Albany: a Trilinos-based multi-physics partial differential equation research tool created using the AgileComponents code development strategy

Irina Tezaur

Quantitative Modeling & Analysis Dept., Sandia National Laboratories, Livermore, CA.

SIAM CSE 2019 Spokane, WA Feb. 25 - Mar. 1 2019
Acknowledgements

“Father” of Albany, early advocate for AgileComponents strategy:
• Andy Salinger [SNL]

Albany contributors (58, from github):

• I. Tezaur [SNL]
• A. Mota [SNL]
• D. Ibanez [RPI/SNL]
• G. Hansen [SNL]
• M. Perego [FSU/SNL]
• L. Bertagna [FSU/SNL]
• A. Bradley [SNL]
• J. Ostien [SNL]
• B. Granzow [RPI/SNL]
• A. Salinger [SNL]
• D. Littlewood [SNL]
• J. Foulk [SNL]
• C. Alleman [SNL]
• J. Overfelt [SNL]
• E. Nielsen [SNL]
• I. Demeshko [LANL]
• O. Guba [SNL]
• M. Juha [U de La Sabana]
• E. Phipps [SNL]
• J. Frederick [SNL]
• J. Watkins [SNL]
• J. Robbins [SNL]
• T. Smith [SNL]
• R. Jones [SNL]
• B. Spotz [SNL]
• S. Besu [RPI]
• J. Foucar [SNL]
• Q. Chen [Clemson]
• P. Lindsay [SNL]
• J. Fike [SNL]
• S. Gao [SNL]
• J. Redhorse [SNL]
• S. Bova [SNL]
• P. Bosler [SNL]
• H. Yuan [Tokyo]
• C. Siefert [SNL]
• T. Voth [SNL]
• T. Wildey [SNL]
• C. Smith [RPI]
• J. Clough [USC]
• W. Sun [Columbia]
• Z. Wang [USC]
• J. Robbins [SNL]
• T. Fuller [SNL]
• A. Vacanti [Kitware]
• R. Tuminaro [SNL]
• M. Parks [SNL]
• J. Robbins [SNL]
• M. Hoffman [LANL]
• R. Pawlowski [SNL]
• M. Bloomfield [RPI]
• G. Phlipot [CalTech]
• B. Perschbacher [SNL]
• J. Willenbring [SNL]
• ...
Outline

1. AgileComponents code-development strategy
2. What is Albany?
3. Albany code design
   - Global discretization & libraries
   - Problem abstraction & finite element assembly
   - Nonlinear model abstraction & libraries
   - Linear model abstraction & libraries
   - Software quality tools
4. Applications hosted by Albany
5. Algorithmic projects hosted by Albany
6. Coupling with other codes
7. Summary
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the rapid development of new production codes embedded with transformational capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software **components**
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the **rapid** development of new **production** codes embedded with **transformational** capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software **components**
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

<table>
<thead>
<tr>
<th>“Components”</th>
<th>Libraries</th>
<th>Software Quality Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interfaces</td>
<td>Demonstration Applications</td>
</tr>
</tbody>
</table>
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the rapid development of new production codes embedded with transformational capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software components
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

  
  
  “Components” = ✓ Libraries ✓ Software Quality Tools
  ✓ Interfaces ✓ Demonstration Applications

- **Business strategy:**
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the rapid development of new production codes embedded with transformational capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software components
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

<table>
<thead>
<tr>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libraries</td>
</tr>
<tr>
<td>Software Quality Tools</td>
</tr>
<tr>
<td>Interfaces</td>
</tr>
<tr>
<td>Demonstration Applications</td>
</tr>
</tbody>
</table>

- **Business strategy:**

Base of Software Components
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the **rapid** development of new production codes embedded with **transformational** capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software **components**
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

<table>
<thead>
<tr>
<th>Components</th>
<th>Libraries</th>
<th>Software Quality Tools</th>
<th>Interfaces</th>
<th>Demonstration Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

- **Business strategy:**

![Base of Software Components](Image1)

![Projects](Image2)
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the rapid development of new production codes embedded with transformational capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software *components*
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

“Components” = ✓ Libraries ✓ Software Quality Tools ✓ Interfaces ✓ Demonstration Applications

- **Business strategy:**

Leverage the base →  ECP

Base of Software Components

Projects
AgileComponents: a PDE code strategy

**Strategic Goal:** To enable the **rapid** development of new **production** codes embedded with **transformational** capabilities.

- **Technical strategy:** projects create, use, and improve a common base of modular, independent-yet-interoperable, software **components**
  - 2012 white paper by A. Salinger: “Component-Based Scientific Application Development” (right).

  ```markdown
  “Components” = ☑ Libraries ☑ Software Quality Tools
  ☑ Interfaces ☑ Demonstration Applications
  ```

- **Business strategy:**

---

Leverage the base → Projects

Base of Software Components

← Grow the base

---
What is Albany? (high-level description)

<table>
<thead>
<tr>
<th>“Components”</th>
<th>Libraries</th>
<th>Software Quality Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>✔️</td>
<td>✔️ ❯ <em>Demonstration Applications</em></td>
</tr>
</tbody>
</table>

2008

Albany
What is Albany? (high-level description)

**Albany**: open-source*, parallel, C++, unstructured-grid, mostly-implicit multi-physics finite element code that demonstrates *AgileComponents* vision.

Albany houses a variety of diverse algorithmic projects and applications:

- Ice sheets
- Quantum devices
- Computational mechanics
- Additive manufacturing
- Arctic costal erosion

* Albany github repo: https://github.com/SNLComputation/Albany.
What is Albany? (high-level description)

**Distinguishing features of Albany:**

- **Funded** entirely by applications residing within.
- Both a “sand-box” for prototyping **new approaches** and a **production** code.
- **Algorithms/software** are developed/matured directly on applications.
- Applications are “born” scalable, fast, robust, and...
- Equipped with **embedded advanced analysis capabilities**: sensitivities, bifurcation analysis, adjoint-based inversion, *embedded UQ*, *model reduction*.

*Italicized capabilities are in feature branches/tags.*
The components effort: libraries & tools

*Components in Albany* = cutting-edge technology from Trilinos, SCOREC, SierraToolKit, DAKOTA, FASTMath, QUEST, Kitware, etc.

Many components are Trilinos* packages:

- Distributed linear algebra (*Tpetra*)
- Mesh tools (*STK*)
- Discretization tools (*Intrepid2*)
- Nonlinear solver (*NOX*)
- Linear solver (*Belos*)
- Preconditioners (*Ifpack2*)
- Automatic differentiation (*Sacado*)
- Shared memory parallelism (*Kokkos*)
- Optimization (*ROL*)
- Many more...

*: 40+ packages; 120+ libraries

*Trilinos github repo: https://github.com/trilinos/Trilinos.
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”

Legend:
- Libraries
- Interfaces
- Albany Code

- **Analysis Tools**
  - Optimization
  - UQ

- **Application**
  - Solvers
    - Nonlinear
    - Transient

- **Solvers**
  - Linear Solve
    - Linear Solvers
      - Iterative
      - Direct

- **Main**
  - Input Parser

- **Nonlinear Model**

- **Global Discretization**

- **Mesh Processing**

- **Evaluation Engine**
  - PDE terms, BCs, responses

- **Problem Abstraction**
  - FEM Assembly
    - Field Manager
    - Derivatives
    - Discretization

- **Mesh Tools**
  - Mesh Database
  - Mesh I/O
  - Load Balancing

- **“Glue Code”** connects together interfaces or instances of components
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”
What is Albany? (under-the-hood)

Albany = Component Libraries + Abstract Interfaces + “Glue Code”

1. Global Discretization

Legend:
- Libraries
- Interfaces
- Albany Code
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”

- **Analysis Tools**
  - Optimization
  - UQ

- **Application**

- **Solvers**
  - Nonlinear
  - Transient

- **Linear Solve**
  - Linear Solvers
  - Iterative
  - Direct

- **Main**
  - Input Parser

- **Nonlinear Model**

- **Albany Glue Code**

- **Evaluation Engine**

- **Problem Abstraction**
  - PDE terms, BCs, responses
  - FEM Assembly
    - Field Manager
    - Derivatives
    - Discretization

- **Mesh Tools**
  - Mesh Database
    - Mesh I/O
    - Load Balancing

- **Legend:**
  - Libraries
  - Interfaces
  - Albany Code
**What is Albany? (under-the-hood)**

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”
What is Albany? (under-the-hood)

Albany = Component Libraries + Abstract Interfaces + “Glue Code”

Legend:
- Libraries
- Interfaces
- Albany Code

1. Global Discretization

serves as general interface to mesh tools/services
Global discretization abstraction & libraries

**Discretization interface:** currently has *two independent implementations*

1. **SierraToolKit (STK)** package in Trilinos.
   - Supports reading in *Exodus* mesh files (e.g., from CUBIT), *inline meshing* via Pamgen, *simple rectangular meshes* constructed in Albany.
   - Meshes can be *structured/unstructured* but are *static*.

2. **Parallel Unstructured Mesh Infrastructure (PUMI)** package, developed at the Scientific Computation Research Center (SCOREC) at RPI.
   - Supports *VTK mesh files* (generated by Symmetrix).
   - Goal-oriented generalized *error estimation* and *in-memory mesh adaptation*.

**Element types**: variety of element types supported, with basis functions/quadrature routines from *Intrepid2* Trilinos library:

- **Isoparametric** elements (tet, hex, wedge, ...).
- 2D **spectral elements** of arbitrary orders.
- Some physics-specific elements, e.g., **composite 10-node tetrahedron** for solid mechanics.

**Discretization interface**: currently has *two independent implementations*

1. **SierraToolKit (STK)** package in Trilinos.
   - Supports reading in *Exodus* mesh files (e.g., from CUBIT), inline meshing via Pamgen, **simple rectangular meshes** constructed in Albany.
   - Meshes can be **structured/unstructured** but are **static**.

2. **Parallel Unstructured Mesh Infrastructure (PUMI)** package, developed at the Scientific Computation Research Center (SCOREC) at RPI.
   - Supports *VTK mesh files* (generated by Symmetrix).
   - Goal-oriented generalized **error estimation** and **in-memory mesh adaptation**.

Global discretization abstraction & libraries

Discretization interface: currently has two independent implementations

1. SierraToolKit (STK) package in Trilinos.
   - Supports reading in Exodus mesh files (e.g., from CUBIT), inline meshing via Pamgen, simple rectangular meshes constructed in Albany.
   - Meshes can be structured/unstructured but are static.

2. Parallel Unstructured Mesh Infrastructure (PUMI) package, developed at the Scientific Computation Research Center (SCOREC) at RPI.
   - Supports VTK mesh files (generated by Symmetrix).
   - Goal-oriented generalized error estimation and in-memory mesh adaptation.

Albany is a Continuous Galerkin (CG) unstructured grid finite element code.

Element types: variety of element types supported, with basis functions/quadrature routines from Intrepid2 Trilinos library:

- Isoparametric elements (tet, hex, wedge, ...).
- 2D spectral elements of arbitrary orders.
- Some physics-specific elements, e.g., composite 10-node tetrahedron for solid mechanics.
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”

Purpose of problem abstraction & FEA: given a mesh, evaluate discrete FE residual and Jacobian → involves three key ingredients...
1. Templated-based automatic differentiation

**Automatic differentiation (AD)** provides exact derivatives without time/effort of deriving and hand-coding them.

- Template equation implementation on scalar type.
- Libraries (Sacado) provides new scalar types that **overload the math operators** to propagate embedded quantities via chain rule.
  - **Derivatives**: DFad<double>
  - **Hessians**: DFad<SFad<double,N>>
  - **Stochastic Galerkin resid**: PCE<double>
  - **Stochastic Galerkin Jac**: DFad<PCE<double>
  - **Sensitivities**: DFad<double>

**No finite difference truncation error!**

- Great for **multi-physics codes** (e.g., many Jacobians) and **advanced analysis** (e.g., sensitivities, optimization)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Overloaded AD impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = a \pm b$</td>
<td>$\dot{c} = \dot{a} \pm \dot{b}$</td>
</tr>
<tr>
<td>$c = ab$</td>
<td>$\dot{c} = a\dot{b} + \dot{a}b$</td>
</tr>
<tr>
<td>$c = a/b$</td>
<td>$\dot{c} = (\dot{a} - \dot{b}c)/b$</td>
</tr>
<tr>
<td>$c = a^r$</td>
<td>$\dot{c} = ra^{r-1}\dot{a}$</td>
</tr>
<tr>
<td>$c = \sin(a)$</td>
<td>$\dot{c} = \cos(a)\dot{a}$</td>
</tr>
<tr>
<td>$c = \cos(a)$</td>
<td>$\dot{c} = -\sin(a)\dot{a}$</td>
</tr>
<tr>
<td>$c = \exp(a)$</td>
<td>$\dot{c} = c\dot{a}$</td>
</tr>
<tr>
<td>$c = \log(a)$</td>
<td>$\dot{c} = \dot{a}/a$</td>
</tr>
</tbody>
</table>

---

2. Template-based generic programming (TBGP)

**Albany Finite Element Assembly (FEA):**

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

→ Global, linear algebra storage
→ Local, element storage
2. Template-based generic programming (TBGP)

Albany Finite Element Assembly (FEA):

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

Evaluators templated on evaluation type:

<EvalT> (Generic)

→ Global, linear algebra storage
→ Local, element storage
2. Template-based generic programming (TBGP)

Albany Finite Element Assembly (FEA):

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

- Evaluators templated on evaluation type:
  - `<EvalT>` (Generic)
  - Template Specializations:
    - Residual

- Global, linear algebra storage
- Local, element storage
2. Template-based generic programming (TBGP)

Albany Finite Element Assembly (FEA):

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

Evaluators templated on evaluation type:
\[ <\text{EvalT}> \quad \text{(Generic)} \]

Template Specializations:
- Residual
- Jacobian

→ Global, linear algebra storage
→ Local, element storage
2. Template-based generic programming (TBGP)

**Albany Finite Element Assembly (FEA):**

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

**Evaluator Specializations:**
- Residual
- Jacobian
- Tangent
- Hessian
- Adjoint
- PCE
- Shape Opt

→ Global, linear algebra storage
→ Local, element storage
2. Template-based generic programming (TBGP)

**Albany Finite Element Assembly (FEA):**

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

Enables **advanced analyses** (sensitivities, optimization, ...)

---

2. Template-based generic programming (TBGP)

**Albany Finite Element Assembly (FEA):**

- **Gather Solution** extracts values from global structures, puts in element local structures
- **Evaluators** operate on element local data structures
- **Scatter** adds local contributions to global structures

**Evaluators templated on evaluation type:**

- `<EvalT>` (Generic)

**Template Specializations:**

- Residual
- Jacobian
- Tangent
- Hessian
- Adjoint
- PCE
- Shape Opt

→ Global, linear algebra storage
→ Local, element storage

Employs **AD overloads** (Sacado, Stokhos)

Enables **advanced analyses** (sensitivities, optimization, ...)

3. Graph-based finite element assembly (FEA)

**Assembly** of physics pieces comes down to the evaluation of a **directed acyclic graph (DAG)** of computations of field data.

**Phalanx package:** Local field evaluation kernel designed for assembly of arbitrary equation sets (i.e. evaluating residuals/Jacobians).

- **Decomposes** a complex model into a graph of **simple kernels** (functors)
- A node in the graph evaluates one or more **temporary fields**
- **Runtime** DAG construction of graph
- Achieves **flexible multi-physics assembly**

\[
R_u^i = \int_\Omega \left[ \phi_u^i \dot{u} - \nabla \phi_u^i \cdot q + \phi_u^i s \right] \, d\Omega
\]
3. Graph-based finite element assembly (FEA)

**Assembly** of physics pieces comes down to the evaluation of a **directed acyclic graph (DAG)** of computations of field data.

**Phalanx package**: Local field evaluation kernel designed for assembly of arbitrary equation sets (i.e. evaluating residuals/Jacobians).

- ** Decomposes** a complex model into a graph of **simple kernels** (functors)
- A node in the graph evaluates one or more **temporary fields**
- **Runtime** DAG construction of graph
- Achieves **flexible multi-physics assembly**

DAG-based assembly enables **flexibility, extensibility, rapid development**: to add new PDE, all you need to code is problem-specific residual $R^i_u$!

\[
R^i_u = \int_\Omega \left[ \phi^i_u \dot{u} - \nabla \phi^i_u \cdot q + \phi^i_u s \right] \, d\Omega
\]
What is Albany? (under-the-hood)

**Albany** = Component Libraries + Abstract Interfaces + “Glue Code”

### Analysis Tools
- Optimization
- UQ
- Application

### Solvers
- Nonlinear
- Transient
- Linear Solve
- Linear Solvers
- Iterative
- Direct

### Libraries
- Nonlinear Model
- PDE terms, BCs, responses
- FEM Assembly
- Field Manager
- Derivatives
- Discretization

### Interfaces
- Global Discretization
- Mesh Database
- Mesh I/O
- Load Balancing

### Code
- Main
- Input Parser
- Albany Glue Code
- Mesh Processing
- Evaluation Engine
- Problem Abstraction

**Legend:**
- Libraries
- Interfaces
- Albany Code

---

3. **Nonlinear Model**

*physics pieces are required to satisfy nonlinear model abstraction*
Nonlinear model abstraction & libraries

**“ModelEvaluator”:**

<table>
<thead>
<tr>
<th>Given:</th>
<th>Computes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$f(\dot{x}, \ddot{x}, x, p, t)$</td>
</tr>
<tr>
<td>$\dot{x}$</td>
<td>$W = \alpha \frac{df}{dx} + \beta \frac{df}{dx} + \omega \frac{df}{dx}, W_{prec}$</td>
</tr>
<tr>
<td>$\ddot{x}$</td>
<td>$\frac{df}{dp}$</td>
</tr>
<tr>
<td>$p$</td>
<td>$g$</td>
</tr>
<tr>
<td>$t$</td>
<td>$\frac{dg}{dx}, \frac{dg}{dp}$</td>
</tr>
</tbody>
</table>

$f = \text{residual}; x = \text{solution}; p = \text{params}; g = \text{responses}; t = \text{time}; W = \text{Jacobian}; W_{prec} = \text{Preconditioner}$

- Interface is **general** to accommodate computation of Jacobians, user-defined preconditioners, and stochastic Galerkin expansions.

- Enables **“beyond-forward analysis”**: analysts/physics experts are not burdened with analysis algorithm requirements, i.e., programming sensitivities for implicit solvers, optimization, stability, bifurcation analysis.

  ➢ **Advanced capabilities**: optimization (ROL), homotopy continuation (LOCA), embedded UQ (Stokhos).

Access to Trilinos **embedded solvers** requires satisfaction of **ModelEvaluator** (nonlinear model) abstraction.
What is Albany? (under-the-hood)

Albany = Component Libraries + Abstract Interfaces + “Glue Code”

Legend:
- Libraries
- Interfaces
- Albany Code

**Analysis Tools**
- Optimization
- UQ

**Application**
- Solvers
  - Nonlinear
  - Transient

**Solvers**
- Nonlinear
- Transient

**Main**
- Input Parser

**Global Discretization**

**Mesh Tools**
- Mesh Database
- Mesh I/O
- Load Balancing

**Mesh Processing**

**Problem Abstraction**

**Evaluation Engine**
- PDE terms, BCs, responses

**Albany Glue Code**

**Nonlinear Model**

**Mesh Tools**
- Mesh Database

**Load Balancing**

**Libraries**
- Interface

**Main**
- Input Parser

**Nonlinear Solver Invokes Linear Solver When Needed**

**Legend:**
- Libraries
- Interfaces
- Albany Code
Linear solver abstraction & libraries

- **Linear solver abstraction** provides full access to all Trilinos linear solvers (direct and iterative), eigensolvers and preconditioners through Stratimikos interface.

- **Factory class** supports run-time solution configuration through input file options.

- **Available direct solvers**: Amesos, Amesos2 (UMFPACK, MUMPS, SuperLU, SCALAPACK, etc.).

- **Available iterative solvers**: AztecOO, Belos (CG, GMRES)

- **Available preconditioners**: Ifpack, Ifpack2 (ILU); ML, MueLu (AMG); Teko (block)

- **Eigensolvers**: Anasazi

---

**Linear Solver**

Given:
- Matrix operator \( A \)
- RHS vector \( b \)
- Matrix entries (optional)
- Parameter/options list ➢ Tolerance

Compute:
- Solution \( x \), with \( Ax = b \)
- Eigenvalues/vectors

---

**Stratimikos**

Linear Solver and Preconditioner Wrappers

- Belos
- Amesos2
- Ifpack2
- Teko
Software quality tools & processes

- Repository*
- Version control
- Build system
- Config mgmt
- Regression tests

Nightly test harness
- Unit tests
- Verification tests
- Code coverage
- Performance tests

Mailing lists
- Issue tracking
- Web pages
- Licensing
- Release process

Performance monitored via **CDash nightly testing** on a variety of architecture including GPU (P100, V100), Xeon Phi, Skylake, ARM platforms.

* Albany github repo: [https://github.com/SNLComputation/Albany](https://github.com/SNLComputation/Albany)
Software quality tools & processes

- Repository*
- Version control
- Build system
- Config mgmt
- Regression tests
- Nightly test harness
- Unit tests
- Verification tests
- Code coverage
- Performance tests
- Mailing lists
- Issue tracking
- Web pages
- Licensing
- Release process

Performance monitored via CDash nightly testing on a variety of architecture including GPU (P100, V100), Xeon Phi, Skylake, ARM platforms.

* Albany github repo: https://github.com/SNLComputation/Albany.
Applications hosted by Albany

- Quantum Devices (QCAD)
- Ice Sheets (Albany Land-Ice)
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran).
Applications hosted by Albany

- Quantum Devices (QCAD)
- Ice Sheets (Albany Land-Ice)
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran).

Applications are “born” scalable, fast, robust, and equipped with advanced analysis capabilities!
Applications hosted by Albany

- Quantum Devices (QCAD)
- Ice Sheets (Albany Land-Ice)
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran).

Applications are “born” scalable, fast, robust, and equipped with advanced analysis capabilities!
Ice sheets: Albany Land-Ice (ALI)

Albany enabled the **rapid development** of a production **land-ice dycore** for providing **actionable predictions** of 21st century sea-level rise as a part of the DOE Energy Exascale Earth System Model (E3SM).

**Capabilities:**

- **Unstructured grid** finite elements.
- **Scalable, fast** and **robust**
- **Verified** and **validated**
- **Advanced analysis:** inversion, UQ
- **Portable** to GPU, KNL, ... via Kokkos
- **Multi-physics:** velocity-temperature, velocity-thickness, velocity-hydrology

**Above:** Thwaites glacier retreat under parametrized submarine melting

Laboratory for computational mechanics (LCM)

The Albany LCM suite contains sophisticated **material models, physics** and **technologies** for solid mechanics.

- **“Sand-box”** for new algorithms/methods:
  - Composite 10-node tetrahedron
  - Pressure projection stabilization
  - Multi-scale coupling via Schwarz
  - In-memory mesh adaptation

- **Models**: elasticity, Neohookean, J₂ plasticity, crystal plasticity, elasto-visco-plastic, ...

- **Physics (PDEs)**: elasticity, mechanics, electro-mechanics, thermo-mechanics, thermo-poro-mechanics, ...

- **Fracture** and **damage** simulation capabilities
Arctic Coastal Erosion (ACE)

Mechanistic modeling within Albany is advancing state-of-the-art coastal erosion/permafrost modeling.

Permafrost modeling background:
- Predominant geomorphology is ice-wedge polygons (right).
- State-of-the-art erosion modeling: trend projection, empirical relationships, 1D steady-state heat flow, ...

Albany modeling of degradation:
- Leverages years of LCM R&D.
- Time-varying input variables over the duration of a storm (water level, temperature, salinity)
- Multi-physics FEM model of coastline: finite deformation plasticity model + 3D unsteady thermal flow + chemical characteristics
  - Failure modes develop from constitutive relationships (no empirical model!)
Algorithmic projects hosted by Albany

*Algorithms and software are matured directly on applications.*

Algorithmic projects within Albany:

- Scalable multi-level solvers (PISCEES/ProSPect) – R. Tuminaro, I. Tezaur.
- Nonlinear solvers (FASTMath) – R. Pawlowski, M. Perego
- In-memory mesh adaptation (FASTMath) – M. Sheppard, M. Bloomfield (RPI), A. Oberai, J. Smith (USC), D. Ibanez, B. Granzow, G. Hansen
- Multi-scale coupling via Schwarz (P&EM) – A. Mota, I. Tezaur, C. Alleman, G. Phlipot (CalTech)
- Stabilized mechanics (FASTMath) – A. Bradley, J. Ostien, G. Hansen
- Adjoint-based inversion (FASTMath) – M. Perego, E. Phipps, ROL team
- Optimization-based coupling (ASCR) – M. Perego, M. D’Elia, D. Littlewood, P. Bochev
- UQ workflow (PISCEES) – J. Jakeman, I. Tezaur, M. Perego
- Embedded UQ (Equinox) – E. Phipps, J. Fike
- Performance portable FEM (PISCEES/ProSPect/ATDM) – I. Demeshko (LANL), E. Phipps, R. Pawlowski, E. Cyr, I. Tezaur, A. Bradley, J. Watkins
- Development of composite tet-10 for solid mechanics (PE&M) – J. Foulk, J. Ostien, A. Mota
Algorithmic projects hosted by Albany

Algorithms and software are matured directly on applications.

Algorithmic projects within Albany:

- Scalable multi-level solvers (PISCEES/ProSPect) – R. Tuminaro, I. Tezaur.
- Nonlinear solvers (FASTMath) – R. Pawlowski, M. Perego
- **In-memory mesh adaptation** (FASTMath) – M. Sheppard, M. Bloomfield (RPI), A. Oberai, J. Smith (USC), D. Ibanez, B. Granzow, G. Hansen
- **Multi-scale coupling via Schwarz** (P&EM) – A. Mota, I. Tezaur, C. Alleman, G. Phlipot (CalTech)
- Stabilized mechanics (FASTMath) – A. Bradley, J. Ostien, G. Hansen
- **Adjoint-based inversion** (FASTMath) – M. Perego, E. Phipps, ROL team
- Optimization-based coupling (ASCR) – M. Perego, M. D’Elia, D. Littlewood, P. Bochev
- UQ workflow (PISCEES) – J. Jakeman, I. Tezaur, M. Perego
- Embedded UQ (Equinox) – E. Phipps, J. Fike
- Performance portable FEM (PISCEES/ProSPect/ATDM) – I. Demeshko (LANL), E. Phipps, R. Pawlowski, E. Cyr, I. Tezaur, A. Bradley, J. Watkins
- Development of composite tet-10 for solid mechanics (PE&M) – J. Foulk, J. Ostien, A. Mota
Adjoint-based optimization/inversion

AD & TBGP in Albany enabled the efficient solution of adjoint-based PDE-constrained optimization/inversion problems.

\[ \text{Find } p \text{ that minimizes } g(u, p) \]
\[ \text{subject to } f(u, p) = 0 \leftarrow \text{PDE} \]

**Application**: inversion for basal friction and ice thickness in Albany Land-Ice model to initialize dynamic simulation.

- Inversion approach **significantly reduces non-physical transients**.

\( \beta \) (kPa y/m) obtained through inversion

\( |u| \) (m/yr) computed with estimated \( \beta \)

\( |u| \) (m/yr) for observed surface velocity

Multi-scale coupling via Schwarz

A domain decomposition alternating-Schwarz-based method has been developed in Albany for concurrent multi-scale coupling in solid mechanics*.

• **Targeted application**: failure of *bolted components*.

• **“Plug-and-play” framework**: simplifies meshing complex geometries!
  - Couple regions with different non-conformal meshes, element types, levels of refinement, solvers/time-integrators.

---

* See talk by A. Mota: MS 350, Fri. Mar. 1, 10:10-10:30AM, Room 302B
In-memory mesh adaptation

Collaboration with SCOREC*: development of mesh adaptation capabilities in Albany to enable multi-scale/multi-physics adaptive simulation

- Automated parallel goal-oriented adaptive simulation
  - Use adjoint solution to drive mesh adaptation
  - $\sim 100 \times$ DoF-efficiency observed
  - Scaling out to at least 8K MPI ranks
  - Performance portable via Kokkos

Some applications:
- 3D manufacturing (right)
- Creep/plasticity in large solder joint arrays
- Coupled dislocation dynamics

Additive manufacturing simulation showing temperature in evolving geometry

* Scientific Computation Research Center at RPI (Mark Shephard et al.)
Coupling with other codes

Albany has been **interfaced/coupled** with a number of other codes.

- **Albany-Peridigm**: local-nonlocal coupling of continuum mechanics + peridynamics
- **Albany-ParaDis**: coupled dislocation dynamics
- **Albany-PLATO**: Advanced Topology Optimization (ATO)
- **MPAS-Albany-Land-Ice (MALI)**: ice sheets/coupling to E3SM
- **CISM-Albany-Land-Ice (CALI)**: ice sheets/coupling to CESM
- **Albany-PFLOTRAN (Albotran)**: coupled geomechanics problems
**Summary**

*Albany*: open-source, parallel, C++, unstructured-grid, mostly-implicit multiphysics finite element code that demonstrates *AgileComponents* vision and can enable **rapid development** of new physics/algorithms.

**Github**: https://github.com/SNLComputation/Albany


**Albany User Meeting (AUM)**: every ~2 years (TBD)
Albany: open-source, parallel, C++, unstructured-grid, mostly-implicit multi-physics finite element code that demonstrates AgileComponents vision and can enable rapid development of new physics/algorithms.

Github: https://github.com/SNLComputation/Albany


Albany User Meeting (AUM): every ~2 years (TBD)
References


References (cont’d)


Backup Slides
Albany: a component-based finite element code

- Started by A. Salinger in 2008 as **first DemoApp** for AgileComponents code development strategy.

- During **next 10 years**, Albany became:
  - a friendly **early adopter** of cutting-edge technology from Trilinos, SCOREC, SierraToolKit, DAKOTA, FASTMath, QUEST, Kitware.
  - a model for a **Trilinos-based** and **Office of Science** application.
  - a demonstration of **transformational analysis** spanning template-based generic programming to optimization and UQ

- **11 years later**, Albany is:
  - an open-source parallel, mostly-implicit **unstructured-grid multi-physics finite element code** that demonstrates the AgileComponents vision by using, maturing, and spinning-off reusable libraries/abstract interfaces.
  - an **attractive environment** for the development of open-source application codes and research.
  - a Meso-App for maturation of **MPI+X programming model** for next generation architecture
  - the code base underlying a number of **research projects** and **applications**.

2008

2019
Albany “glue code”

“Glue Code”: driver code integrating components + providing overall capabilities
Albany “glue code”

“Glue Code”: driver code integrating components + providing overall capabilities

- Depends on **discretization abstraction** (serves as general interface to a mesh service)
Albany “glue code”

“Glue Code”: driver code integrating components + providing overall capabilities

- Depends on **discretization abstraction** (serves as general interface to a mesh service)
- Employs **evaluation engine** to construct PDEs, BCs, and response calculations
Albany “glue code”

“Glue Code”: driver code integrating components + providing overall capabilities

- Depends on **discretization abstraction** (serves as general interface to a mesh service)
- Employs **evaluation engine** to construct PDEs, BCs, and response calculations
- Uses physics pieces to satisfy **nonlinear model abstraction** (e.g., compute resid/Jac)
Global discretization abstraction

- **Mesh framework**: defines geometry, element topologies, connectivities, boundary info, mesh-dependent fields.
- **Global discretization abstraction**: gives the finite element assembly process access to all of the data distribution information required by the linear algebra objects.
- Mesh info is contained in in-memory mesh database accessed through abstract global discretization interface class.
Problem abstraction & finite element assembly

**Purpose of problem abstraction & finite element assembly:** given a finite element mesh, evaluate discrete finite element residual, Jacobian, and (if applicable) parameter derivatives.

3 key ingredients facilitating multi-physics implementations in Albany:

1. Template-based generic programming (TBGP)
   - Handled by Phalanx package
2. Graph-based finite element assembly (FEA)
   - Handled by Phalanx package
3. Templated-based automatic differentiation
   - Handled by Sacado package
Automatic differentiation via Sacado

Automatic Differentiation (AD) provides exact derivatives w/o time/effort of deriving and hand-coding them!

- How does AD work? → freshman calculus!
  - Computations are composition of simple operations (+, *, sin(), etc.)
  - Derivatives computed line by line then combined via chain rule.
- Great for multi-physics codes (e.g., many Jacobians) and advanced analysis (e.g., sensitivities)
- Albany uses Trilinos package Sacado for AD
  - AD accomplished via operator overloading + templating: floats/double data types replaced by AD types.

Automatic Differentiation Example:

\[ y = \sin(e^x + x \log x), \quad x = 2 \]

<table>
<thead>
<tr>
<th></th>
<th>( \frac{d}{dx} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>7.389</td>
</tr>
<tr>
<td></td>
<td>0.500</td>
</tr>
<tr>
<td></td>
<td>1.301</td>
</tr>
<tr>
<td></td>
<td>8.690</td>
</tr>
<tr>
<td></td>
<td>-1.188</td>
</tr>
</tbody>
</table>

Derivatives are as accurate as analytic computation – no finite difference truncation error!
Nonlinear model abstraction & libraries

- Access to the **embedded solvers** in Trilinos requires satisfaction of **ModelEvaluator** (nonlinear model) abstraction.

- Interface is **general** to accommodate computation of Jacobians, user-defined preconditioners, and stochastic Galerkin expansions.

- **Advanced capabilities**: embedded UQ (Stokhos), optimization (ROL), homotopy continuation (LOCA).

### “ModelEvaluator” Abstraction:

<table>
<thead>
<tr>
<th>Given:</th>
<th>Computes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$f(\dot{x}, \ddot{x}, x, p, t)$</td>
</tr>
<tr>
<td>$\dot{x}$</td>
<td>$W = \alpha \frac{df}{dx} + \beta \frac{df}{dx} + \omega \frac{df}{\ddot{x}}$</td>
</tr>
<tr>
<td>$\ddot{x}$</td>
<td>$\frac{df}{dp}$</td>
</tr>
<tr>
<td>$p$</td>
<td>$g$</td>
</tr>
<tr>
<td>$t$</td>
<td>$\frac{dg}{dx}$</td>
</tr>
</tbody>
</table>

$f = \text{residual}; x = \text{solution vec}; p = \text{parameters}; g = \text{responses}; t = \text{time}; W = \text{Jacobian}$
Core building blocks of Albany

➢ Component-based design.
➢ Template-based generic programming.
➢ Assembly/field evaluation via Phalanx.
➢ Automatic differentiation.
➢ Discretizations/meshes, mesh adaptivity.
➢ Solvers, time-integration schemes.
➢ Performance-portable kernels.
➢ Software quality tools: git, cmake, ctest, CDash.
**Assembly** of physics pieces comes down to the evaluation of a **directed acyclic graph (DAG)** of computations of field data.

**Phalanx package:** Local field evaluation kernel designed for assembly of arbitrary equation sets (i.e. evaluating residuals/Jacobians).

---

**Graph-based finite element assembly (FEA)**

**DAG from Albany:**

Elasticity

---

Template-specialized evaluators

Evaluator common to all FEAs

Problem-specific evaluator

---

Graph-based finite element assembly (FEA)

**Assembly** of physics pieces comes down to the evaluation of a **directed acyclic graph (DAG)** of computations of field data.

**Phalanx package:** Local field evaluation kernel designed for assembly of arbitrary equation sets (i.e. evaluating residuals/Jacobians).

- Template-specialized evaluators
- Evaluator common to all FEAs
- Problem-specific evaluator

---

**DAG from Albany:**

Elasticity

To add **new PDE**, all you need to code is problem-specific evaluator!

---

**Optimization** and **UQ** can be done through DAKOTA:

- **Optimization algorithms**: gradient-based local algorithms, pattern searches, and genetic algorithms, etc.
- **UQ algorithms**: Latin hypercube stochastic sampling, stochastic collocation, PCE, MCMC, etc.

“**Black box**” analysis tools at top level of software stack can perform a single forward solve, sensitivity analysis, parameter studies, bifurcation analysis, optimization, and UQ runs.
Libraries/algorithm whose development was significantly aided by Albany

<table>
<thead>
<tr>
<th>Libraries Developed in Albany:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Piro</td>
</tr>
<tr>
<td>➢ TriKota</td>
</tr>
<tr>
<td>➢ MiniTensor</td>
</tr>
<tr>
<td>➢ Razor (MOR)</td>
</tr>
<tr>
<td>➢ buildAgainstTrilinos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Libraries Matured in Albany:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Tempus</td>
</tr>
<tr>
<td>➢ Phalanx</td>
</tr>
<tr>
<td>➢ STK</td>
</tr>
<tr>
<td>➢ ModelEvaluator</td>
</tr>
<tr>
<td>➢ Stratimikos</td>
</tr>
<tr>
<td>➢ TPetra</td>
</tr>
<tr>
<td>➢ PUMI</td>
</tr>
<tr>
<td>➢ ROL</td>
</tr>
<tr>
<td>➢ DTK</td>
</tr>
<tr>
<td>➢ Intrepid2/Kokkos</td>
</tr>
<tr>
<td>➢ DynRankView</td>
</tr>
<tr>
<td>➢ And counting...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Libraries Driven by Albany:</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Stokhos Embedded UQ</td>
</tr>
<tr>
<td>➢ Semi-Coarsening AMG</td>
</tr>
<tr>
<td>➢ PAALS</td>
</tr>
<tr>
<td>➢ Advanced Topological Opt</td>
</tr>
<tr>
<td>➢ Embedded Ensembles</td>
</tr>
<tr>
<td>➢ CUBIT Mesh-Morpher</td>
</tr>
</tbody>
</table>
Quantum device modeling (QCAD)

Albany enabled the rapid stand-up of a world-class quantum device design tool.

- **Application**: quantum computing
- **Objective**: simulation/optimization of semiconductor quantum double dots
  - Provide fast feedback on which device layouts are most likely to lead to few-electron behavior
- **Key to QCAD’s success**: interfaces
  - Various multi-physics couplings of Poisson + Schrodinger
  - DAKOTA* for optimization.
- QCAD is used by experimentalist in Sandia’s world-class experimental facilities (CINT) as design tool for quantum device fabrication
  - iQCAD: GUI for experimentalists

QCAD = “Quantum Computer Aided Design”

1. Solid Model
2. GUI


* https://dakota.sandia.gov
Advanced Topology Optimization (ATO)

Coupling of Albany code and **PLATO** engine for optimization-based topology optimization.

**Goals:**

- **Qualification:** assure quality
- **Design:** effectively utilize AM

**PDE-constrained optimization:**

- **Physics:** elastostatics, Poisson
- **Objectives:** compliance, p-norm
- **Constraints:** volume/mass

**Multiple simultaneous Albany runs can inform a single design optimization by PLATO:**

- Albany implements objective + gradient evaluation, optimization loop
- **New “meshless” ATO capability:** allows user to include arbitrarily many simultaneous load cases (linear thermal/electrical, mechanical)

*Topology optimization-based design environment developed by SNL.*
Advanced Topology Optimization (ATO)

Coupling of Albany code and PLATO* engine for optimization-based topology optimization.

Goals:
➢ Qualification: assure quality
➢ Design: effectively utilize AM

PDE-constrained optimization:
➢ Physics: elastostatics, Poisson
➢ Objectives: compliance, p-norm
➢ Constraints: volume/mass

Multiple simultaneous Albany runs can inform a single design optimization by PLATO:
➢ Albany implements objective + gradient evaluation, optimization loop
➢ New “meshless” capability: geometry defined in constructive solid geometry (CSG) format and meshed inline
  ❖ Allows user to include arbitrarily many simultaneous load cases (linear thermal/electrical, mechanical, etc.)

* PLAtform for Topology Optimization: topology optimization-based design environment developed by SNL.
Ice sheets: Albany Land-Ice (ALI)

Albany enabled the **rapid development** of a production **land-ice dycore** for providing **actionable predictions** of **21st century sea-level rise** as a part of the DOE Energy Exascale Earth System Model (E3SM).

**Above**: ABUMIP-Antarctica experiment  
**Right**: Thwaites glacier retreat under parametrized submarine melting.

Multi-scale coupling via Schwarz

A domain decomposition alternating-Schwarz-based method has been developed in Albany for concurrent multi-scale coupling in solid mechanics.

- **Crux of Method**: use solutions in simple domains to iteratively build a solution for the more complex domain.
- **Targeted application**: failure of bolted components.
- **“Plug-and-play” framework**: simplifies meshing complex geometries!
  - Couple regions with different non-conformal meshes, element types, levels of refinement, solvers/time-integrators.

https://github.com/ORNL-CEES/DataTransferKit
Multi-scale coupling via Schwarz

A domain decomposition alternating-Schwarz-based method has been developed in Albany for concurrent multi-scale coupling in solid mechanics.

See talk by A. Mota: MS 350, Fri. Mar. 1, 10:10-10:30AM, Room 302B

Single domain

Schwarz coupling of hex (parts) + composite tet 10 (bolts) elements ($J_2$ material model from LCM suite)

Future work: extend method for multi-physics coupling

Performance-portable FEM

**Kokkos**: C++ library/programming model providing performance portability for applications

- Algorithm is written once for multiple architectures
- Template parameters specify optimal data layout for a given architecture.

*Kokkos github repo*: https://github.com/kokkos/Kokkos

I. Demeshko et. al., 2018.
Performance-portable FEM

Performance-portability of Albany FEA achieved using **Kokkos**; Albany usage has in turn led to **Kokkos improvements**

Algorithm is written **once** for multiple architectures

Template parameters specify **optimal data layout** for a given architecture.
In-memory mesh adaptation

Collaboration with SCOREC*: development of mesh adaptation capabilities in Albany to enable multi-scale/multi-physics adaptive simulation

PAALS (Parallel Albany Adaptive Loop with SCOREC)

- Fully-coupled in-memory adaptation, solution transfer.
- Parallel mesh infrastructure and services via PUMI.
- Dynamic load balancing (ParMetis/Zoltan, ParMA).

- Automated parallel goal-oriented adaptive simulation
  - Use adjoint solution to drive mesh adaptation
  - \(~100\times\) DoF-efficiency observed
  - Scaling out to at least 8K MPI ranks

- Performance portability to GPUs via Kokkos.

- Applications: 3D manufacturing, creep/plasticity in large solder joint arrays, coupled dislocation dynamics (Albany + ParaDis), ...

* Scientific Computation Research Center at RPI (Mark Shephard et al.)
3D manufacturing

- **Additive & subtractive** capabilities
- Employs **advanced adaptive meshing** and **evolving geometries** (using Symmetrix)
- Coupling with **feedback control**

**Right**: simulation of subtractive manufacturing with picosecond laser*

**Left**: Additive manufacturing simulation showing temperature in evolving geometry

Local-nonlocal coupling for integrated fracture modeling & multi-physics peridynamics simulations

- **Peridynamics**: nonlocal extension of continuum mechanics that remains valid at discontinuities/cracks
- **Peridigm** = open-source* peridynamics code
  - Nonlocal meshfree approach (Silling et al., 2005).
- “Best of both worlds” by combining FEM + peridynamics: peridynamics applied in regions susceptible to material failure, easy delivery to applications via FEM.
- **Optimization-based** local-to-nonlocal coupling using ROL (D’Elia et al. 2016).

* Peridigm github repo: https://github.com/peridigm/peridigm.
Albotran seeks to create a multi-physics geomechanical application that couples the flow response in PFLOTRAN with a mechanical response from Albany.

Albany-PFLOTRAN (Albotran)

- Albany + PFLOTRAN coupling strategy can be viewed as hybrid of a fully-coupled implicit solver + more loosely coupled iterative-solvers
  - Integrates extensive domain expertise (Albany/LCM)
  - Specialized solvers are not required for either code

**Right:** Albotran consolidation problem results. Porosities tracked independently in each code are identical.
Mesh adaptation applications

**Creep/Plasticity in Large Solder Joint Arrays***

- Strategic reliability process in semiconductor manufacturing
- Automated workflows with locally refined meshing
- Novel materials models
- Scaling out to 16K processors, 1B+ elements

---

**Coupled Dislocation Dynamics**

- Integrates Albany and ParaDis
- Computes dislocation dynamics (DD) in complex geometry
- Allows intersection of dislocations with free surfaces
- **Left**: prismatic dislocation loops in finite domain

---

Work in progress

Performance portability:

➢ Code optimizations for finite element assembly
➢ Performance portable solvers [WIP by Trilinos team]

Infrastructure work:

• Refactor of code to use block data structures to facilitate multi-physics coupling
  ➢ “Plug-and-play” different PDEs within Albany
  ➢ Ability to use block preconditioners (Teko*)

• Add support for mixed finite elements
  ➢ Can be accomplished via incorporation of Panzer and DOFManager.

Application-driven development:

**Land-ice:**
- Improved **basal hydrology** models for land-ice.
- **Level set** formulation to track better the calving.
- **Uncertainty quantification** (Bayesian inference, forward propagation).
- Seeking funding for developing **solid-mechanics-based ice fracture/calving models** for improved ice sheet models.

**LCM:**
- Modeling of structural components in **hypersonic vehicles** with large mechanical and thermal loads (USC).
- Enhancement of **subtractive manufacturing** capabilities in Albany (RPI).

**ATO:**
- **Meshless** topology optimization using Albany-PLATO.

*Much more...!*