding $\mathbf{\Theta}$ are modeled simply. We assume that the values of each component of $\mathbf{\Theta}$ are independent and are modeled as Gaussians with large standard deviations i.e., these are vague priors, which allow the data to determine the parameter values. Thus

$$\pi(\mathbf{\Theta}) = \prod_{j=1}^{m} \pi_j(\theta_j), \tag{3.4}$$

where m is the size of the $\mathbf{\Theta} = \{\theta_j\}$ vector (i.e., the number of parameters to be estimated), and π_j are Gaussians. The exact specification of the priors π_j and σ are problem dependent and will be mention for each of the tests in Sec. 3.4. Substituting Eq. 3.3 and Eq. 3.4 into Eq. 3.1 completes the inverse problem formulation.

3.3 Description of the forward problems

We consider two forward problems, a linear one to check the accuracy of DRAM versus DrEAM and a nonlinear one to compare their efficiency. These are described below.

3.3.1 Linear forward problem

We consider a unit domain $[0, 1]$ discretized with a uniform grid with 10 grid cells. A field $\mathbf{x}$ is described at the cell-centers. $\mathbf{x}$ varies smoothly in space and is a sample drawn from a multivariate (10-dimensional) Gaussian $\mathcal{N}(0, \mathbf{\Gamma})$, with a stationary covariance matrix $\mathbf{\Gamma}$, given by the correlation function $C(\Delta_{ij}) = \exp(-\Delta_{ij}^2/\lambda^2)$. The correlation length $\lambda = 0.3$ and Δ_{ij} is the distance between grid cells i and j. The model predictions $\mathbf{y}$ are given by $\mathbf{y} = \mathbf{K}_t\mathbf{x}$, where $\mathbf{K}_t$ is a matrix. The elements of $\mathbf{K}_t$ were chosen randomly.

3.3.2 Soil moisture dynamics

The soil moisture dynamics model, also referred to as the Zeng & Decker (ZD) model, is described in [27]. Its incorporation into the CLM4 is described in Chapter 7, Section 4 of [5]. CLM4 uses a highly stretched, 10-block grid to model subsurface hydrological dynamics. This (almost exponentially) stretched grid, which reaches 3.44 m below the land surface, and the position of a saturated zone (in case of a shallow water table) leads to numerical instabilities when Richard's equation is solved using conventional PDE discretization and time-integration techniques; the ZD model is a reformulation, with a particular time-implicit formulation that preserves stability. The equation is written as

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial z}\left[k\frac{\partial(\psi - \psi_E)}{\partial z}\right] - Q, \tag{3.5}$$

where ϕ (mm^3 of water per mm^3 of soil) is the soil moisture fraction, ψ is the soil matric potential [mm] and ψ_E is the equilibrium soil matric potential. Both ψ and ψ_E are complicated algebraic functions of ϕ. The hydraulic conductivity k is dependent on (via nonlinear algebraic relationships) on the soil moisture fraction, and the volumetric content (volume fractions) of clay, sand and organic matter in the soil. The relationships are documented in [5]. Q captures the external flux of moisture into the soil. This consists of precipitation, seepage of surface water and loss of moisture from the subsurface via evapotranspiration. In our problem, we will ignore the seepage of surface water. Precipitation will follow observed data. Loss of soil moisture via evapotranspiration is modeled according to the models in Chapter 8, Section 1 (stomatal resistance) in [5].

The model was specialized to our site (Diablo plateau, east of El Paso, Texas). The only vegetation considered in evapotranspiration was C4 grass (typical for Diablo plateau). The water table was assumed to be deeper than 3.44 meter (i.e., the soil was assumed to be partially saturated). The grid spacing, the true sand and clay profiles and precipitation (gathered weekly, over a 20 week period) were taken from [28] for the "dry-location" test case. The evapotranspiration profile (variation with depth) is taken from [6].

3.4 Tests

In this section, we first test the accuracy of the DrEAM and DRAM solution in a problem with an analytical solution (Sec. 3.4.1). We follow this up with a test using the nonlinear ZD model using two simple clay profile models in Sec. 3.4.2. In the same section, we demonstrate the use model selection scores (CRPS, MAE and IS) to select between the two clay profile models. We finally compare DrEAM versus DRAM with a 10-dimensional inverse problem in Sec. 3.4.3. The Matlab code for DRAM was obtained from http://www.helsinki.fi/∼mjlaine/mcmc/. The Matlab code for DrEAM was obtained from http://jasper.eng.uci.edu/software.html. The samples were checked for convergence (independence of the Markov chain) via the `mcgibbsit` package (http://cran.r-project.org/web/packages/mcgibbsit/) which is based on the theory in Chapter 7 of [8]. For multi-chain DrEAM, the Gelman-Rubin statistic (Chapter 8 in [8]) was also used to monitor convergence.

3.4.1 A linear inverse problem

We consider the 10-dimension linear problem $\mathbf{y} = \mathbf{K}_t\mathbf{x}$ described in Sec. 3.3.1. We create a synthetic data vector $\mathbf{y}^{obs}$, $y_j^{obs} = y_j + e_j, \quad e_j \sim \mathcal{N}(0, \epsilon^2), \quad j = 3, 5, \ldots 9$, where $\epsilon = 0.001$ so that

$$\mathbf{y}^{obs} = \mathbf{K}\mathbf{x} + \mathbf{e}, \tag{3.6}$$

where the sensitivity matrix $\mathbf{K}$ contains rows $3, 5, \ldots 9$ of the matrix $\mathbf{K}$. We model the inferred solution $\mathbf{x}^{'}$ as a multivariate Gaussian drawn from a Gaussian posterior distribution i.e. $\mathbf{x}^{'} \sim \mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{\Gamma}})$. We assume a prior $\pi(\mathbf{x}) \equiv \mathcal{N}(\mathbf{x}_a, \mathbf{\Gamma})$. The analytical expressions [17] for $\{\hat{\mathbf{x}}, \hat{\mathbf{\Gamma}}\}$ are

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{x}_a + \mathbf{\Gamma}\mathbf{K}^T \left(\mathbf{K}\mathbf{\Gamma}\mathbf{K}^T + S_\epsilon\right)^{-1} (\mathbf{y}^{obs} - \mathbf{K}\mathbf{x}_a) \\ \hat{\mathbf{\Gamma}} &= \mathbf{\Gamma} - \mathbf{\Gamma}\mathbf{K}^T(\mathbf{K}\mathbf{\Gamma}\mathbf{K}^T + S_\epsilon)^{-1}\mathbf{K}\mathbf{\Gamma}. \end{aligned} \tag{3.7}$$

We set $\mathbf{x}_a = 0$ and $S_\epsilon = \epsilon^2\mathbf{I}$ and calculate the particulars of the posterior distribution $\mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{\Gamma}})$ exactly.

The same problem is solved using DRAM and DrEAM. In both cases, 5,000,000 samples were drawn by the two methods. DrEAM was run with 20 chains. We model $\mathbf{x}^{'} = L\mathbf{z}$, where L is the Cholesky decomposition of $\mathbf{\Gamma}$ and $\mathbf{z}$ is a 10-dimension vector whose elements are i.i.d standard normals. Samples (of $\mathbf{z}$) from the posterior distribution are converted into samples of $\mathbf{x}^{'}$. In Fig. 3.1 we plot the true $\mathbf{x}$, the analytical solution $\hat{\mathbf{x}}$ and the median, 25th and 75th percentiles of the estimate of $\hat{\mathbf{x}}$ calculated using DRAM and DrEAM. We see that the numerical results agree very closely with each other, as well as with the analytical results. The L_2 norm of the difference between the numerical $\hat{\mathbf{x}}$ and the analytical one are 0.275 (DRAM) and 0.2588 (DrEAM). This conveys the impression that distributions e.g., the covariance matrices may also agree with the analytical result. In Fig. 3.2 (top row), we plot the DRAM and DrEAM covariance matrices. We see that they are symmetric but do not agree with each other - the Frobenius norm of the difference between the DrEAM and DRAM covariance matrices is 0.195. In the bottom row, left,

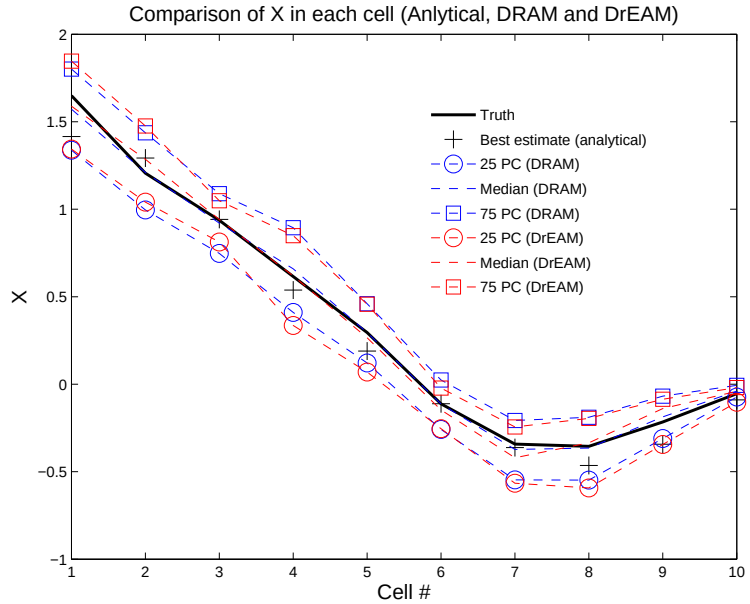


Figure 3.1. Comparison of the true $\mathbf{x}$ (thick line), $\hat{\mathbf{x}}$ (crosses) and the inferences calculated using DRAM (in blue) and DrEAM (in red). We see that the median of the $\mathbf{x}'$ samples are very close to the analytical result, regardless of whether DRAM or DrEAM was used. The 25th and 75th percentiles also show very little differences. The L_2 norm of the difference between the numerical $\hat{\mathbf{x}}$ and the analytical one are 0.275 (DRAM) and 0.2588 (DrEAM).

we plot the analytical results. Comparing with the top row, we see that neither the DRAM nor the DrEAM solutions are close to the analytical result. The Frobenius norm of the difference between the empirical covariance matrices and the analytical one are 0.68 (DRAM) and 0.705 (DrEAM). Thus the two empirical covariance matrices are closer to each other than they are to the analytical solution. In the bottom right subfigure, we plot the diagonal elements of the analytical, DRAM and DrEAM covariance matrices. While the DrEAM and DRAM solutions agree (to a degree), they are quite different from the analytical solution, reinforcing the conclusions obtained by comparing the covariance matrices.

In Figs. 3.3 and 3.4, we plot the marginals for x_3, x_5, x_7, x_9 as obtained from DrEAM and DRAM. We see that the marginal distributions for DrEAM are sparse and noisy; while the samples may provide plausible estimates for integrated measures like various quantiles (as seen in Fig. 3.1, the distributions clearly leave a lot to be desired. In contrast, the DRAM results in Fig. 3.4 show smooth behaviors as may be expected from a multiGaussian distribution. Further, clearly, the entire parameter space seems to have been well sampled.

To conclude, both DrEAM and DRAM draw samples which can used to calculate "integrated" measures like medians, the higher quantiles etc reliably i.e., they provide results that agree with each other, and to a large degree, with the true solution. However, higher statistical moments

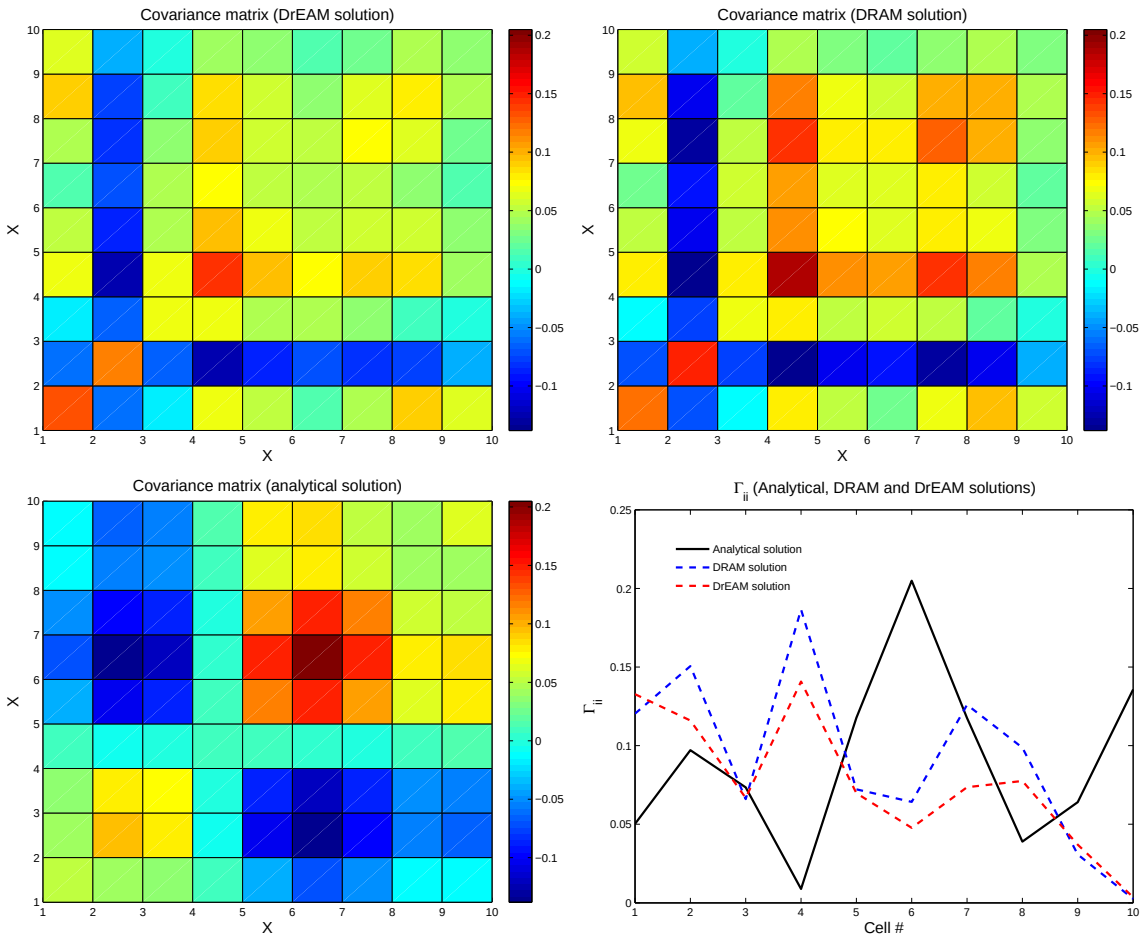


Figure 3.2. Top row: Empirical covariance matrices generated by DrEAM (left) and DRAM (right). We see differences between the two (the Frobenius norm of differences is 0.195). Bottom row: We plot the analytical covariance matrix $\hat{\mathbf{\Gamma}}$ on the left. We see significant differences with the covariance matrix generated by DrEAM (Frobenius norm of difference is 0.705) and DRAM (Frobenius norm of 0.68). In the bottom right subfigure, we plot the diagonal entries of the three covariance matrices (analytical, DrEAM and DRAM). The differences between the analytical results and the numerical one are large.

or the more extreme quantiles may be suspect since the two methods obtain covariance matrices which are quite different from the analytical solution. The distributions obtained by DrEAM are not very realistic, whereas those constructed from DRAM samples "look" real. However, given the discrepancy in the covariance matrix vis-à-vis analytical results, the distribution is approximate.

3.4.2 Comparison using a low-dimensional inverse problem

In this test, we use DrEAM and DRAM to solve a nonlinear soil moisture problem. We simulated the time-dependent soil moisture volume fraction over a 20 week period, as described in Sec. 3.3.2. The moisture values, at the end of every week, in the center of grid-blocks $2, 4, \ldots 8$ were retained as "observations", after adding a measurement noise $\sim \mathcal{N}(0, 0.005^2)$.

The aim of the test was to estimate the clay profile. The true clay profile and the soil-moisture distribution (with depth) are shown in Fig. 3.5. We see that the clay profile shows shows a decreasing trend, till about 1.75 m depth, at which point it becomes a constant. This may be because the last 1.75 meters are covered by a single grid-block in CLM4. The evolution of the soil moisture profile over 20 weeks shows a progressive drying out, whereas the lower depths barely change. The richer dynamics in the upper reaches indicate that the clay profile may be inferred accurately there, whereas the lack of information/dynamics in the lower grid-blocks (whose centers are shown in the soil moisture profile as symbols) indicate that the inference may incur large errors there.

We propose a "truncated linear" clay profile model

$$f(x) = \begin{cases} a - bx & \text{if } x \geq c \\ a - bc & \text{if } x < c \end{cases}, \tag{3.8}$$

with the aim of estimating $\mathbf{\Theta} = \{a, b, c\}$. $f(x)$ is the volume fraction of clay, as a function of depth x. Both b and c are constrained to be positive, and so we infer their log-transformed counterparts $\ln(b)$ and $\ln(c)$. The three objects of inference (OOI) are assumed independent with normal priors, whose specifics are listed in Table 3.1.

The problem is solved using both DrEAM and DRAM, using 40,000 (DRAM) and 60,000 (DrEAM) samples. Three chains were used for DrEAM. The convergence was monitored using `mcgibbsit` for the median of the distribution. In Fig. 3.6, top row, we plot the estimated clay profiles for DrEAM (left) and DRAM (right). We see that the profiles generated using DrEAM are narrower. This is reflected also in the posterior predictive test for the soil moisture at the end of Week 18 (Fig. 3.6, bottom row). It is clear from the posterior predictive check that the DrEAM results do not predict the observations well; the median of the predictions are often quite far away from the observations. Such a poor estimation, vis-à-vis DRAM is not unexpected; DrEAM with 3 chains is not very different from a blocked MCMC scheme without proposal adaptation. We next use the scores discussed in Sec. 3.1 to compare DrEAM versus DRAM with respect to their predictive skills. These are tabulated in Table 3.2; it is clear that while the DrEAM chains become independent quicker (i.e., with about 30% fewer samples), the parameter estimation leaves a lot to be desired.

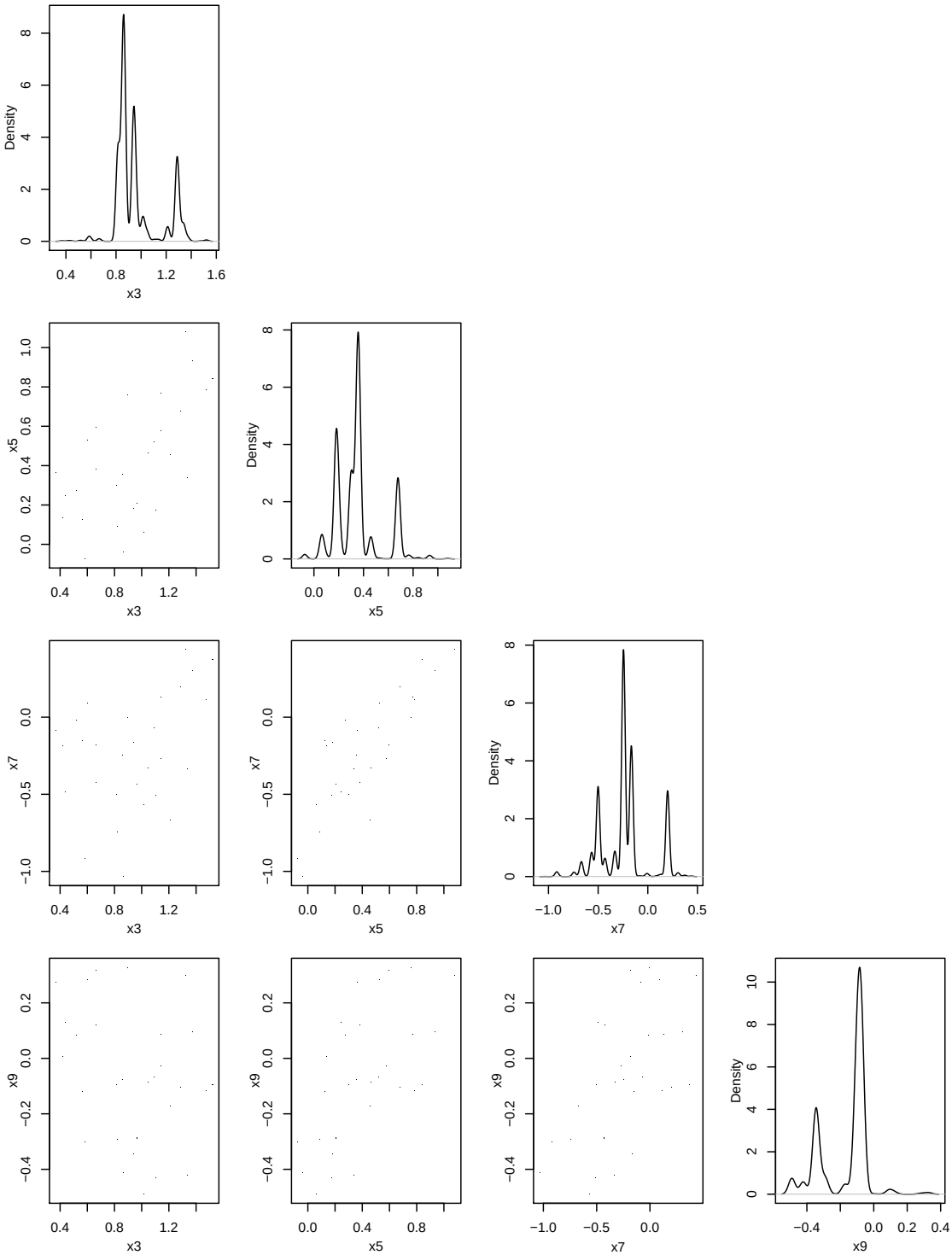


Figure 3.3. Marginals and joint distributions for x_3, x_5, x_7, x_9, obtained via DrEAM. We also plot the PDFs for each variable, which show a craggy behavior.

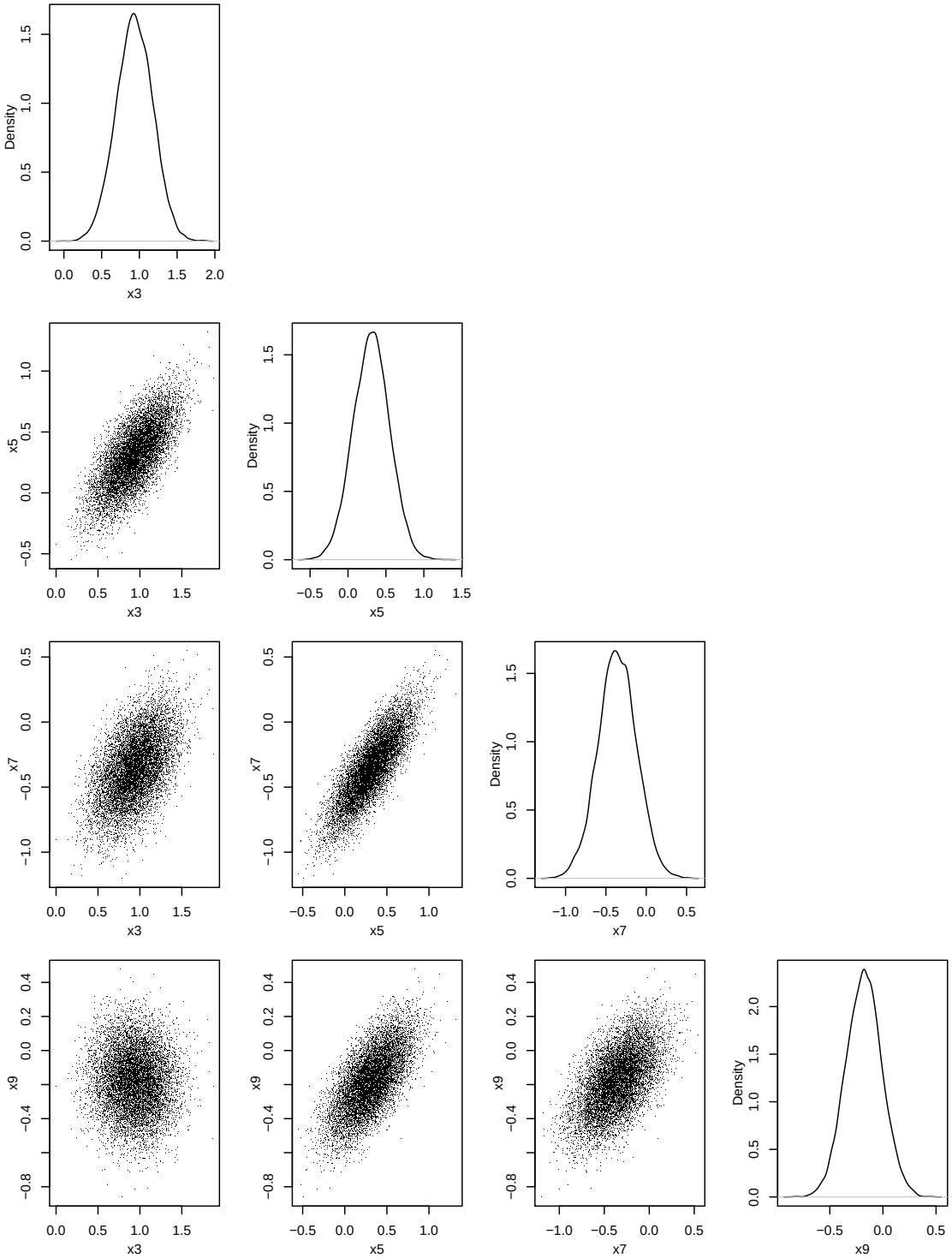


Figure 3.4. Marginals and joint distributions for x_3, x_5, x_7, x_9, obtained via DRAM. We also plot the PDFs for each variable, which show a smooth, Gaussian behavior, as might be expected of marginals of a multivariate Gaussian.

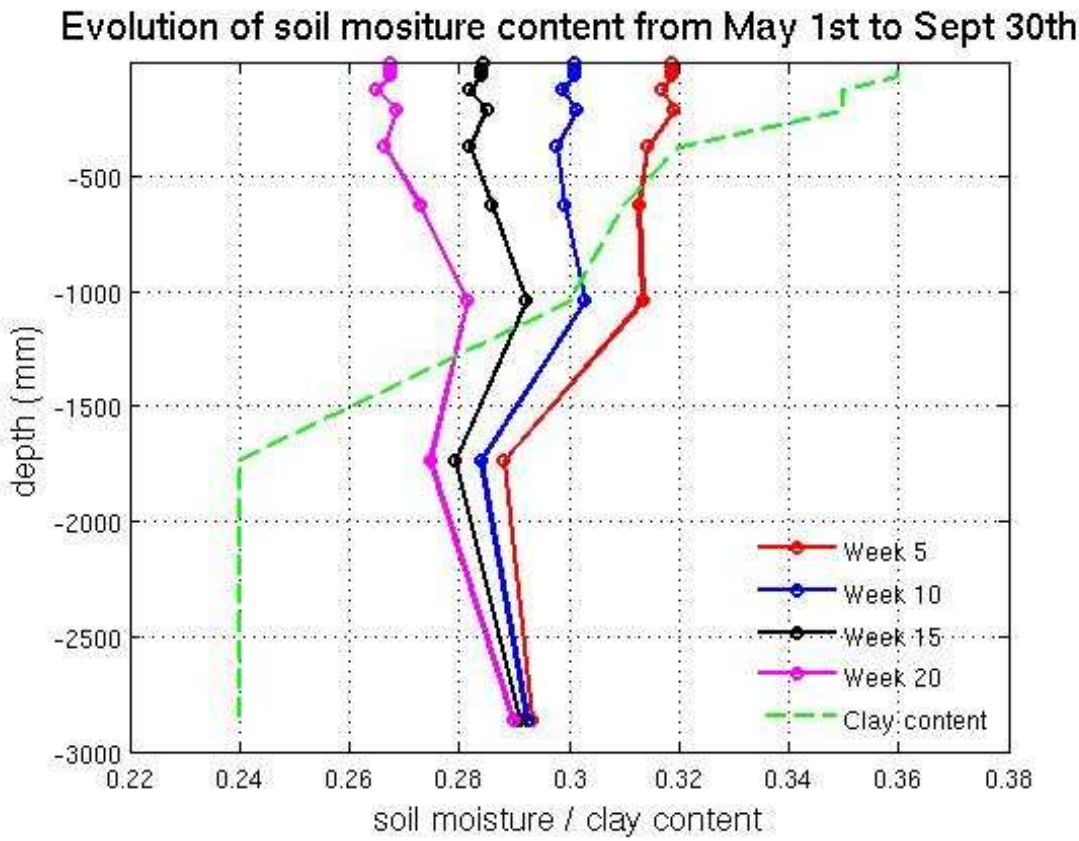


Figure 3.5. Plot of the true clay profile (in green) and the evolution of the soil moisture profile over 20 weeks. We see a progressive drying-out of the upper reaches of the soil, whereas the lower depths hardly record any change. The symbols in the soil moisture profiles indicate grid-block centers.

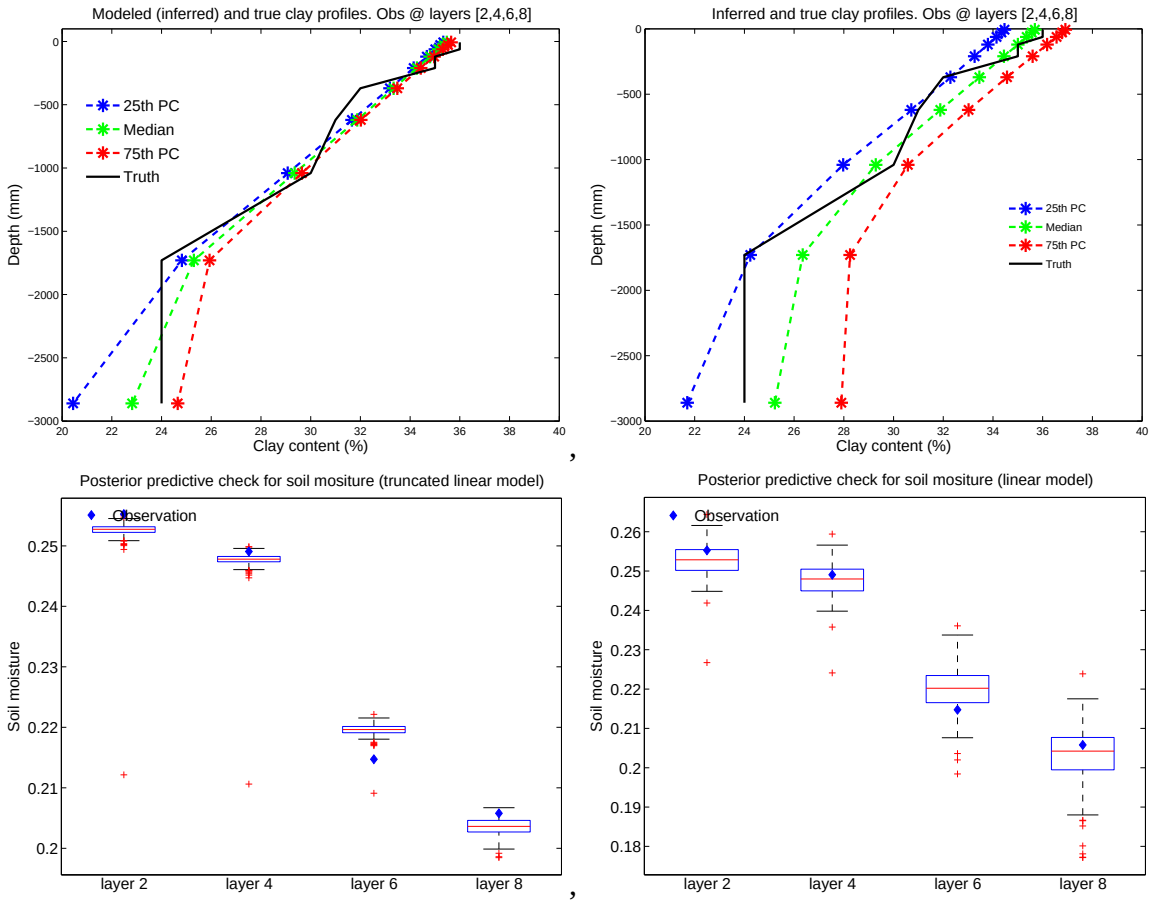


Figure 3.6. Top: Estimated clay profiles (medians and quartiles) obtained using DrEAM (left) and DRAM (right). The quartiles and medians are calculated from the posterior predictive test for the clay content in each grid-block independently. Bottom: The results from the posterior predictive tests for the soil moisture at the end of Week 18, as obtained from DrEAM (left) and DRAM (right).

Table 3.1. Specifics of the normal priors used for the log-transformed variables for the "truncated linear" and "exponential" clay profile parameters in Sec. 3.4.2. The third column contains the mean and standard deviations of the normal distributions and the last column the extreme values where the priors are truncated. Length is measured in meters.

Variable	Clay profile model	(μ, σ)	Max/min
a	Truncated linear	(30, 20)	50/15
$\ln(b)$	Truncated linear	$(\ln(7 \times 10^{-3}), 3.0)$	$\ln(4 \times 10^{-3}) / \ln(1.0 \times 10^{-2})$
$\ln(c)$	Truncated linear	$(\ln(1.5), 0.5)$	$(\ln(3.44) / \ln(1.0))$
a	Exponential	(30, 20)	50/15
b	Exponential	(4, 3)	8/0.5

Table 3.2. Comparison of the predictive skill of DrEAM versus DRAM. The last column indicates the number of iterations to convergence, per chain, as measured by `mcgibbsit`. We see that the predictive skill of the DRAM-fitted model is uniformly better.

Method	CRPS	MAE	IS	Iterations per chain
DrEAM	2.212×10^{-3}	2.716×10^{-3}	0.009959	12770
DRAM	1.869×10^{-3}	2.6299×10^{-3}	0.010075	18,286

CRPS, MAE and IS which, above, were used to gauge the predictive skills of models fitted with DrEAM and DRAM, can also be used to discriminate between competing models. We demonstrate this by proposing an exponential clay profile i.e. $g(x) = a \exp(-bx)$ and estimating $\boldsymbol{\Theta} = \{a, b\}$. The test described above is repeated with $g(x)$ using DRAM. The priors for $\{a, b\}$ are in Table 3.1. Fig. 3.7 shows the inferred clay profile and the results of the posterior predictive test for soil moisture at the end of Week 18. We see that the fitted clay profile deviates significantly from the true one; further, comparing with Fig. 3.6, the uncertainty in the inferred profile is larger than that obtained using the "truncated linear" clay profile. The CRPS, MAE and IS scores of the posterior predictive tests are, respectively, 1.91×10^{-3}, 2.98×10^{-3} and 0.006965. Comparing with the scores for the "truncated linear" profile in Table 3.2, we see clearly that the predictive skill of the "exponential" profile is inferior, leading to its rejection.

3.4.3 Comparison using a high-dimensional inverse problem

Finally, we address the problem of high-dimensional inference. Noticing that the true clay profile in Fig. 3.5 is rather irregular, we propose a Markov random field (MRF) [14] model for clay profile. More precisely, we propose $h(x) = f(x; a, b, c) + \boldsymbol{\delta}$, where $f(x)$ is the "truncated linear"

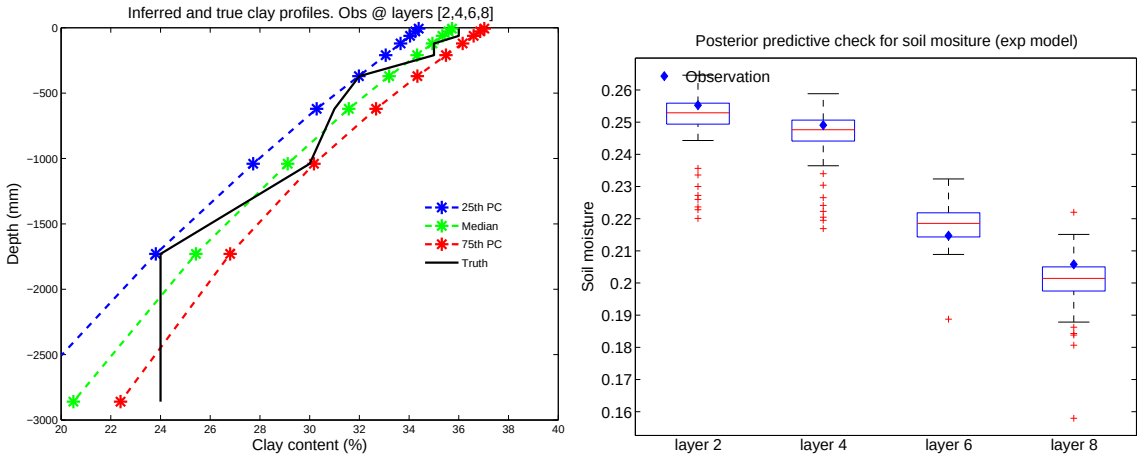


Figure 3.7. Left: Inferred clay profiles using the "exponential" clay profile model. The quartiles and medians are calculated from the posterior predictive test for the clay content in each grid-block independently. We see that the profile is significantly worse than the profile inferred in Fig. 3.6. Right: we plot the results of the posterior predictive test for soil moisture at the end of Week 18. In comparison to the results from the "truncated linear" profile in Fig. 3.6, the predictive skill of the exponential model is considerably less.

clay profile in Eq. 3.8 and $\boldsymbol{\delta}$ are deviations from it at each grid-block center. We model the discrete form of $\boldsymbol{\delta}$ with a 10-dimensional MRF. An MRF imposes a small degree of smoothness among the elements δ_i by imposing a likelihood function

$$P(\boldsymbol{\delta}|\alpha) \propto \alpha^{m/2}|\mathbf{H}|^{1/2} \exp\left(-\frac{1}{2}\boldsymbol{\delta}^T\mathbf{H}\boldsymbol{\delta}\right), \tag{3.9}$$

where $m = 10$ in our case, $\mathbf{H}$ is the precision matrix (symmetric, and positive semi-definite) and α is the precision parameter. For any site i on the 1D grid, the full conditional of any δ_i is determined by all others by the expression

$$\delta_i|\delta_{-i} \sim \mathcal{N}\left(-\frac{\sum_{j\neq i} h_{ij}\delta_j}{h_{ii}}, \frac{1}{\alpha h_{ii}}\right).$$

On a uniform 1D grid, the elements of $\mathbf{H}$ are given by

$$h_{ij} = \begin{cases} -1 & \text{if } i \text{ and } j \text{ are indices of adjacent grid-blocks} \\ 0 & \text{if } i \text{ and } j \text{ are indices of non-adjacent grid-blocks} \\ n_i & \text{if } i = j \text{ and } n_i \text{ is the number of grid-blocks neighboring } i \end{cases}, \tag{3.10}$$

In our particular case, we take the "mean" clay profile as $f(x; \hat{a}, \hat{b}, \hat{c})$, where the "hatted" values indicate the MAP (maximum *a posteriori*) estimates obtained from the DRAM test in Sec. 3.4.2. α

Table 3.3. Comparison of the predictive skill of DrEAM versus DRAM, using the MRF model. The last column indicates the number of iterations to convergence, per chain, as measured by `mcgibbsit`. We see that the predictive skill of the DrEAM-fitted model is uniformly better, and at a far lower cost.

Method	CRPS	MAE	IS	Iterations per chain
DrEAM	3.98×10^{-4}	2.8×10^{-4}	0.001997	2,098
DRAM	1.41×10^{-3}	9.2×10^{-4}	0.0074	39,682

is set to 2. The parameters, $\boldsymbol{\Theta}$, to be estimated are the 10 elements of $\boldsymbol{\delta}$. In order to accommodate a stretched grid, we use a modified $\widetilde{\mathbf{H}}$ which is constructed as follows.

We commence with an upper triangular forward difference matrix, $\mathbf{D}$ where the elements d_{ij} are given by

$$d_{ij} = \begin{cases} -1/l_i & \text{if } i = j \\ 1/l_i & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

where l_i is the distance between the centers of grid-blocks i and $i+1$. $\mathbf{D}$ calculates the first-order forward finite-difference slopes of the field it is applied to. $\widetilde{\mathbf{H}} = D^T D$ and $\boldsymbol{\delta}^T \mathbf{H} \boldsymbol{\delta}$ is the sum of square of the slopes calculated at grid-block centers using a forward-difference operator. An augmented likelihood function $\mathcal{L}_a$ is used in Eq. 3.3, formed by

$$\mathcal{L}_a(\mathbf{y}^{obs}|\boldsymbol{\Theta}) \propto \mathcal{L}(\mathbf{y}^{obs}|\boldsymbol{\Theta}) P(\boldsymbol{\delta}|\alpha)$$

where $P(\boldsymbol{\delta}|\alpha)$ is obtained from Eq. 3.9 and $\mathcal{L}$ from Eq. 3.3. The prior $\pi(\boldsymbol{\delta})$ is modeled as i.i.d. Gaussian ($\mathcal{N}(0,3)$) for all 10 elements of $\boldsymbol{\delta}$.

In Fig. 3.8 we plot the inferred clay profiles, using the MRF model, as calculated using DrEAM and DRAM. The DrEAM runs were computed with 10 chains. We see that the clay profiles computed using DrEAM are tightly clustered around the true profile, vis-à-vis DRAM. This is reflected in the posterior predictive test for the soil moisture at the end of Week 18 (bottom row), where the medians predicted by both DRAM and DrEAM agree with observations, with the DRAM-fitted model predicting a wider scatter. In Fig. 3.9 and Fig. 3.10 we plot the marginals for $\delta_i, i = 3, 5, \ldots 9$. We see that DRAM explores the parameter space densely, leading to smoother marginal posterior distributions; the DrEAM equivalents are quite rough. Yet `mcgibbsit` and the Gelman-Rubin statistic indicate that both the chains have converged.

Finally we use CRPS, MAE and IS to compute the accuracy of the posterior predictive tests using DrEAM and DRAM, and compare the accuracy obtained against the computational cost. These are summarized in Table 3.3. It is clear that the DrEAM results are about 3 times more accurate that DRAM and were obtained with 20 times fewer samples (per chain). This is quite a surprise given the rather unprepossessing marginals constructed using DrEAM samples in Fig. 3.9.

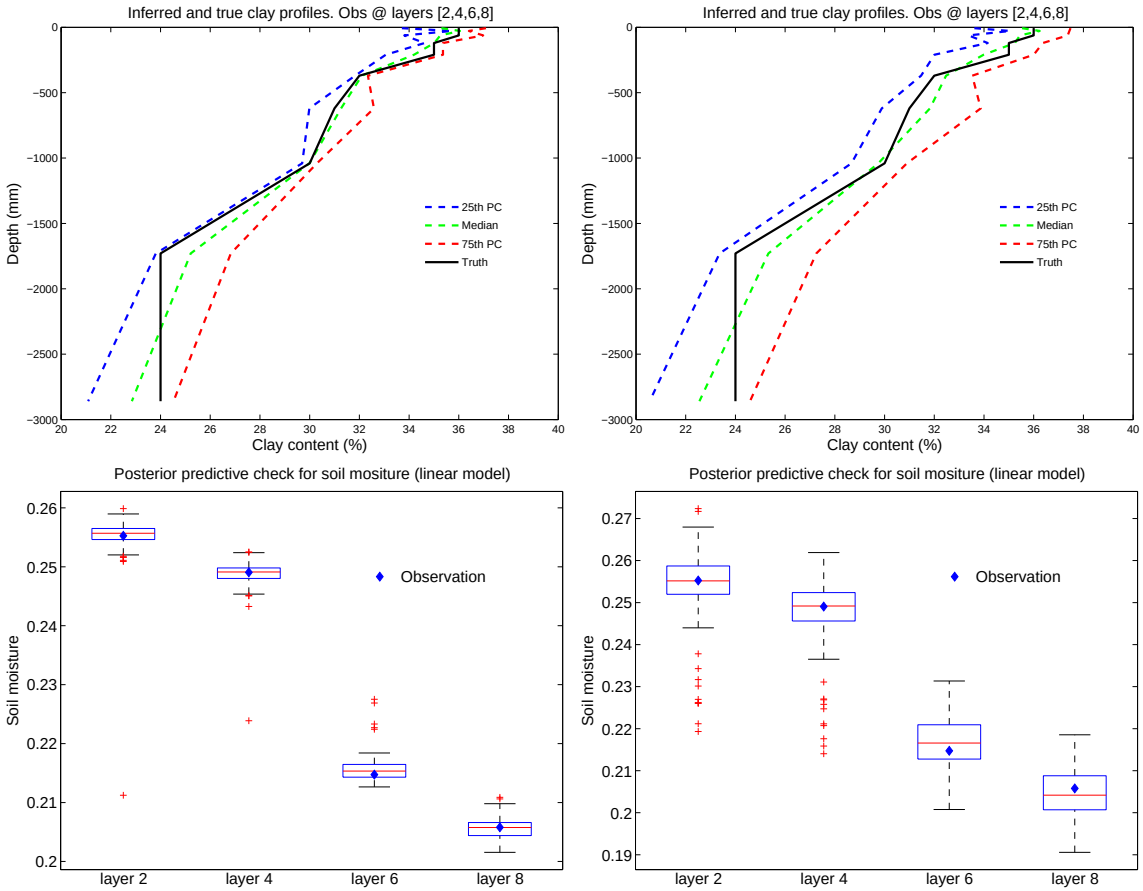


Figure 3.8. Top row: Estimated clay profiles (medians and quartiles) using the MRF model, as computed using DrEAM (left) and DRAM (right). The quartiles and medians are calculated from the posterior predictive test for the clay content in each grid-block independently. The true profile is also plotted. The DRAM profiles are more spread out. Bottom row: The posterior predictive tests for soil moisture at the end of Week 18, as computed using DrEAM (left) and DRAM (right). The wider spread of the clay profiles as computed by DRAM translates into a wider scatter of predicted soil moisture, as seen in the bottom right figure. The median soil moisture agrees with the observations, for both DrEAM and DRAM.

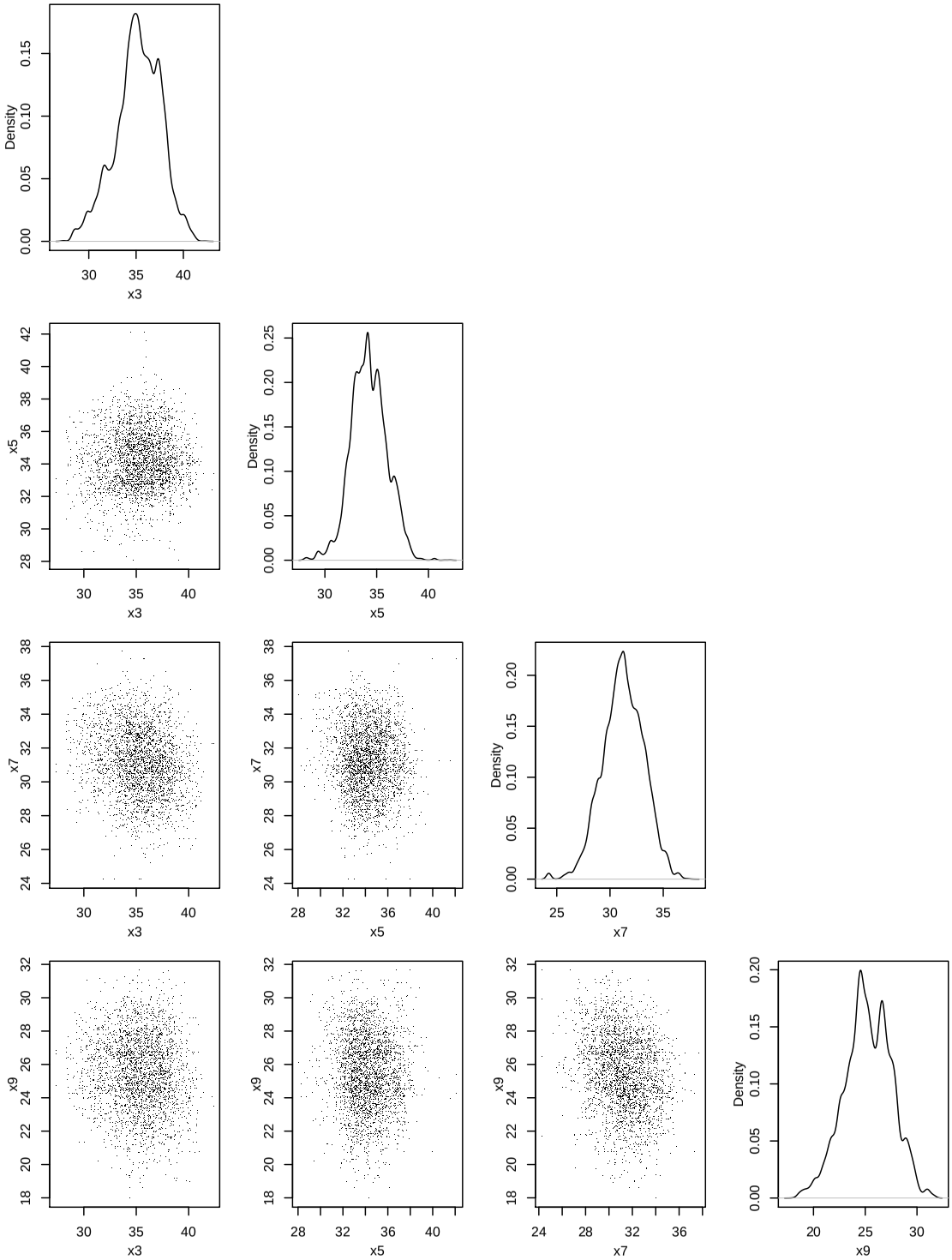


Figure 3.9. Marginals for $\delta_i, i = 3, 5, \ldots 9$, as computed from the DrEAM solutions. We see that the distributions are craggy, and the scatter plots are sparse. However, the 10 chains provide a far fuller sampling of the space, compared to Fig. 3.3.

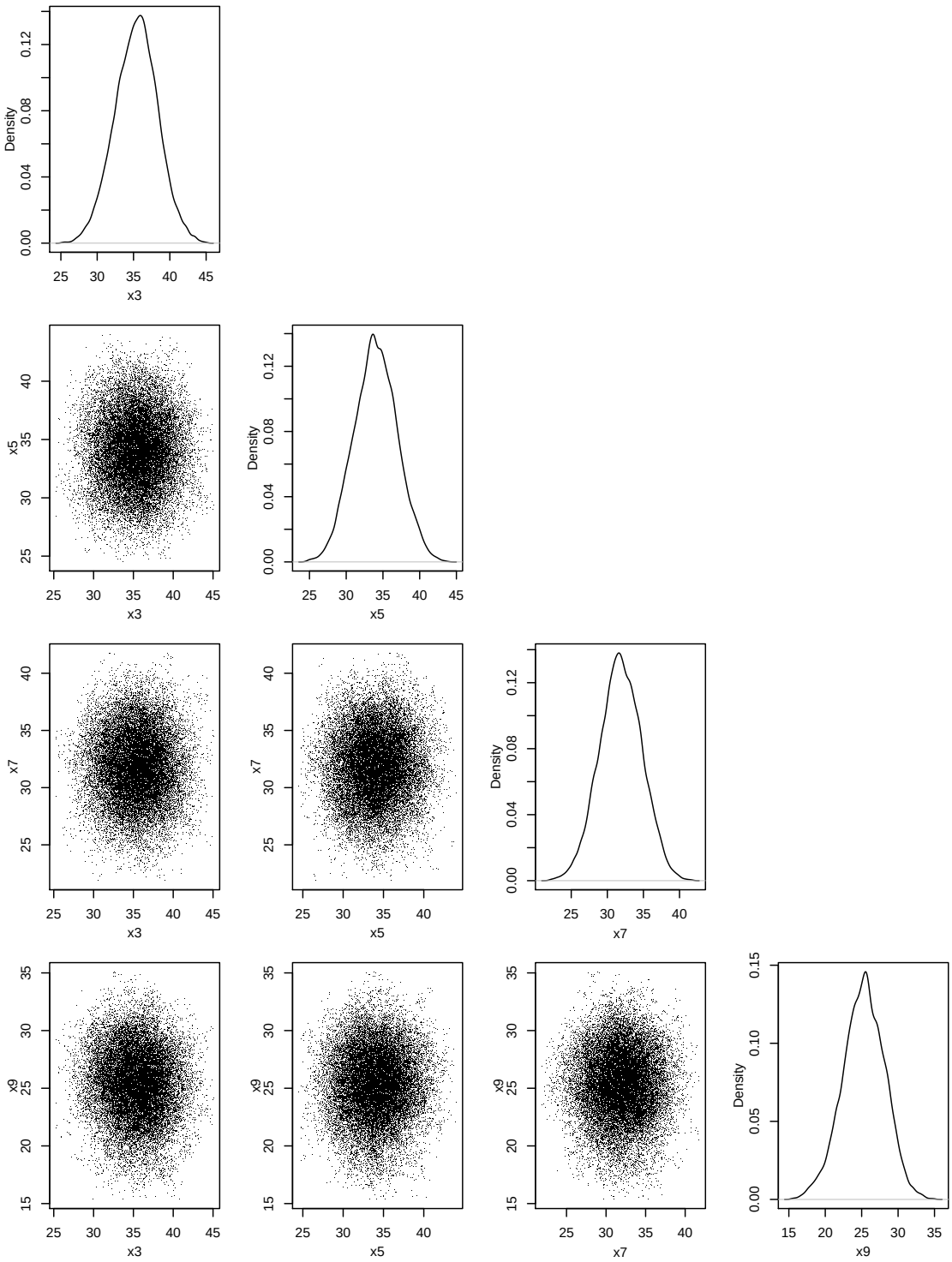


Figure 3.10. Marginals for $\delta_i, i = 3, 5, \ldots 9$, as computed from the DRAM solutions. We see that the distributions are smooth, and the scatter plots show dense exploration of the parameter space. This is in contrast with the sparse exploration in Fig. 3.9 by DrEAM.

3.5 Conclusions

We have conducted a study of how well (and efficiently) PDE models found in CLM4 could be calibrated to data. Our focus was on developing model parameter estimates as distributions, so that the uncertainties in the parameter estimates and model predictions could be rigorously evaluated. Such distributions can be constructed by posing a Bayesian inverse problem and solving it using an MCMC method. Due to computational expense of PDE models, our interest was on evaluating multichain MCMC methods like DrEAM, which can be easily parallelized. We have chosen 3 separate problems, solved them using DrEAM and compared the results with DRAM, a well-established *single-chain* MCMC method.

Judging from the three problems, it is clear that the distributions developed by DrEAM are inferior to DRAM, especially when the problem has few chains (around 5). When more chains are used, the sampling by DrEAM improves, but does not quite equal DRAM. However, a better sampling of the parameter space does not seem to result in fitted model with more predictive skill - in a three-parameter problem, DrEAM was marginally less accurate (and about 30% cheaper, per chain, computationally). In case of the 10-dimensional MRF model, DrEAM beat DRAM both in accuracy and computational efficiency, by wide margins.

Our tests with the linear problem, where an analytical expression was available, show that both DrEAM and DRAM are equally accurate when medians or quartiles are desired. However, the covariance matrix constructed by the sampling schemes are quite different from the analytical solution, though they are close to each other. Thus, it appears that the higher moments of distribution generated by both the methods may be approximate. However, given the accuracy of the posterior predictive tests in the MRF test, the subtle discrepancies may not matter greatly for most predictive purposes. This does not hold true if the aim is to predict rare/extreme occurences e.g., for risk analysis purposes.

Comparing the craggy PDFs generated by DrEAM with the smoother ones developed by DRAM for the Markov random field problem (Figs. 3.9 versus 3.10) and the lnear problem (Figs. 3.3 versus 3.4), lead us to believe that the distributions developed by DrEAM may be suspect. Thus investigating a method for reducing the computational time of the serial DRAM algorithm may be a worthwhile task. This can be partially accomplished by increasing the efficiency which with DRAM explores a high dimensional parameter space, perhaps, as performed in [3], by using multiple Metropolis-couled MCMC chains. Alternatively, one may also investigate the use of Ensemble Kalman Filters (EnKF) to investigate the same inverse problem. EnKF are scalable and while they make Gaussian assumptions about the posterior distribution, the currently uncertainty about what the posterior distribution, as developed using different methods, suggest that the approximation may be defensible.

Ultimately, the choice of a calibration method depends upon the model in question and final goal of calibration. If a single parameter value is desired, deterministic, optimization methods, e.g., those in PEST [2], are far more efficient that the Bayesian methods described above. However, in keeping with CSSEF's focus on uncertainty quantification, point-estimates of parameters are unlikely to contribute much (except, perhaps, as a starting guess for MCMC chains). In keeping with the

contents of the Land UQ section (Sec.VI.3.2.2 in [1]), DRAM (or its Metropolis-coupled version), in conjunction with surrogate models, may be most suitable. Gridded parameter estimation, also with surrogate model, will require a multi-chain method, e.g., a parallelized implementation of DrEAM, since some of the parameters may have to be modeled as random fields (thus increasing the dimensionality of the inverse problem). Finally, parameter estimation using the full CLM4 model will require a parallel implementation of an EnKF (modified for parameter estimation), which the CSSEF team does not currently have. As a first step, its applicability to the ZD problem, and the comparison of its posterior distributions to DRAM and DrEAM should be explored.

Part II

Land UQ

Chapter 4

Polynomial Surrogate Construction for Community Land Model

The Community Land Model (CLM) is the terrestrial component of the Community Earth System Model (CESM), which is used extensively for projections of the future climate system. With over 100 uncertain input parameters and strong nonlinearities, the CLM presents a number of challenges for uncertainty quantification (UQ) methods. Besides, as a single run of the CLM requires significant computational effort, constructing a polynomial surrogate model as a response surface is a crucial component for performing both forward and inverse UQ.

4.1 Problem Formulation and Challenges

Consider an implementation of the Community Land Model (CLM) with $\tilde{d}$ input parameters, $\lambda_1, \ldots, \lambda_{\tilde{d}}$, and a single scalar output quantity of interest (QoI) $y = f(\boldsymbol{\lambda})$. Note that, due to input dependencies, the number of physical input parameters $\tilde{d}$ may be different from the true number of degrees of freedom d, hence the 'tilde' notation. The *forward function* $f(\cdot)$ is a deterministic function that acts as a black box and replicates a single-site CLM simulation. Typically, our methodology relies on CLM simulations at appropriately chosen input parameter regimes: these runs are called training samples or training runs.

Our ultimate goal is two-fold:

- (forward) uncertainty quantification and global sensitivity analysis,
- (inverse) parameter inference and calibration.

Below we list major challenges that both forward and inverse uncertainty quantification (UQ) methods face specific to the CLM:

- Parameter constraints: some parameters need to satisfy constraints imposed by their own definition or physics.

- Curse of dimensionality: the number of input parameters is large (about eighty, in the default study), making both parameterization of input-output relationship and the input space coverage challenging.
- Computational cost of the forward function: the CLM, even in the single-site mode, is expensive to run (a 1000-year simulation takes about 10 hours on a single processor), leading to *sparsity* of the training data set.

We will focus on the forward UQ and sensitivity analyses, leaving the related inverse problem as the next logical step, outside the scope of this report.

4.2 Community Land Model Input Parameters

Tables 4.1 and 4.2 present the list of CLM input parameters varied in our study. Besides range restrictions the input parameters need to satisfy the following constraints, by definition, or in order to remain consistent with associated physics:

$$\begin{aligned} \lambda_{18} &< \lambda_{22}, \\ \lambda_{30} + \lambda_{31} + \lambda_{32} &= 1, \\ \lambda_{33} + \lambda_{34} + \lambda_{35} &= 1. \end{aligned} \tag{4.1}$$

Besides the curse of dimensionality, the CLM input parameter set presents an additional challenge as some input parameters are related by addition constraints. For example, there are parameter triples $(\lambda_i, \lambda_j, \lambda_k)$ that lie on a plane $\lambda_i + \lambda_j + \lambda_k = 1$, in addition to their respective range constraints such as $\lambda_i \in [a_i, b_i]$. Another constraint that arises in the CLM input is a pair of parameters that need to have specific order $\lambda_i < \lambda_j$ due to certain physics restrictions, again, in addition to the range constraints $\lambda_i \in [a_i, b_i]$.

For example, Figure 4.1 illustrates a uniform sample set on a polygons that are obtained due to constraints $\lambda_{33} + \lambda_{34} + \lambda_{35} = 1$ and $\lambda_{18} < \lambda_{22}$, respectively.

4.3 Rosenblatt Transformation

In this section, we introduce a transformation that maps input parameter vector $\boldsymbol{\lambda}$ with dependent or constrained components to a vector of i.i.d. uniform variables $\boldsymbol{\eta}$. This transformation is called Rosenblatt transformation [18] and is essentially a generalization of the CDF transformation (1.4) to multiple dimensions.

To clarify the upcoming notation, let us remove one input parameter from the triple $(\lambda_i, \lambda_j, \lambda_k)$ for each constraint of a form $\lambda_i + \lambda_j + \lambda_k = 1$, since one of the parameters in the triple is completely

Table 4.1. CLM input parameters: part one

Notation	Name	Default	Min	Max	Units	Description
λ_1	displar	0.67	0.1	1	m	displacement length: canopy top
λ_2	dleaf	0.04	0.01	0.1	m	characteristic leaf dimension
λ_3	mp	6	3	16	none	slope of conductance to photosynthesis
λ_4	qe25	0.06	0.04	0.08	umol	C/umol phot Quantum efficiency
λ_5	rholvis	0.07	0.01	1	none	leaf reflectance (vis)
λ_6	rholnir	0.35	0.01	1	none	leaf reflectance (nir)
λ_7	rhosvis	0.16	0.01	1	none	stem relectance (vis)
λ_8	rhosnir	0.39	0.01	1	none	stem reflectance(nir)
λ_9	taulvis	0.05	0.01	1	none	leaf transmittance (vis)
λ_{10}	taulnir	0.1	0.01	1	none	leaf transmittance (nir)
λ_{11}	tausvis	0.001	0.0001	0.01	none	stem transmittance (vis)
λ_{12}	tausnir	0.001	0.0001	0.01	none	stem transmittance (nir)
λ_{13}	xl	0.01	0.01	1	none	leaf/stem orientation index
λ_{14}	roota_par	7	1	20	m-1	rooting distribution parameter
λ_{15}	rootb_par	2	0.5	10	m-1	rooting distribution parameter
λ_{16}	slatop	0.01	0.08	0.12	m2/gC	SLA at top of canopy
λ_{17}	dsladlai	0.0012	0.001	0.007	m2/gC/LAI	SLA/dLAI
λ_{18}	leafcn	35	23	70	gC/gN	leaf C to N ratio
λ_{19}	flnr	0.05	0.04	0.1	none	frac of leaf N in Rubisco
λ_{20}	smpso	-66000	-120000	-20000	mm	soil water pot. at full opening
λ_{21}	smpsc	-255000	-300000	-120000	mm	soil water pot. at closure
λ_{22}	lflitcn	70	39	143	gC/gN	leaf litter C:N
λ_{23}	frootcn	42	25	85	gC/gN	fine root C:N
λ_{24}	livewdcn	50	25	75	gC/gN	live wood C:N
λ_{25}	deadwdcn	500	200	1400	gC/gN	dead wood C:N
λ_{26}	froot_leaf	1	0.3	5	gC/gC	new fine root alloc C /leaf C
λ_{27}	stem_leaf	1.5	0.6	5.3	gC/gC	new stem alloc C per leaf C
λ_{28}	croot_stem	0.3	0.1	0.7	gC/gC	new croot alloc C per stem C
λ_{29}	flivewd	0.1	0.06	0.28	none	fraction of new wood that is live
λ_{30}	lf_flab	0.25	0.14	0.54	none	leaf litter labile fraction
λ_{31}	lf_fcel	0.5	0.37	0.49	none	leaf litter cellulose fraction
λ_{32}	lf_flig	0.25	0.1	0.38	none	leaf litter lignin fraction
λ_{33}	fr_flab	0.25	0.18	0.25	none	fine root labile fraction
λ_{34}	fr_fcel	0.5	0.38	0.5	none	fine root cellulose fraction
λ_{35}	fr_flig	0.25	0.16	0.36	none	fine root lignin fraction
λ_{36}	leaf_long	1.5	2	10	yr	leaf longevity
λ_{37}	resist	0.12	0	0.5	none	fire resistance index
λ_{38}	grperc	0.3	0.2	0.4	none	growth respiration factor 1
λ_{39}	grpnow	1	0	1	none	growth respiration factor 2
λ_{40}	bdnr	0.25	0	0.8	(1/s)	bulk denitrification rate

determined by the other two. With the appropriate shifting of the indices, we will be left with $d = \tilde{d} - n_t$ input parameters, where n_t is the number of input parameter triples that sum up to one.

Given a vector of random variables $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d)$ with known joint cumulative distribution function (CDF) $F(\lambda_1, \ldots, \lambda_d)$, one can obtain a set of η_i's that are independent uniform random

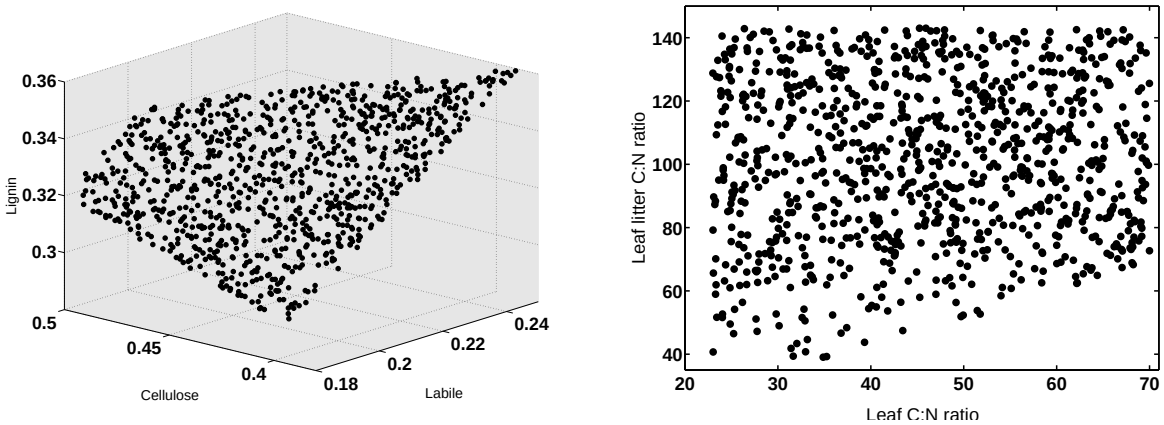
Table 4.2. CLM input parameters: part two

Notation	Name	Default	Min	Max	Units	Description
λ_{41}	dayscrecover	300	1	90	days	days to recover negative cpool
λ_{42}	rc_npool	100	0.5	50	none	resistance for uptake from plant npool
λ_{43}	br_mr	$2.53e-06$	$4e-07$	$1e-05$	gC/gN/s	base rate for maintenance respiration
λ_{44}	q10_mr	1.5	1	4.5	none	q10 for maintenance respiration
λ_{45}	cn_s1	12	8	20	gC/gN	carbon:nitrogen for SOM 1
λ_{46}	cn_s2	12	8	20	gC/gN	carbon:nitrogen for SOM 2
λ_{47}	cn_s3	10	6	20	gC/gN	carbon:nitrogen for SOM 3
λ_{48}	cn_s4	10	6	20	gC/gN	carbon:nitrogen for SOM 4
λ_{49}	rf_l1s1	0.39	0.35	0.45	none	resp. fraction for litter 1 $\rightarrow$ SOM 1
λ_{50}	rf_l2s2	0.55	0.385	0.715	none	resp. fraction for litter 2 $\rightarrow$ SOM 2
λ_{51}	rf_l3s3	0.29	0	0.9	none	resp. fraction for litter 3 $\rightarrow$ SOM 3
λ_{52}	rf_s1s2	0.28	0.26	0.3	none	resp. fraction for SOM 1 $\rightarrow$ SOM 2
λ_{53}	rf_s2s3	0.46	0.032	0.6	none	resp. fraction for SOM 2 $\rightarrow$ SOM 3
λ_{54}	rf_s3s4	0.55	0	1	none	resp. fraction for SOM 3 $\rightarrow$ SOM 4
λ_{55}	k_l1	1.2	0.9	1.5	1/day	decomp rate for litter 1
λ_{56}	k_l2	0.0726	0.05	0.1	1/day	decomp rate for litter 2
λ_{57}	k_l3	0.0141	0.005	0.028	1/day	decomp rate for litter 3
λ_{58}	k_s1	0.0726	0.038	0.11	1/day	decomp rate for SOM 1
λ_{59}	k_s2	0.0141	0.005	0.022	1/day	decomp rate for SOM 2
λ_{60}	k_s3	0.0014	0.0004	0.005	1/day	decomp rate for SOM 3
λ_{61}	k_s4	0.0001	0	0.0004	1/day	decomp rate for SOM 4
λ_{62}	k_frag	0.001	0.0002	0.005	1/day	fragmentation rate for CWD
λ_{63}	cwd_fcel	0.769	0.66	0.81	none	fraction of cellulose in CWD
λ_{64}	dnp	0.01	0.001	0.1	none	denitrification proportion
λ_{65}	minpsi_hr	-10	-15	-5	MPa	minimum psi for heterotrophic resp
λ_{66}	q10_hr	1.5	1	4.5	none	q10 for heterotrophic respiration
λ_{67}	r_mort	0.02	0.002	0.2	1/year	mortality rate
λ_{68}	sf_minn	0.1	0.02	0.4	none	solulble fraction of mineral N
λ_{69}	crit_dayl	39300	35000	45000	seconds	critical daylength for senescence onset
λ_{70}	ndays_on	30	5	60	days	no. of days to complete leaf onset
λ_{71}	ndays_off	15	5	40	days	no. of days to complete leaf offset
λ_{72}	fstor2tran	0.5	0.1	1	none	fraction of strage to move to transfer
λ_{73}	crit_onset_fdd	15	5	30	days	no. of freezing days to set GDD counter
λ_{74}	crit_onset_swi	15	5	30	days	no. of water stress-free days for leaf onset
λ_{75}	soilpsi_on	-2	-5	-0.75	MPa	critical soil water potential for leaf onset
λ_{76}	crit_offset_fdd	15	5	30	days	no. of freezing days for leaf offset
λ_{77}	crit_offset_swi	15	5	30	days	no. of water stress days for leaf offset
λ_{78}	soilpsi_off	-2	-5	-0.75	MPa	critical soil water potential for leaf offset
λ_{79}	lwtop_ann	0.7	0.5	1	1/year	live wood turnover proportion
λ_{80}	gddfunc_p1	4.8	3	7	none	gdd threshold parameter 1
λ_{81}	gddfunc_p2	0.13	0.05	0.3	none	gdd threshold parameter 2

variables on $[-1, 1]$ for all $i = 1, 2, \ldots, d$, using the *scaled* conditional cumulative distributions

$$
\begin{aligned}
\eta_1 &= R_1(\lambda_1) \\
\eta_2 &= R_{2|1}(\lambda_2|\lambda_1) \\
\eta_3 &= R_{3|2,1}(\lambda_3|\lambda_2, \lambda_1) \\
&\vdots \\
\eta_d &= R_{d|d-1,\ldots,1}(\lambda_d|\lambda_{d-1}, \ldots, \lambda_1).
\end{aligned}
\tag{4.2}
$$

Each map $R_*(\cdot)$ is a scaled version of the corresponding CDF $F_*(\cdot)$ to ensure $\eta_i \in [-1, 1]$. That is,



(a) triple $(\lambda_{33}, \lambda_{34}, \lambda_{35})$ **(b)** pair $(\lambda_{18}, \lambda_{22})$

Figure 4.1. Input parameter samples for some of the constrained inputs.

$R_1(\lambda_1) = 2F_1(\lambda_1) - 1$ and, similarly, for the rest of the conditional CDFs in (4.2).

This map, denoted by the shorthand notation $\boldsymbol{\eta} = R(\boldsymbol{\lambda})$, is called the *Rosenblatt transformation* (RT) [18]. Note that the RT is not unique: by ordering the λ_i's in different ways, one can obtain $d!$ different sets of uniform random variables.

The RT will be employed to map the input parameter samples from the $\boldsymbol{\lambda}$-space to the $\boldsymbol{\eta}$-space, or $[-1, 1]^d$. Note that the inverse RT can be used, say, when one needs to obtain CLM inputs corresponding to quadrature points in $[-1, 1]^d$, or when one needs to obtain uniformly distributed samples in the constrained, λ-space.

4.4 Polynomial Basis Reduction via Bayesian Compressive Sensing

In order to have proper coverage of the input parameter space that respects the constraints and uses all available information, the training set of input parameters is taken to be uniformly distributed on the constrained space. This is consistent with the maximum entropy principle, see [12], for instance. With Rosenblatt transformation in place, which maps the input parameters $\boldsymbol{\lambda}$ to a uniform random vector $\boldsymbol{\eta}$, one can build Polynomial Chaos expansion with respect to $\boldsymbol{\eta}$, as described in Section 1.1.1. In other words, the forward function is evaluated at the training input data set to arrive at input-output pairs $(\boldsymbol{\lambda}_i, f(\boldsymbol{\lambda}_i))$ for $i = 1, 2, \ldots, N$. With the RT in mind, we are seeking

an expansion of the form

$$\tilde{f}_{\boldsymbol{c}}(\boldsymbol{\eta}) \approx \sum_{k=0}^{K} c_k \Psi_k(\boldsymbol{\eta}) \tag{4.3}$$

to serve as a surrogate. In terms of the CLM input parameters, the surrogate will take the form

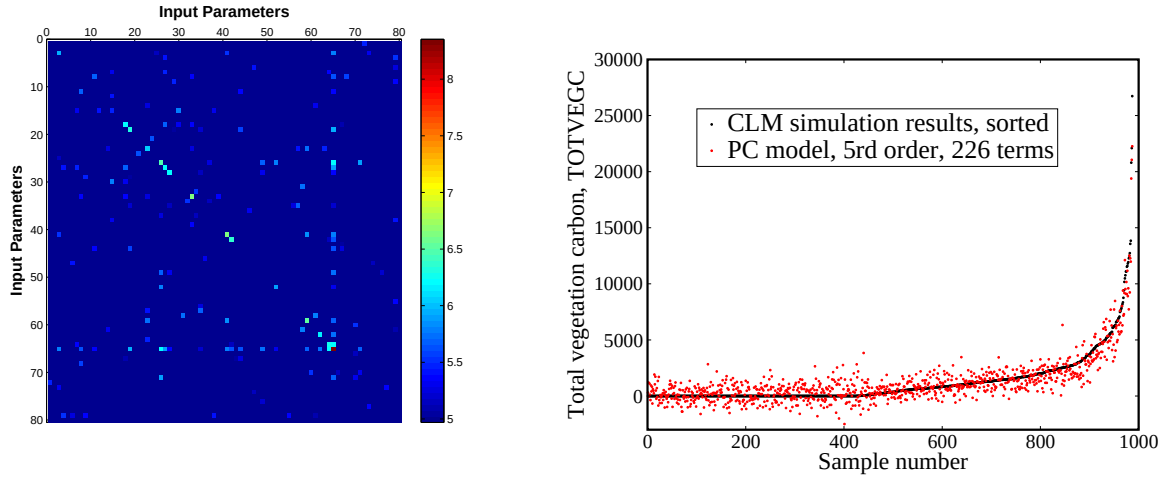
$$f_{\boldsymbol{c}}(\boldsymbol{\lambda}) \approx \sum_{k=0}^{K} c_k \Psi_k(R(\boldsymbol{\lambda})). \tag{4.4}$$

Table 4.3. CLM output quantities of interest

Notation	Name	Units	Description
y_1	TOTVEGC	gC/m^2	Total vegetation carbon
y_2	TOTSOMC	gC/m^2	Total soil carbon
y_3	GPP	gC/m^2/s	Gross primary production
y_4	ERR	W/m^2	Energy conservation error
y_5	TLAI	none	Total leaf area index
y_6	EFLX_LH_TOT	W/m^2	Total latent heat flux
y_7	FSH	W/m^2	Sensible heat flux

We applied the iterative BCS algorithm of polynomial basis reduction, described in Section 1.1.3, to the CLM with 7 output quantities of interest (QoI), shown in Table 4.3. For each QoI, a 10-year average of a 1000-year CLM simulation is taken.

Let us study the total vegetation carbon (output TOTVEGC) more closely. We rely on $N = 987$ training simulations sampled uniformly in the constrained parameter space. We run the iterative BCS algorithm starting with the second order ($l_0 = 2$ and, therefore, 3240 basis terms initially) up to the fifth order. The first step reveals all the second order terms that are important if one had to represent the forward function with only second order polynomial expansion. Figure 4.2(a) shows a matrix of important dimensions and couplings for the resulting reduced second order basis. The diagonal terms correspond to the sum of the logarithms of the absolute values of the PC modes corresponding to basis terms $\psi_1(\eta_i) = \eta_i$ and $\psi_2(\eta_i) = (3\eta_i^2 - 1)/2$, while the off-diagonal terms are the logarithms of the absolute values of the PC modes for the terms $\eta_i\eta_j$. For clarity of presentation, these joint PC modes are split to entries in the 'matrix' in Figure 4.2(a). Furthermore, the iterative BCS algorithm is carried out up to the 5-th order, leading to a PC representation with only 226 terms. As a comparison, a second order, 79-dimensional PC basis without reduction has over 3000 terms. Figure 4.2(b) shows the output values, sorted for more clear visualization, as well as the PC representation values evaluated at the same training points. Note that, while the overall trend is captured, the strongly nonlinear behavior of the output renders the PC representation imprecise in the regions with low or no vegetation, i.e. where the output $y_1 \approx 0$. As can be seen in Figure 4.2(b), about half of the samples lead to zero vegetation, i.e. $y_1 = 0$. In principle, one should identify the regions in the input parameter space that correspond to low vegetation and split the input domain accordingly, leading to a mixture PC representation [19]. The study of such classification approaches in high-dimensional input spaces is outside the scope of the current report and will be carried out in future.



(a) Matrix of relevant input parameter couplings for output TOTVEGC

(b) Training data (ordered) and their PC representation

Figure 4.2. Important parameter couplings and reduced basis representation for the output TOTVEGC.

Figure 4.4 reports the matrices of relevant input variable couplings for the other six QoIs, while Table 4.4 shows the first 10 important dimensions for each output by running a single, first order BCS.

Table 4.4. Ten most important parameters for each output.

rank	TOTVEGC	TOTSOMC	GPP	ERR	TLAI	EFLX_LH_TOT	FSH
1	r_mort	q10_mr	leafcn	k_s4	froot_leaf	leafcn	rholnir
2	q10_mr	leafcn	k_s4	froot_leaf	q10_mr	q10_mr	q10_mr
3	froot_leaf	froot_leaf	froot_leaf	q10_hr	q10_hr	froot_leaf	leafcn
4	br_mr	br_mr	flnr	fflnr	leaf_long	k_s4	br_mr
5	q10_mr	fflnr	q10_mr	q10_mr	k_s4	br_mr	flnr
6	leafcn	dnp	q10_hr	dnp	br_mr	flnr	k_s4
7	k_s4	q10_hr	dnp	rf_s3s4	dnp	leaf_long	taulnir
8	stem_leaf	leaf_long	rf_s3s4	leaf_long	stem_leaf	q10_hr	froot_leaf
9	flnr	k_s4	leaf_long	mp	r_mort	rf_s3s4	frootcn
10	dnp	frootcn	br_mr	bdnr	rf_s3s4	stem_leaf	f_frag

4.5 Exploration of the parameter space

Several simulation ensembles were run to asses the behavior of select model observables for a range of values for relevant model parameters identified in by the BCS analysis described in Sec-

tion 1.1.2. These model parameters are shown in Table 4.3. The initial conditions were generated through three model spinup steps as outlined below:

- Accelerated decomposition (AD) spinup - a 600 year simulation with soil C and N pool turnover times reduced by a factor of 20 to accelerate equilibration.
- Exit spinup - a one year simulation with normal C and N turnover times.
- Final spinup - a 1000 year simulation to re-equilibrate the model with the correct C and N turnover time.

Time series for the total vegetation carbon (TOTVEGC), total soil carbon (totsomc) and total leaf area index (TLAI), corresponding to AD and final spinup runs are shown in Fig. 4.4. The inset plots zoom in on the last 30 years of the final spinup run. Despite the fact that the model did not reach a quasi-steady state at the end of the final spinup, the decadal changes are small enough to enable sensitivity studies using these results are initial conditions.

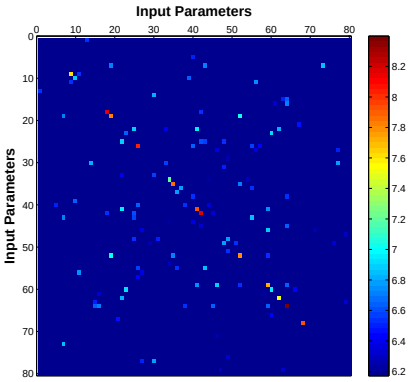
Starting with the model state at the end of the final spinup run, an ensemble of 49 runs was performed to study the effect of λ_{26} ("froot_leaf") and λ_{67} ("r_mort") parameters on the output observables of interest. The values for the other 79 parameters out of the 81 considered for this study were set at the nominal values listed in Tables 4.1 and 4.2. The values for λ_{26} and λ_{67} were chosen to span the physical ranges for these parameters, also listed in Tables 4.1-4.2.

Figure 4.5 shows average values for select CLM output observables. The averages are taken over the last 10 years of 1000 year simulations for various pairs $(\lambda_{26}, \lambda_{67})$. These results show a shart drop in vegetation for certain combinations of λ_{26} and $\lambda_{67})$ values. Time series for TOTVEGC and TOTSOMC are shown in Fig. 4.6. These time series correspond to the runs shown with filled circles in the same figure.

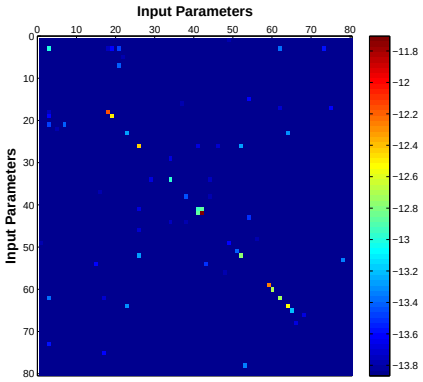
4.6 Porting CLM to Sandia

The CLM software framework was installed at SNL to enable proof-of-concept studies for model calibration and uncertainty quantification. Several prerequisite software libraries were required prior to building CLM. The HDF5 (version 1.8.5), netcdf (version 3.6.2), NCL (version 6) as well as Python's Numeric, Scientific.IO, Numpy, and Scipy modules were installed.

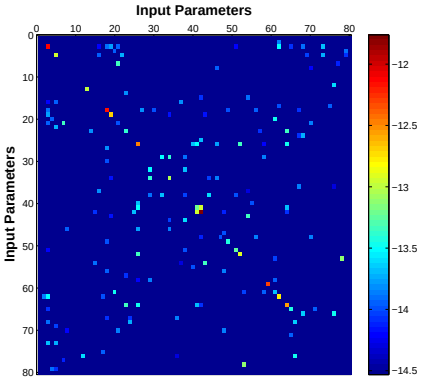
CLM was built using PGI compilers (version 9.0). Most of the high level build tasks are handled by Python scripts included in the distribution. These scripts incorporate case-specific keywords and generate directory trees with setup data, and executable files required for each simulation.



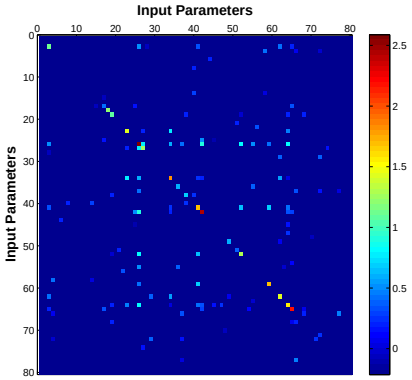
(a) TOTSOMC



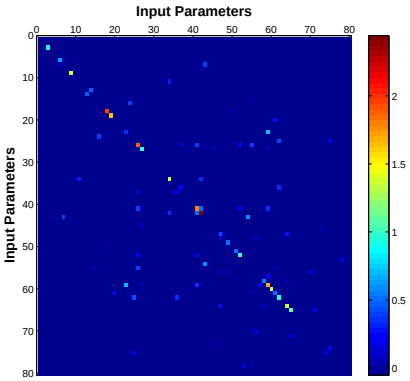
(b) GPP



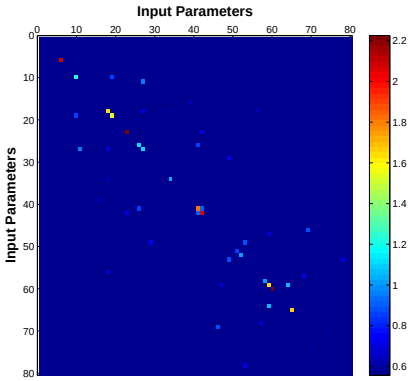
(c) ERR



(d) TLAI



(e) EFLX_LH_TOT



(f) FSH

Figure 4.3. Matrix of relevant input parameter couplings for six different outputs

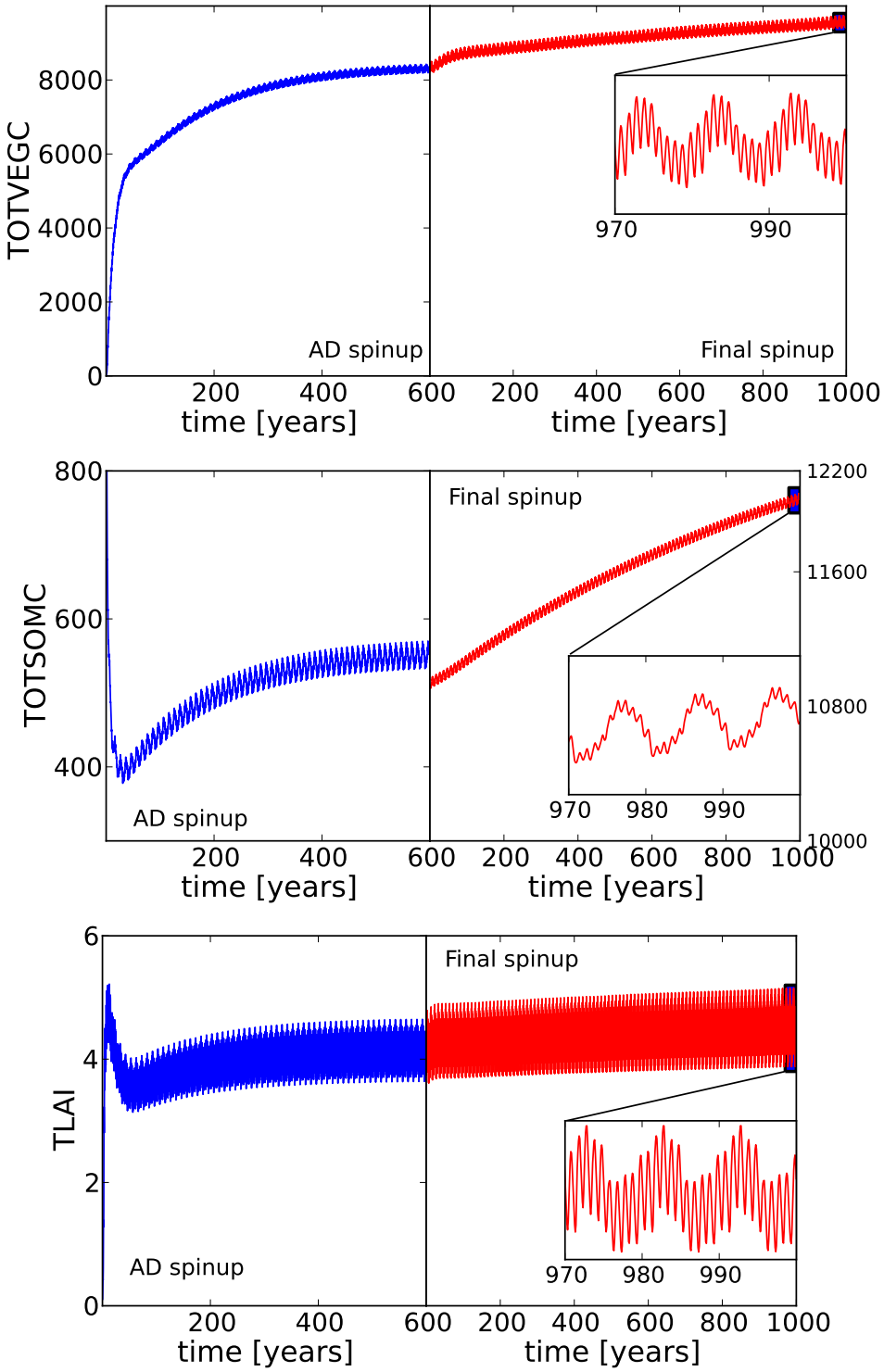


Figure 4.4. Time series for select CLM observables during the solution spinup.

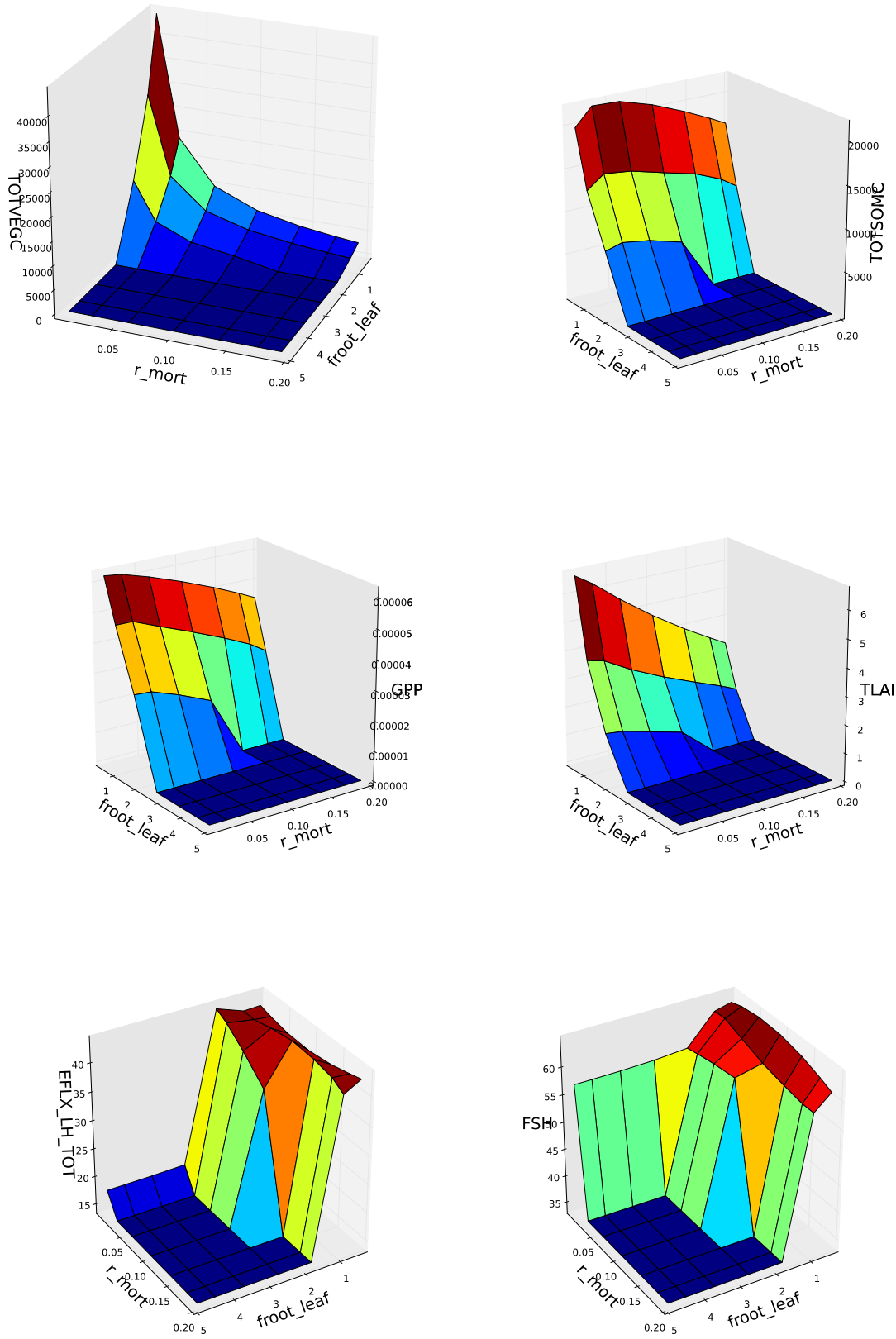


Figure 4.5. CLM observables corresponding to an ensemble of simulations spanning a 2D grid of "froot_leaf" and "r_mort" partameter values. The ouput observables are averages over the last 10 years from 1000 year simulations.

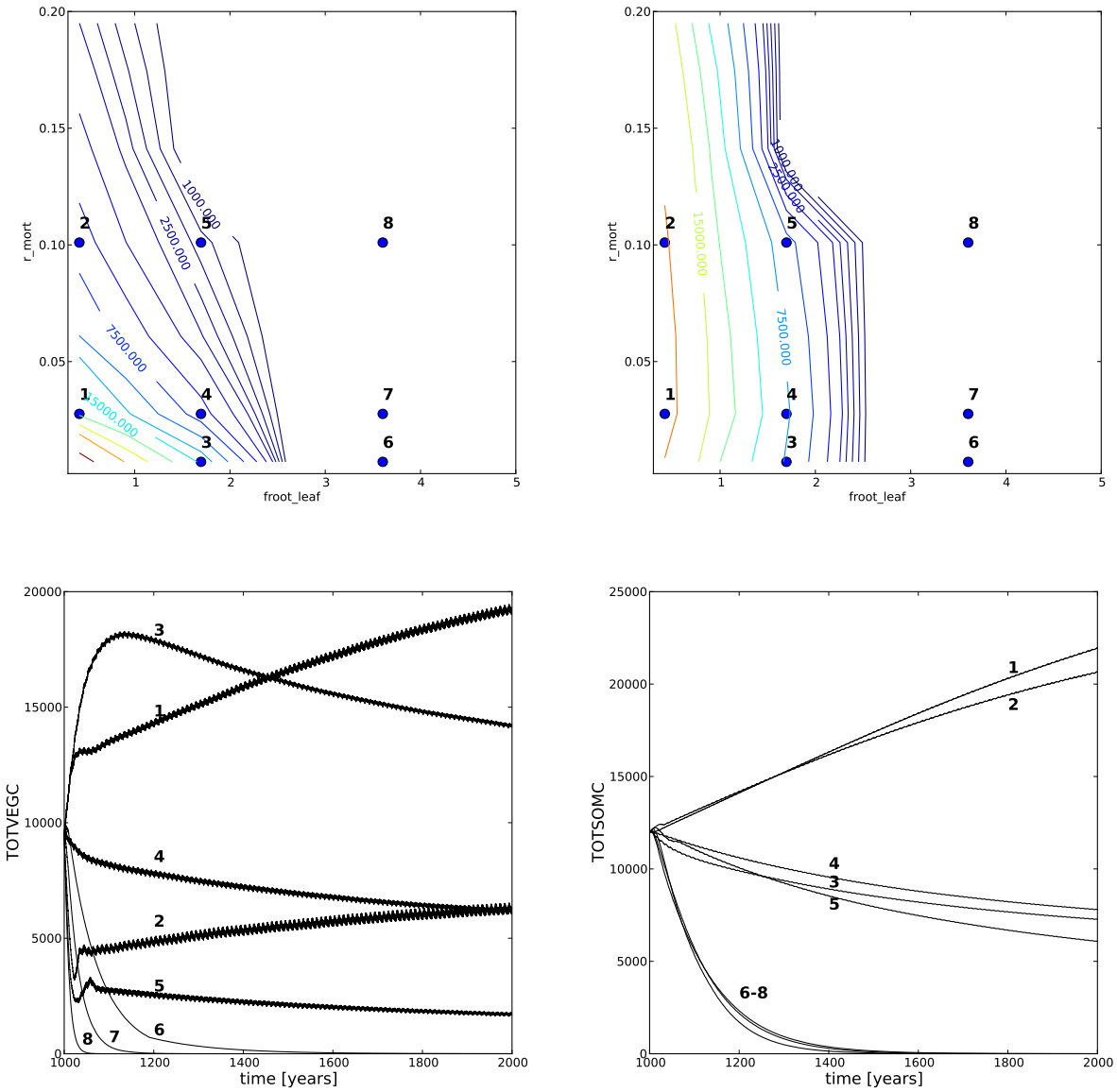


Figure 4.6. Time evolution of "totvegc" and "totsomc" for select runs in the ensemble shown in Fig. 4.5. The parameter values for the corresponding runs are shown with blue circles in the contour plots for these observables.

References

[1] Climate Science for a Sustainable Energy Future. A proposal to the US DoE, Ofice of Science; contact the authors.

[2] PEST Homepage: Model-independent parameter estimation. http://www.pesthomepage.org/.

[3] J. M. R. Byrd, S. A. Jarvis, and A. H. Bhalerao. On the parallelisation of MCMC by speculative chain execution. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1 –8, april 2010.

[4] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1997.

[5] K. W. Oleson et al. Technical description of version 4.0 of the Community Land Model. NCAR Technical Note NCAR/TN-478+STR, National Center for Atmospheric Research, PO Box 3000, Boulder Colorado 80307-300, April 2010. http://www.cesm.ucar.edu/models/ccsm4.0/clm/CLM4_Tech_Note.pdf.

[6] C. M. Feldhake and D. G. Boyer. Effects of soil temperature on evapotranspiration by C3 and C4 grasses. *Agricultural and Forest Meteorology*, 37:309–318, 1986.

[7] R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer Verlag, New York, 1991.

[8] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[9] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[10] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[11] Heikki Haario, Marko Laine, Antoinietta Mira, and Eero Saksman. DRAM-Efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, 2006.

[12] E.T. Jaynes. *Probability Theory: The Logic of Science*, G.L. Bretthorst, Ed. Cambridge University Press, Cambridge, UK, 2003.

[13] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer, New York, 2004.

[14] H. Lee, D. Higdon, Z. Bi, M. Ferreira, and M. West. Markov random field models of high-dimensional parameters in simulations of fluid flow in porous media. *Technometrics*, 44(3):230–241, 2002.

[15] T. Patterson. The optimum addition of points to quadrature formulae. *Mathematics of Computation*, 22(104):847–856, 1968.

[16] J. Ray, S. Lefantzi, K. Klise, L. Salazar, S. A. McKenna, B. van Bloemen Waanders, M. D. Parno, and Y. M. Marzouk. Bayesian data assimilation for stochastic multiscale models of transport in porous media. SAND Report SAND2011-6811, Sandia National Laboratories, Livermore, CA 94551-0969, October 2011. Unclassified and unlimited release.

[17] Clive D. Rodgers. *Inverse methods for atmospheric sounding*. Worl Scientific, Singapore, 2008. Chapter 4.

[18] M. Rosenblatt. Remarks on a multivariate transformation. *Annals of Mathematical Statistics*, 23(3):470 – 472, 1952.

[19] K. Sargsyan, C. Safta, B. Debusschere, and H. Najm. Multiparameter spectral representation of noise-induced competence in Bacillus subtilis. *in preparation*, 2011.

[20] D.S. Sivia. *Data Analysis: A Bayesian Tutorial*. Oxford Science, 1996.

[21] Cajo J. F. ter Braak and Jasper A. Vrugt. Differential Evolution Markov Chain with snooker updater and fewer chains. *Statistical Computing*, 18:435–446, 2008.

[22] J. A. Vrugt, H. V. Gupta, W. Bouten, and S. Sorooshian. A shuffled complex evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. *Water Resources Research*, 39(8), 2003.

[23] J. A. Vrugt, B. Ó. Nualláin, B. A. Robinson, Willem Bouten, Stefan C. Dekker, and Peter M. A. Sloot. Application of parallel computing to stochastic parameter estimation in environmental models. *Computers and Geosciences*, 32:1139–1155, 2006.

[24] J. A. Vrugt, B. A. Robinson, and V. V. Vesselinov. Improved inverse modeling for flow and transport in subsurface media: Combined parameter and state estimation. *Geophysical Research Letters*, 32:L18408–, 2005.

[25] J. A. Vrugt, C. J. F. ter Braak, C. G. H Diks, B. A. Robinson, J. M. Hyman, and D. Higdon. Accelerating Markov chain Monte Carlo simulation by self-adaptive differential evolution with randomized subspace sampling. *International Journal of Nonlinear Scientific Numerical Simulation*, 10(3), 2009.

[26] N. Wiener. The homogeneous chaos. *Am. J. Math.*, 60:897–936, 1938.

[27] Xubin Zeng and Mark Decker. Improving the numerical solution of soil-mositure based Richard's equation for land models with a deep or shallow water table. *Journal of Hydrometeorology*, 10(1):308–319, 2009.

[28] S-W Zhang, X. Zeng, W. Zhang, and M. Barlage. Revising the Ensemble-based Kalman filter covariance for the retrieval of deep-layer soil moisture. *Journal of Hydrometeorolog*, 11(1):219–227, 2010.

DISTRIBUTION:

1	MS	1320	Scott Collis, 1442 (electronic copy)
1	MS	1325	John Mitchiner, 1460 (electronic copy)
1	MS	9051	Khachik Sargsyan, 8351 (electronic copy)
1	MS	9051	Cosmin Safta, 8954 (electronic copy)
1	MS	9051	Robert Berry, 8351 (electronic copy)
1	MS	9155	Jaideep Ray, 8954 (electronic copy)
1	MS	9051	Bert Debusschere, 8351 (electronic copy)
1	MS	9051	Habib Najm, 8351 (electronic copy)
1	MS	1318	Laura Swiler, 1441 (electronic copy)
1	MS	0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories