The *Albany/FELIX* First-Order Stokes Finite Element Ice Sheet Dynamical Core Built Using Trilinos Software Components: Performance, Next-Generation Capabilities and Validation

I. Tezaur¹, A. Salinger¹, M. Perego¹, R. Tuminaro¹, S. Price², J. Watkins¹, G. Hansen¹

¹ Sandia National Laboratories, Albuquerque, NM and Livermore, CA, USA.
² Los Alamos National Laboratory, Los Alamos, NM, USA.

PISCEES* Project for Land-Ice Modeling

Sandia’s Role in the PISCEES Project: to develop and support a robust and scalable land ice solver based on the “First-Order” (FO) Stokes equations → Albany/FELIX**

Requirements for Albany/FELIX:

- **Unstructured grid** finite elements.
- **Scalable, fast** and **robust**
- **Verified** and **validated**.
- **Portable** to new/emerging architecture machines (multi-core, many-core, GPU)
- **Advanced analysis** capabilities: deterministic inversion, calibration, uncertainty quantification.

As part of ACME DOE earth system model, solver will provide actionable predictions of 21st century sea-level rise (including uncertainty).

**Finite Elements for Land Ice eXperiments**

*PISCEES = Predicting Ice Sheet Climate Evolution at Extreme Scales.*
PISCEES* Project for Land-Ice Modeling

Sandia’s Role in the PISCEES Project: to develop and support a robust and scalable land ice solver based on the “First-Order” (FO) Stokes equations → Albany/FELIX**

Requirements for Albany/FELIX:

- **Unstructured grid** finite elements.
- **Scalable, fast** and **robust**
- **Verified** and **validated**.
- **Portable** to new/emerging architecture machines (multi-core, many-core, GPU)
- **Advanced analysis** capabilities: deterministic inversion, calibration, uncertainty quantification.

As part of ACME DOE earth system model, solver will provide actionable predictions of 21st century sea-level rise (including uncertainty).

**Finite Elements for Land Ice eXperiments**

*PISCEES = Predicting Ice Sheet Climate Evolution at Extreme Scales.*
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.

Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.


Steps towards creating a production-ready code
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.

Ice Sheet Equations and Codes

**Momentum Balance:** *First-Order Stokes* PDEs

\[
\begin{align*}
- \nabla \cdot (2\mu \dot{e}_1) &= -\rho g \frac{\partial s}{\partial x}, \quad \text{in } \Omega \\
- \nabla \cdot (2\mu \dot{e}_2) &= -\rho g \frac{\partial s}{\partial y}
\end{align*}
\]

with *Glen’s law* viscosity \( \mu = \frac{1}{2} A(T)^{-\frac{1}{3}} \left( \frac{1}{2} \sum_{ij} \dot{e}_{ij}^2 \right)^{\left( -\frac{2}{3} \right)} \).

**Conservation of Mass:** *thickness* evolution PDE

\[
\frac{\partial h}{\partial t} = - \nabla \cdot (\bar{u}h) + \dot{b}
\]

**Energy Balance:** *temperature* advection-diffusion PDE

\[
\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) - \rho c \bar{u} \cdot \nabla T + 2 \dot{\varepsilon} \sigma
\]
Ice Sheet Equations and Codes

**Momentum Balance:** *First-Order Stokes* PDEs

\[
\begin{align*}
-\nabla \cdot (2\mu \dot{e}_1) &= -\rho g \frac{\partial s}{\partial x}, \quad \text{in } \Omega \\
-\nabla \cdot (2\mu \dot{e}_2) &= -\rho g \frac{\partial s}{\partial y}
\end{align*}
\]

with *Glen’s law* viscosity \( \mu = \frac{1}{2} A(T)^{-\frac{1}{3}} \left( \frac{1}{2} \sum_{ij} \dot{e}_{ij}^2 \right)^{\left(\frac{-2}{3}\right)} \).

**Conservation of Mass:** *thickness* evolution PDE

\[
\frac{\partial h}{\partial t} = -\nabla \cdot (\overline{uh}) + \dot{b}
\]

**Energy Balance:** *temperature* advection-diffusion PDE

\[
\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k\nabla T) - \rho c \mathbf{u} \cdot \nabla T + 2\dot{\varepsilon} \sigma
\]

Ice Sheet Equations and Codes

**Momentum Balance:** *First-Order Stokes* PDEs

\[
\begin{align*}
-\nabla \cdot (2\mu \dot{e}_1) &= -\rho g \frac{\partial s}{\partial x}, \quad \text{in } \Omega \\
-\nabla \cdot (2\mu \dot{e}_2) &= -\rho g \frac{\partial s}{\partial y}
\end{align*}
\]

with *Glen’s law* viscosity \( \mu = \frac{1}{2} A(T)^{-\frac{1}{3}} \left( \frac{1}{2} \sum_{ij} \dot{e}^{ij}_{ij} \right)^{\frac{2}{3}} \).

**Conservation of Mass:** *thickness* evolution PDE

\[
\frac{\partial h}{\partial t} = -\nabla \cdot (\bar{u}h) + \dot{b}
\]

**Energy Balance:** *temperature* advection-diffusion PDE

\[
\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) - \rho c \bar{u} \cdot \nabla T + 2\dot{\varepsilon} \sigma
\]

*https://github.com/gahansen/Albany. **FELIX = Finite Elements for Land Ice eXperiments.*
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.


Steps towards creating a production-ready code
Mesh Adaptivity

Why?

New capability for land-ice solver!
Mesh Adaptivity

Why? To *concentrate* computational power where it is needed.

New capability for land-ice solver!
Mesh Adaptivity

How?

New capability for land-ice solver!
Mesh Adaptivity

How?

PAALS = Parallel Albany Adaptive Loop with SCOREC

New capability for land-ice solver!
Mesh Adaptivity

How?

**PAALS** = Parallel Albany Adaptive Loop with SCOREC*

- In collaboration with **Rensselaer Polytechnical Institute** (M. Shephard, C. Smith, D. Ibanez): added mesh adaptation capabilities (PAALS) to **Albany**.

*SCOREC = Scientific Computation Research Center at RPI: https://github.com/SCOREC

New capability for land-ice solver!
Mesh Adaptivity

How?

PAALS = Parallel Albany Adaptive Loop with SCOREC*

• In collaboration with Rensselaer Polytechnical Institute (M. Shephard, C. Smith, D. Ibanez): added mesh adaptation capabilities (PAALS) to Albany.

PAALS provides:

• Fully-coupled, in-memory adaptation and solution transfer services.

*SCOREC = Scientific Computation Research Center at RPI: https://github.com/SCOREC

New capability for land-ice solver!
Mesh Adaptivity

**How?**

**PAALS** = Parallel Albany Adaptive Loop with SCOREC*

- In collaboration with **Rensselaer Polytechnical Institute** (M. Shephard, C. Smith, D. Ibanez): added mesh adaptation capabilities (PAALS) to **Albany**.

**PAALS provides:**

- Fully-coupled, *in-memory adaptation* and solution transfer services.
- **Parallel mesh infrastructure** and services via **PUMI** (Parallel Unstructured Mesh Infrastructure): an efficient, distributed mesh data structure that supports adaptivity.

* **SCOREC** = Scientific Computation Research Center at RPI: https://github.com/SCOREC

New capability for land-ice solver!
Mesh Adaptivity

How?

PAALS = Parallel Albany Adaptive Loop with SCOREC*

- In collaboration with Rensselaer Polytechnical Institute (M. Shephard, C. Smith, D. Ibanez): added mesh adaptation capabilities (PAALS) to Albany.

PAALS provides:

- Fully-coupled, in-memory adaptation and solution transfer services.
- Parallel mesh infrastructure and services via PUMI (Parallel Unstructured Mesh Infrastructure): an efficient, distributed mesh data structure that supports adaptivity.
- Predictive dynamic load balancing via ParMetis/Zoltan + ParMA.

*SCOREC = Scientific Computation Research Center at RPI: https://github.com/SCOREC

New capability for land-ice solver!
Mesh Adaptivity

In collaboration with Rensselaer Polytechnical Institute (M. Shephard, C. Smith, D. Ibanez): added mesh adaptation capabilities (PAALS) to Albany.

PAALS provides:

- Fully-coupled, *in-memory adaptation* and solution transfer services.
- *Parallel mesh infrastructure* and services via PUMI (Parallel Unstructured Mesh Infrastructure): an efficient, distributed mesh data structure that supports adaptivity.
- Predictive *dynamic load balancing* via ParMetis/Zoltan + ParMA.
- SPR**-based generalized *error estimation* of velocity gradient drives adaptation.

**Ryder glacier (north coast)**

Left: before mesh adaptation; Right: after mesh adaptation

**Super-convergent Patch Recovery: technique for estimating $\nabla u$ using quadratic approximation within a patch of elements.**
Parallel Adaptive Loop with SCOREC (PAALS)

Error bound = 0.5

Error bound = 0.1

Error bound = 0.01
Dynamic Load Balancing

- Partition via **Zoltan** or **ParMetis + ParMA**.

- **Libraries/Algorithms:**
  - **ParMetis**: multi-level graph partitioning $\rightarrow$ minimizes communication
  - **Zoltan**: Recursive Inertial Bisection (RIB) $\rightarrow$ faster than graph partitioning
  - **ParMA**: Unstructured mesh-based diffusive improvement $\rightarrow$ rebalances refined mesh

---

**In each box below**
- **Left**: Initial ParMetis/Zoltan partition
- **Right**: ParMA partition

---

*ParMetis (left) + ParMA (right)*

*Zoltan (left) + ParMA (right)*
Mesh Generation/Adaptation Algorithm

PAALS = Parallel Albany Adaptive Loop with SCOREC

- **Step 1**: determine geometry and generate initial tetrahedral mesh (e.g., using Triangle or Simmetrix).
- **Step 2**: 2D slice of initial mesh adaptively refined in-memory via PAALS based on gradient of velocity in Albany.
- **Step 3**: 3D mesh obtained by extruding 2D mesh vertically as prisms or tetrahedra in Albany.
Mesh Generation/Adaptation Algorithm

PAALS = Parallel Albany Adaptive Loop with SCOREC

• **Step 1**: determine geometry and generate initial tetrahedral mesh (e.g., using *Triangle* or *Simmetrix*).

• **Step 2**: 2D slice of initial mesh adaptively refined in-memory via *PAALS* based on gradient of velocity in *Albany*.

• **Step 3**: 3D mesh obtained by extruding 2D mesh vertically as *prisms* or *tetrahedra* in *Albany*.

**Discussion:**
Mesh Generation/Adaptation Algorithm

PAALS = Parallel Albany Adaptive Loop with SCOREC

- **Step 1**: determine geometry and generate initial tetrahedral mesh (e.g., using Triangle or Simmetrix).
- **Step 2**: 2D slice of initial mesh adaptively refined in-memory via PAALS based on gradient of velocity in Albany.
- **Step 3**: 3D mesh obtained by extruding 2D mesh vertically as prisms or tetrahedra in Albany.

Discussion:

- PAALS adaptive loop driven by homotopy continuation using LOCA package in Trilinos.
Mesh Generation/Adaptation Algorithm

PAALS = Parallel Albany Adaptive Loop with SCOREC

offline
• **Step 1**: determine geometry and generate initial tetrahedral mesh (e.g., using Triangle or Simmetrix).

online
• **Step 2**: 2D slice of initial mesh adaptively refined *in-memory* via PAALS based on gradient of velocity in Albany.
• **Step 3**: 3D mesh obtained by extruding 2D mesh vertically as prisms or tetrahedra in Albany.

Discussion:
• PAALS adaptive loop driven by *homotopy continuation* using LOCA package in Trilinos.
• Currently adaptation is done in *stand-alone* Albany.
Mesh Generation/Adaptation Algorithm

PAALS = Parallel Albany Adaptive Loop with SCOREC

**Step 1**: determine geometry and generate initial tetrahedral mesh (e.g., using Triangle or Simmetrix).

**Step 2**: 2D slice of initial mesh adaptively refined in-memory via PAALS based on gradient of velocity in Albany.

**Step 3**: 3D mesh obtained by extruding 2D mesh vertically as prisms or tetrahedra in Albany.

**Discussion**:

- PAALS adaptive loop driven by *homotopy continuation* using LOCA package in Trilinos.
- Currently adaptation is done in *stand-alone* Albany.
- **Future work**: integrating adaptation into *dynamical cores* (MPAS-Albany).

PAALS = Parallel Albany Adaptive Loop with SCOREC
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.


Steps towards creating a production-ready code
Initial Weak Scalability Study Using ILU

Greenland Ice Sheet

Antarctic Ice Sheet
Initial Weak Scalability Study Using ILU

Scalability results are **not** acceptable!
Initial Weak Scalability Study Using ILU

Scalability results are **not** acceptable!

**Why is scalability so **bad** for out-of-the-box preconditioners?**

1. Ice sheet geometries have **bad aspect ratios** \((dx \gg dz)\).
2. **Ice shelves** give rise to **severely ill-conditioned** matrices.
3. **Islands** and **hinged peninsulas** lead to **solver failures**.
Initial Weak Scalability Study Using ILU

**Scalability results are not acceptable!**

Why is scalability so bad for out-of-the-box preconditioners?

1. Ice sheet geometries have bad aspect ratios ($dx \gg dz$).
2. Ice shelves give rise to severely ill-conditioned matrices.
3. Islands and hinged peninsulas lead to solver failures.

We **mitigate** these difficulties through the development of:

- New **AMG preconditioner** based on **semi-coarsening**.
- **Island/hinge removal algorithm**.
Scalability via Algebraic Multi-Grid Preconditioning with Semi-Coarsening

Bad aspect ratios \((dx \gg dz)\) ruin classical AMG convergence rates!
- relatively small horizontal coupling terms, hard to smooth horizontal errors
\(\Rightarrow\) Solvers (AMG and ILU) must take aspect ratios into account

We developed a **new AMG solver** based on aggressive **semi-coarsening** (available in ML/MueLu packages of Trilinos)

See (Tezaur et al., 2015), (Tuminaro et al., 2016).

*Scaling studies (next slides):* New AMG preconditioner vs. ILU
Greenland Controlled Weak Scalability Study

- Weak scaling study with fixed dataset, 4 mesh bisections.
- ~70-80K dofs/core.
- **Conjugate Gradient (CG) iterative method** for linear solves (faster convergence than GMRES).
- **New AMG preconditioner** developed by R. Tuminaro based on semi-coarsening (coarsening in z-direction only).
- **Significant improvement** in scalability with new AMG preconditioner over ILU preconditioner!

![Graph showing weak scalability](attachment:image.png)

- 4 cores
  - 334K dofs
  - 8 km Greenland, 5 vertical layers

- 16,384 cores
  - 1.12B dofs (!)
  - 0.5 km Greenland, 80 vertical layers

\[ \times 8^4 \text{ scale up} \]
Greenland Controlled Weak Scalability Study

- Weak scaling study with fixed dataset, 4 mesh bisections.

- \( \sim 70 \text{--} 80K \) dofs/core.

- Conjugate Gradient (CG) iterative method for linear solves (faster convergence than GMRES).

- New AMG preconditioner developed by R. Tuminaro based on semi-coarsening (coarsening in \( z \)-direction only).

- Significant improvement in scalability with new AMG preconditioner over ILU preconditioner!

**New AMG preconditioner**

**ILU preconditioner**

- 4 cores: 334K dofs, 8 km Greenland, 5 vertical layers

- Scale up: \( \times 8^4 \)

- 16,384 cores: 1.12B dofs (!), 0.5 km Greenland, 80 vertical layers

- **Significant improvement** in scalability with new AMG preconditioner over ILU preconditioner!
Moderate Resolution Antarctica Weak Scaling Study

Antarctica is fundamentally different than Greenland: AIS contains large ice shelves (floating extensions of land ice).

- **Along ice shelf front:** open-ocean BC (Neumann).
- **Along ice shelf base:** zero traction BC (Neumann).

⇒ For vertical grid lines that lie within ice shelves, top and bottom BCs resemble Neumann BCs so sub-matrix associated with one of these lines is almost* singular.

(\text{vertical} > \text{horizontal coupling})

+ Neumann BCs

= nearly singular submatrix associated with vertical lines

⇒ Ice shelves give rise to severe ill-conditioning of linear systems!

⇒ Ice shelves give rise to severe ill-conditioning of linear systems!

*Completely singular in the presence of islands and some ice tongues.
Moderate Resolution Antarctica Weak Scaling Study

- Weak scaling study on Antarctic problem (8km w/ 5 layers → 2km w/ 20 layers).
- Initialized with realistic basal friction (from deterministic inversion) and temperature field from BEDMAP2.
- **Iterative linear solver**: GMRES.
- **Preconditioner**: ILU vs. new AMG based on aggressive semi-coarsening.

Severe ill-conditioning caused by ice shelves!

AMG preconditioner less sensitive than ILU to ill-conditioning (ice shelves → Green’s function with modest horizontal decay → ILU is less effective).
Improved Linear Solver Performance through Removal of Hinged Peninsulas

Islands and certain hinged peninsulas lead to **solver failures**

- We have developed an algorithm to detect/remove problematic **hinged peninsulas** & **islands** based on coloring and repeated use of connected component algorithms (Tuminaro et al., 2016).
- Solves are ~2x faster with hinges removed.
- Current implementation is MATLAB, but working on C++ implementation for integration into dycores.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>ILU – hinges</th>
<th>ILU – no hinges</th>
<th>ML – hinges</th>
<th>ML – no hinges</th>
</tr>
</thead>
<tbody>
<tr>
<td>8km/5 layers</td>
<td>878 sec, 84 iter/solve</td>
<td>693 sec, 71 iter/solve</td>
<td>254 sec, 11 iter/solve</td>
<td>220 sec, 9 iter/solve</td>
</tr>
<tr>
<td>4km/10 layers</td>
<td>1953 sec, 160 iter/solve</td>
<td>1969 sec, 160 iter/solve</td>
<td>285 sec, 13 iter/solve</td>
<td>245 sec, 12 iter/solve</td>
</tr>
<tr>
<td>2km/20 layers</td>
<td>10942 sec, 710 iter/solve</td>
<td>5576 sec, 426 iter/solve</td>
<td>482 sec, 24 iter/solve</td>
<td>294 sec, 15 iter/solve</td>
</tr>
<tr>
<td>1km/40 layers</td>
<td>--</td>
<td>15716 sec, 881 iter/solve</td>
<td>668 sec, 34 iter/solve</td>
<td>378 sec, 20 iter/solve</td>
</tr>
</tbody>
</table>

*Greenland Problem*
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.

Performance-Portability via *Kokkos*

We need to be able to run *Albany/FELIX* on **new architecture machines** (hybrid systems) and **manycore devices** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).
Performance-Portability via \textit{Kokkos}

We need to be able to run \textit{Albany/FELIX} on \textit{new architecture machines} (hybrid systems) and \textit{manycore devices} (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).

\textbf{MPI (task parallelism) + X* (thread + data-level parallelism)}

\*X = OpenMP, CUDA, etc.
Performance-Portability via *Kokkos*

We need to be able to run *Albany/FELIX* on **new architecture machines** (hybrid systems) and **manycore devices** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).

**MPI** (task parallelism) + **X** (thread + data-level parallelism)

- **Kokkos**: open-source library that provides performance portability across diverse devices with different memory models.
  - A **programming model** as much as a software library.
  - Provides automatic access to OpenMP, CUDA, Pthreads, ...
  - Templated meta-programming: parallel_for, parallel_reduce (templated on an **execution space**).
  - Memory layout abstraction (“array of structs” vs. “struct of arrays”, locality).

*With Kokkos, you write an algorithm once, and just change a template parameter to get the optimal data layout for your hardware (e.g., (i,j,k) vs. (k,i,j)).*

*X = OpenMP, CUDA, etc.*
Performance-Portability via *Kokkos*

We need to be able to run *Albany/FELIX* on **new architecture machines** (hybrid systems) and **manycore devices** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).

- **MPI** (task parallelism) + X* (thread + data-level parallelism)

- **Kokkos**: open-source library that provides performance portability across diverse devices with different memory models.
  - A *programming model* as much as a software library.
  - Provides automatic access to OpenMP, CUDA, Pthreads, ...
  - Templated meta-programming: parallel_for, parallel_reduce (templated on an *execution space*).
  - Memory layout abstraction ("array of structs" vs. "struct of arrays", locality).

With *Kokkos*, you write an algorithm **once**, and just change a template parameter to get the optimal data layout for your hardware (e.g., (i,j,k) vs. (k,i,j)).

- **Finite element assembly** in *Albany* has recently been rewritten using *Kokkos* functors.

*X = OpenMP, CUDA, etc.*
Performance-Portability via Kokkos

We need to be able to run Albany/FELIX on new architecture machines (hybrid systems) and manycore devices (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.).

| MPI (task parallelism) + X* (thread + data-level parallelism) |

- **Kokkos**: open-source library that provides performance portability across diverse devices with different memory models.
  - A *programming model* as much as a software library.
  - Provides automatic access to OpenMP, CUDA, Pthreads, ...
  - Templated meta-programming: parallel_for, parallel_reduce (templated on an execution space).
  - Memory layout abstraction (“array of structs” vs. “struct of arrays”, locality).

With Kokkos, you write an algorithm once, and just change a template parameter to get the optimal data layout for your hardware (e.g., (i,j,k) vs. (k,i,j)).

- **Finite element assembly** in Albany has recently been rewritten using Kokkos functors.
- Linear solvers in Belos package of Trilinos can run on next-generation platforms with simple preconditioners (Jacobi, Gauss-Seidel, Chebyshev, ILU).

*X = OpenMP, CUDA, etc.*
Kokkos-ification of Finite Element Assembly

typedef Kokkos::OpenMP ExecutionSpace;
//typedef Kokkos::CUDA ExecutionSpace;
//typedef Kokkos::Serial ExecutionSpace;

template< typename ScalarT>
vectorGrad< ScalarT >::vectorGrad()
{
    Kokkos::View< ScalarT *, ExecutionSpace > vecGrad("vecGrad", numCells, numQP, numVec, numDim);
}

ExecutionSpace parameter tailors code for device (e.g., OpenMP, CUDA, etc.)
Results on *Shannon*: 4km Greenland & 8km Antarctica Problems

**Shannon**: 32 nodes
- 2 8-core Sandy Bridge Xeon E5-2670 @ 2.6GHz (HT deactivated)/node.
- 128GB DDR3 memory/node
- 2x NVIDIA K20x/node.

- Serial: 1 MPI thread/node
- OpenMP: 16 OpenMP threads/node
- CUDA: 1 GPU/node

Max speedup over Serial for workset size > 1000

<table>
<thead>
<tr>
<th></th>
<th>OpenMP</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time</td>
<td>5.6x</td>
<td>1.7x</td>
</tr>
<tr>
<td>Compute Time</td>
<td>7.2x</td>
<td>6.7x</td>
</tr>
</tbody>
</table>

"# of elements/workset" = threading index (allows for on-node parallelism)
Results on *Titan*: Weak Scalability for Greenland (8km, 4km, 2km, 1km)

- **CUDA** implementation is *slower* than MPI-only for total time but *faster* for compute time (due to communication costs).
- Filling the matrix requires use of *atomics*, which are difficult to optimize (work in progress by Kokkos team).
- Work is *distributed more thinly* on Titan than on Shannon → worse for GPU (less work, more communications).

**Titan**: 18,688 AMD Opteron nodes
- 16 cores per node
- 1 K20X Kepler GPUs per node
- 32GB + 6GB memory per node
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.

Validation*: how well does our model represent the real ice sheet?

*vs. verification: is our code bug free? (some of my past SIAM talks on PISCEES)
Validation: Definition & Workflow

**Validation**: how well does our model represent the real ice sheet?

- There are currently (up to) 2 decades of large-scale *satellite* observations of Greenland ice sheet geometry change:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ICESat1</td>
<td>2003 – 2009</td>
</tr>
<tr>
<td>GRACE</td>
<td>2002 – 201? (ongoing)</td>
</tr>
</tbody>
</table>

*vs. verification: is our code bug free? (some of my past SIAM talks on PISCEES)*
Validation*: how well does our model represent the real ice sheet?

- There are currently (up to) 2 decades of large-scale satellite observations of Greenland ice sheet geometry change:
  - ICESat1: 2003 – 2009
  - GRACE: 2002 – 201? (ongoing)

- Future missions will extend these observational time series:
  - ICESat1: 2017-20??
  - GRACE “follow-on”: 2017-20??
  - GRACE2: 2020s-?

*vs. verification: is our code bug free? (some of my past SIAM talks on PISCEES)
Validation: Definition & Workflow

**Validation**: how well does our model represent the real ice sheet?

- There are currently (up to) 2 decades of large-scale *satellite* observations of Greenland ice sheet geometry change:
  - ICESat1: 2003 – 2009
  - GRACE: 2002 – 201? (ongoing)

- Future missions will *extend* these observational time series:
  - ICESat1: 2017-20??
  - GRACE “follow-on”: 2017-20??
  - GRACE2: 2020s-?

**Validation Workflow:**
- Run ice sheet model over period where ICESat/GRACE observations exist.
- Process model output for comparison to these observations.
- Process observations for comparison to model output.
- Evaluate model performance relative to observations.
  - *ICESat*: ice sheet surface elevation [state comparison]
  - *GRACE*: rate of mass change [trend comparison]

*vs. verification: is our code bug free? (some of my past SIAM talks on PISCEES)*
Validation: Observations & Forcing

- **Validation time period:**
  - ICESat: 2003-2009
  - GRACE: 2003-2011 (CSR Release-05)

- **Model forcing:** the following datasets taken as “truth”
  - Monthly *surface mass balance* (SMB) from RACMO2\(^1\) anomalies applied (1960-present)
    - Well-validated over Greenland
  - Mean-annual *outlet glacier flux*\(^2\) applied at grounding line (1990-present)
    - Figure below: outlet glacier flux forcing (FF) time series

---

\(^{1}\) van Angelen et al. (*Surv. Geophys.*, 2013)  \(^{2}\) Enderlin et al. (*GRL*, 2014)
Validation: 1km GIS initial condition*

*Initial condition obtained through deterministic inversion; see talk by M. Perego.
Validation: Surface Elevation Observations

- **Model evaluated:** *CISM-Albany* with *SMB + FF* in October 2007.
- Surface elevation predictions (states) agree pretty well with *GLAS (Geoscience Laser Altimeter System aboard ICESat)*: *mean differences* are <1 m
Validation: Whole Ice Sheet Mass Trends

- Overall mass trends from the **CISM-Albany** simulations look **fairly realistic** (left figure, red & blue).
- **CISM-Albany** trends look **more realistic** (closer to observations; right figure) than the "idealized" simulations (left figure, pink & black).
- There is **more mass loss** from the simulation when we account for changes in outlet glacier flux (i.e., the evolution of the ice sheet is not only forced by surface mass balance changes, but also be changes in outlet glacier dynamics).

---

1. Geometry held constant in time.
2. RACMO2 SMB time series applied to \( \frac{dh}{dt} \); includes SS “discharge” using 1960-1990 mean SMB.
Validation: Whole Ice Sheet Mass Trends

- *Apples-to-apples* comparison between models and GRACE:
  - Information is all on the *same plot*.
  - Model output and observations were both *processed in the same way*. 
Validation Takeaways

Current generation ice sheet models, when appropriately forced, show skill at mimicking ice sheet observations.
Validation Takeaways

Current generation ice sheet models, *when appropriately forced*, show *skill* at mimicking ice sheet observations

- **Clear improvement over a decade ago**: SLR projections from ice sheet models were not included in the IPCC’s AR4 b/c models could not explain observed ice dynamical behaviors.
Validation Takeaways

Current generation ice sheet models, **when appropriately forced**, show **skill** at mimicking ice sheet observations

- **Clear improvement over a decade ago:** SLR projections from ice sheet models were not included in the IPCC’s AR4 b/c models could not explain observed ice dynamical behaviors.
- For more details on validation study, see our **GMD paper**, currently under review.
Cryospheric Model Comparison Tool (CmCt)

- Validation work involved development of new ice sheet model validation framework – the Cryospheric Model Comparison Tool (CmCt) – which includes proposed qualitative and quantitative metrics for use in comparing models to observations.

CmCt is **online** and ready for testing by community

https://ggsghpcc.sgt-inc.com/cmct/
Outline

1. Ice Sheet Equations and Codes.

2. Mesh Adaptivity.


5. Validation.


Steps towards creating a production-ready code
Summary:

- We have developed a land-ice solver known as **Albany/FELIX** using **Trilinos** libraries.
- This solver is:
  - Equipped with in-memory parallel **unstructured mesh adaption**.
  - **Scalable, fast, robust**.
  - Coupled to *CISM* and *MPAS* codes for **dynamic** runs and integration into ESMs.
  - **Verified** and **validated**.
  - **Portable** to new and emerging architecture machines.
  - Equipped with **advanced analysis capabilities** (deterministic inversion, UQ).

Ongoing/future work:

- **Mesh adaptivity** for **transient** runs.
- Integration of **hinge removal algorithm** into **transient** runs.
- **Science runs** using *CISM-Albany* and *MPAS-Albany*.
- Code optimizations for **new architecture machines** (GPUs, multi-core, many-core).
- **Uncertainty quantification** (not covered in this talk).
- Delivering code to climate community and **coupling to ESMs**.
Thank you! Questions?

Video acknowledgement: B. Carvey (SNL)
Funding/Acknowledgements

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) projects funded by the U.S. Department of Energy, Office of Science (OSCR), Advanced Scientific Computing Research and Biological and Environmental Research (BER) → PISCEES SciDAC Application Partnership.


**Trilinos/DAKOTA collaborators:** M. Eldred, J. Jakeman, E. Phipps, L. Swiler.

**Computing resources:** NERSC, OLCF.
References


References (cont’d)


**PISCEES Project for Land-Ice Modeling**

**PISCEES**
*SciDaC Application Partnership (DOE’s BER + ASCR divisions) 2012-2017*

---

**Goal:** support DOE climate missions (sea-level rise predictions)

---

**Multi-lab/multi-university** project involving mathematicians, climate scientists, and computer scientists.

- Leverages software/expertise from **SciDAC Institutes** (FASTMath, QUEST, SUPER) and hardware from **DOE Leadership Class Facilities**.
- **Sandia staff:** Irina Tezaur, Andy Salinger, Mauro Perego, Ray Tuminaro, John Jakeman, Mike Eldred; **PI:** Steve Price (LANL)

---

**FSU FELIX**
*FSU*
Finite Element Full Stokes Model

**Albany/FELIX**
*SNL*
Finite Element “Higher-Order” Stokes Model

**BISICLES**
*LBNL*
Finite Volume L1L2 Model
Unstructured Mesh Generation & Adaptivity

Original approach:

1. **Step 1**: geometry boundary and possible holes determined in MATLAB
2. **Step 2**: uniform triangular mesh is generated and refined based on *surface velocity gradient* in Triangle (2D meshing software)
3. **Step 3**: 3D mesh obtained by extruding 2D mesh vertically as *prisms*, then splitting each prism into 3 *tetrahedra (Albany)*.

New approach: Parallel Albany Adaptive Loop with SCOREC* (PAALS)

1. **Step 1**: geometry determined from *.*.nc file using *scripts*.
2. **Step 2**: uniform tetrahedral mesh generated (using Triangle or Simmetrix).
3. **Step 3**: 2D slice of initial mesh adaptively refined via *in-memory Parallel Albany Adaptive Loop with SCOREC (PAALS)*.
4. **Step 4**: 3D mesh obtained by extruding 2D mesh vertically as *prisms* in Albany.

*Scientific Computation Research Center at Rensselaer Polytechnic Institute (RPI); https://github.com/SCOREC*
Fine-Resolution Greenland Strong Scaling Study

- Strong scaling on 1km Greenland with 40 vertical layers (143M dofs, hex elements).
- Initialized with realistic basal friction (from deterministic inversion) and temperature fields → interpolated from coarser to fine mesh.
- **Iterative linear solver**: CG.
- **Preconditioner**: ILU vs. new AMG (based on aggressive semi-coarsening).

ILU solver scales better than AMG but ILU solve is slightly slower: AMG solver becomes inefficient when # unknowns/core small (expensive setup; a lot of communications).
Validation - Total Mass Change
2003-2012

GRACE

RACMO2

SMB

SMB+FF