Building an Open-source Multiphysics PDE Research Tool using Trilinos

R. P. Pawlowski, E. C. Cyr, J. N. Shadid, and T. M. Smith
Sandia National Laboratories

Trilinos User Group Meeting
Wednesday, November 2nd, 2011

SAND2011-8330C

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.

Thanks to:

- Chris Baker (ORNL)
- Ross Bartlett (ORNL)
- Todd Coffey (SNL)
- Debbie Fixel (SNL)
- Rick Garcia (SNL)
- Gary Hennigan (SNL)
- Zeses Karoutas (Westinghouse Electric Company)
- Paul Lin (SNL)
- Bill Rider (SNL)
- Sal Rodriguez (SNL)
- Andrew Salinger (SNL)
- Greg Sjaardema (SNL)
- Dan Turner (SNL)
- Alan Williams (SNL)

Sandia National Laboratories

Outline

- Introduction
  - Motivation
  - History
- An example driving application: CASL GTRF Challenge Problem
- Code Design and Algorithms
- GTRF Results
- *A Users(?) Perspective on Trilinos*
- Conclusions

Focus

- Question:
  - If I need to build a multiphysics simulation capability, how much can we leverage Trilinos and how much do I have to develop?
- “Power User Session”: This talk is about pushing the Trilinos open source software stack to it’s limits.
- Hope to Convey: With the right tools in place, you can make significant progress very quickly on very complex physics with advanced software designs!

Motivation

Mathematical /Computational Motivation: Achieving Scalable Predictive Simulations of Complex Highly Nonlinear Multi-physics PDE Systems

- Multiphysics systems are characterized by a myriad of complex, interacting, nonlinear multiple time- and length-scale physical mechanisms.
  - Dominated by **short dynamical time-scales**
  - Widely separated time-scales → **stiff system response**
  - Evolve a solution on a **long time scale relative to component time scales**
  - Balance to produce **steady-state** behavior.

e.g. Nuclear Fission / Fusion Reactors; Conventional /Alternate Energy Systems; High Energy Density Physics; Electro-magnetic Launch; Astrophysics; etc ….

- Our approach:
  - Stable and higher-order accurate implicit formulations and discretizations
  - Robust, scalable and efficient prec. for fully-coupled Newton-Krylov methods
  - Integrate sensitivity and error-estimation to enable UQ capabilities.

Sandia National Laboratories

History

- Research on Discretizations and Implicit Solution Technology for Large-scale PDE Simulation
  - Funding: DOE/SC/ASCR, NNSA/ASC, AFRL
  - Reacting flows, CFD, MHD, Aerosol, Semiconductor Drift Diffusion
- Foundational solvers work that contributed to many Trilinos packages
- Current simulation tool (Charon) will not meet our needs for FY12+:
  - ASCR funding requires new technology that doesn't easily fit the framework
  - ASC Target Problems are more complex
  - Export control restrictions
  - TPL dependencies on commercial software (e.g. Chemkin)
  - Monolithic framekwork
- Redesign for future research and production efforts in a collection of Trilinos packages

Research Requirements

A Research Tool for DOE/OS: ASCR/AMR, ASCR/UQ

- **Formulations: fully coupled fully implicit Newton-Krylov, semi-implicit, FCT**
- **Compatible discretizations:**
  - Mixed basis for DOFs within element block
  - Arbitrary element types (not restricted to nodal basis)
- **Multiphysics:**
  - Fully coupled systems composed of different equation sets in different element blocks
  - Preconditioning: Approximate block factorization/physics based
- **Supports advanced analysis techniques:**
  - Supports Template-based Generic Programming
  - Adjoint-based error analysis
  - Stability, bifurcation, embedded (SAND) optimization, embedded uncertainty quantification (Stokhos/PCE)
- **Open-source (collaborations)**
- **Exascale integration (Tpetra, Kokkos::MDArray)**

Production Requirements

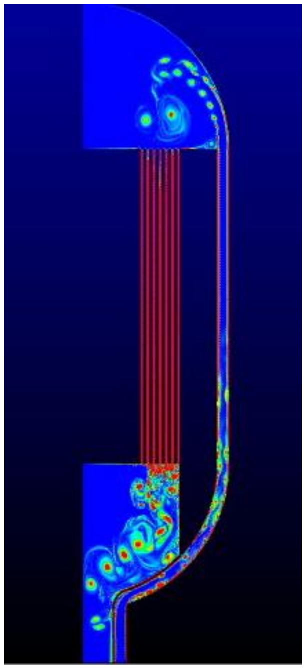
Production Quality Software (ASC, CASL)

- Strict and extensive unit testing (TDD), system testing, automated nightly and CI testing
- Application is a library (for multiphysics coupling)
- Integration with legacy code components
- NOT restricted to any mesh database or I/O format
- Control over granularity of assembly process (efficiency vs flexibility)
- Production Applications:
  - **ASC**: Semiconductor Device (Next-generation Charon) for QASPR
  - **CASL**: Drekar CFD component for VERA simulator

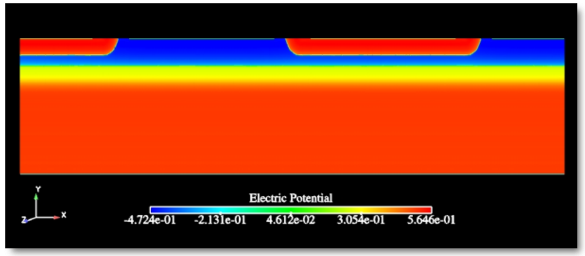
DOE / NNSA

Rapid Development of New Physics

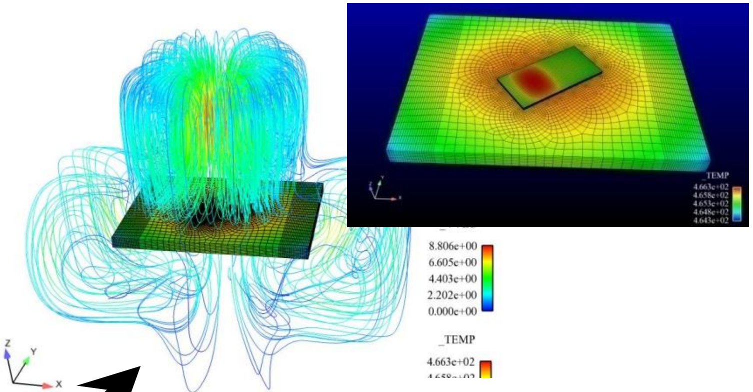
(Single driver and collection of interchangeable physics models)



NGNP Reactor

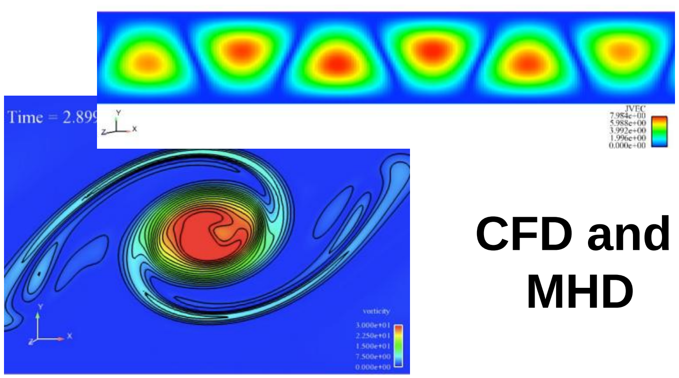


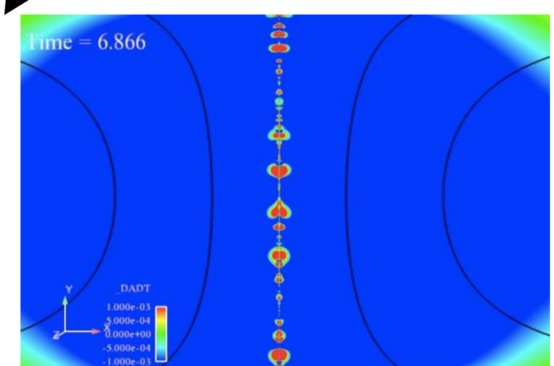
Semiconductor Drift Diffusion



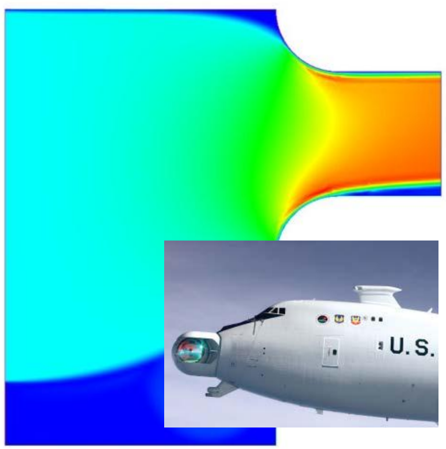
Chemicurrent

CHARON





CFD and MHD



Multi-phase Chemically Reacting Aerosol

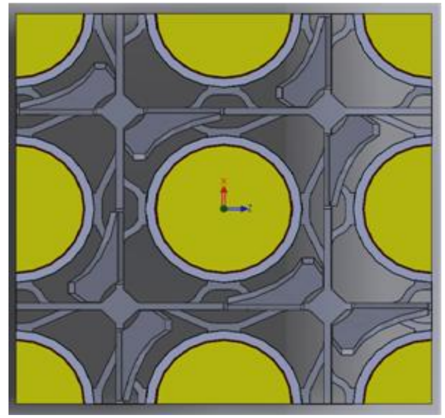
Our Philosophy

- Leverage all of Trilinos
- Where appropriate, generalize your application code into Trilinos packages!
  - Object-oriented, lightweight, flexible components that can be swapped out according to bleeding edge research in Trilinos (Exascale)!
- Relevant Trilinos packages:
  - Trilinos/Phalanx: (Template-based Generic Programming Tools)
    - Developed in 2009
  - Trilinos/Panzer: General Finite Element Assembly Engine
    - Started Oct 2010
  - Drekar (Trilinos package in external repository)
    - Started late December 2010
- A Target application: CASL Grid-to-Rod Fretting:
  - Code V&V, Initial demonstration runs, and VUQ analysis
  - Due June 30th 2011

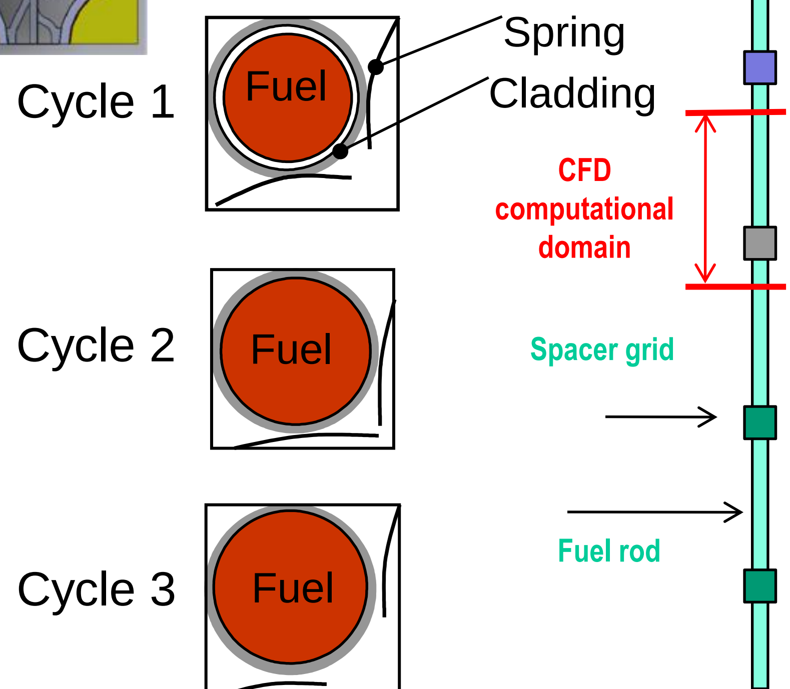
Sandia National Laboratories

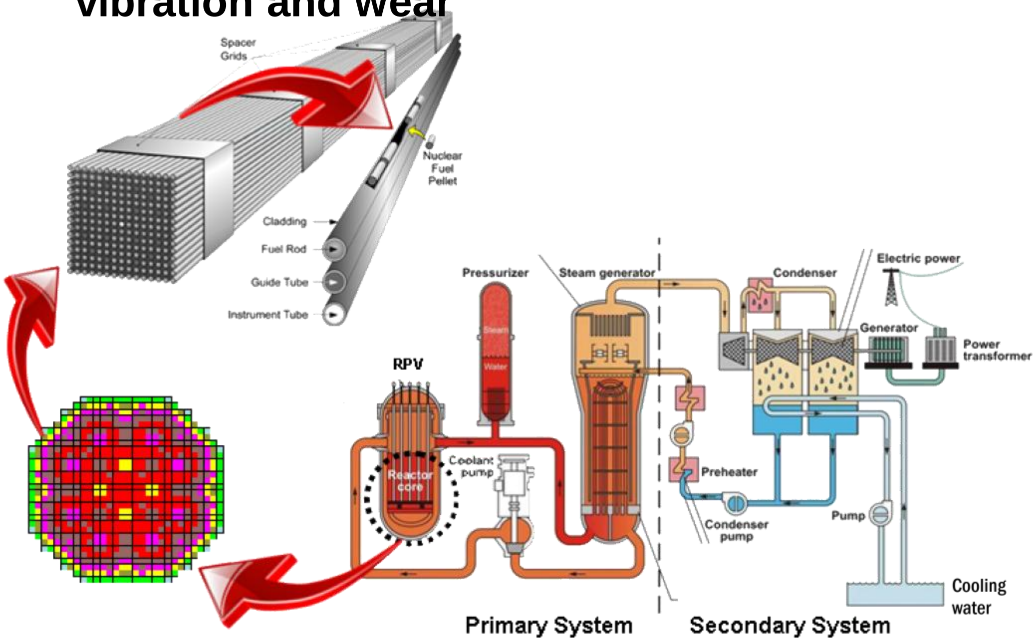
Challenge Problem: Grid-to-Rod Fretting

- Cladding failure can occur as the result of flow induced vibration
- Mixing vanes on spacer grid are used to produce turbulence to improve heat transfer between rod and fluid
- High-fidelity, FSI to predict gap, turbulent flow excitation, rod vibration and wear



Spacer grid cell





Governing Equations
(Unsteady Single-phase, Isothermal, Incompressible Flow)

- Navier-Stokes with spatially filtered LES

Momentum $$R_{\mathbf{u}} = \frac{\partial(\bar{\rho}\bar{\mathbf{u}})}{\partial t} + \nabla \cdot (\bar{\rho}\bar{\mathbf{u}} \otimes \bar{\mathbf{u}} - \mathbf{T})$$

Continuity $$R_p = \frac{\partial \bar{\rho}}{\partial t} + \bar{\rho} \nabla \cdot \bar{\mathbf{v}}$$

over bar denotes spatial filtering in LES

$$\mathbf{T} = -\bar{P}\mathbf{I} + \mu_{eff}(\nabla \bar{\mathbf{u}} + \nabla \bar{\mathbf{u}}^T)$$

$$\mu_{eff} = \mu + \mu_t$$ (molecular and eddy viscosity)

$$\mu_t = \bar{\rho}\nu_t$$

$$\bar{\mathbf{S}}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{\mathbf{u}}_i}{\partial \mathbf{x}_j} + \frac{\partial \bar{\mathbf{u}}_j}{\partial \mathbf{x}_i}\right) \quad \bar{\Omega}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{\mathbf{u}}_i}{\partial \mathbf{x}_j} - \frac{\partial \bar{\mathbf{u}}_j}{\partial \mathbf{x}_i}\right)$$

$$\nu_t = (C_w \Delta)^2 \frac{\left(\bar{\mathbf{S}}^d_{ij}\bar{\mathbf{S}}^d_{ij}\right)^{3/2}}{\left(\bar{\mathbf{S}}_{ij}\bar{\mathbf{S}}_{ij}\right)^{5/2} + \left(\bar{\mathbf{S}}^d_{ij}\bar{\mathbf{S}}^d_{ij}\right)^{5/4}}$$

$$\bar{\mathbf{S}}^d_{ij} = \bar{\mathbf{S}}_{ik}\bar{\mathbf{S}}_{kj} + \bar{\Omega}_{ik}\bar{\Omega}_{kj} - \frac{1}{3}\delta_{ij}\left[\bar{\mathbf{S}}_{mn}\bar{\mathbf{S}}_{mn} + \bar{\Omega}_{mn}\bar{\Omega}_{mn}\right]$$

- Wall Adapting Local Eddy-viscosity model (WALE)
  - Nicoud, F. and Ducros, F., “Subgrid-Scale Stress Modelling on the Square of the Velocity Gradient Tensor,” Flow Turbulence and Combustion, Vol. 62, 1999, pp. 183-200.
- Modified Smagorinsky eddy-viscosity model
  - Filter width is based on the square of the deviatoric stress tensor
  - Requires only "local data" to construct.
  - Recovers the proper near-wall scaling for the eddy viscosity so that it inherently decays to zero as the wall is approached without using a dynamic procedure or wall model

Spatial Discretization

- Stabilized Galerkin Finite Element 2nd order (2nd-8th available)
- PSPG allows equal order interpolation of velocity and pressure
- SUPG operator to limit oscillations in high grid Re flows.

$$\mathbf{F}_{\mathbf{m}i} = \underbrace{\int_{\Omega} \Phi R_{u_i} d\Omega}_{\text{Galerkin}} + \underbrace{\sum_e \int_{\Omega_e} \tau_m \left(\bar{\mathbf{u}} \cdot \nabla \Phi\right) R_{u_i} d\Omega}_{\text{SUPG}}$$

$$\mathbf{F}_{\mathbf{p}} = \underbrace{\int_{\Omega} \Phi R_p d\Omega}_{\text{Galerkin}} + \underbrace{\sum_e \int_{\Omega_e} \bar{\rho} \tau_m \left(\nabla \Phi \cdot R_u\right) d\Omega}_{\text{PSPG}}$$

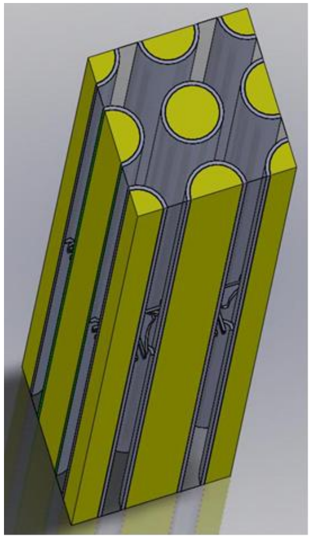
- Future work will extend to full VMS-LES models
  - T.J.R. Hughes, L. Mazzei, K.E. Jansen, Large eddy simulation and the variational multiscale method, Comput. Vis. Sci. Vol. 3, 2000, pp. 47-59

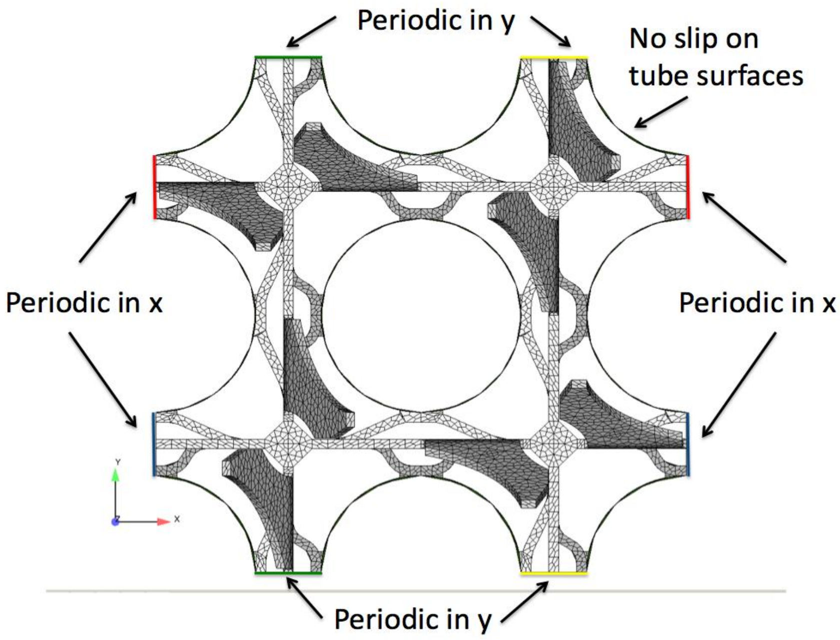
Sandia National Laboratories

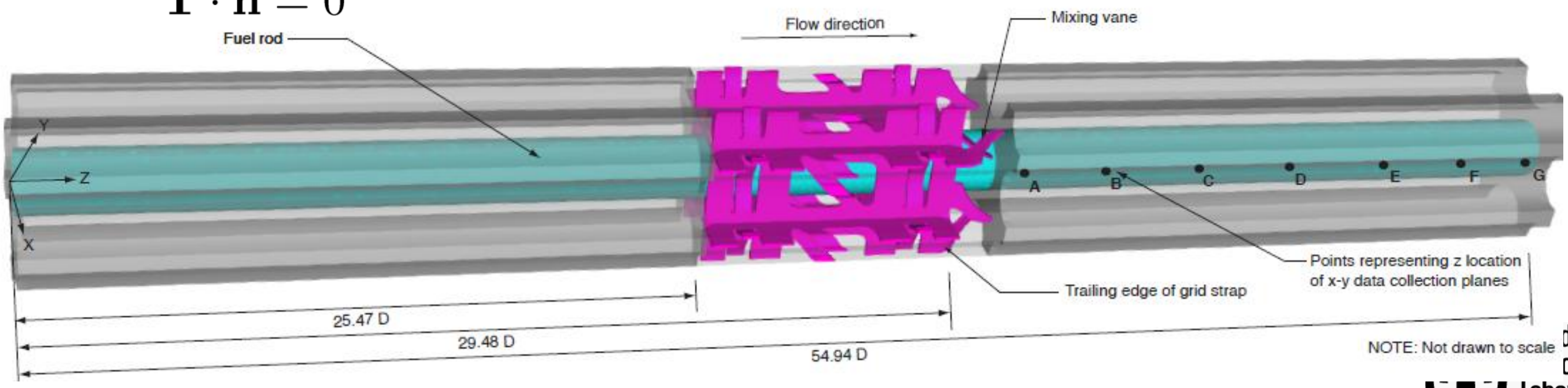
Problem Description (3x3 Rod Bundle)

- Isothermal
- Fluid: Water
  - T: 394K
  - Viscosity: 2.32×10^{-4} Pa sec
  - Density: 924 kg/m^3
- Re ~ 2x10^5
- Symmetry on sides
- No slip (v=0) on rods
- Inflow on bottom
  - 5 m/sec
- Outflow on top:

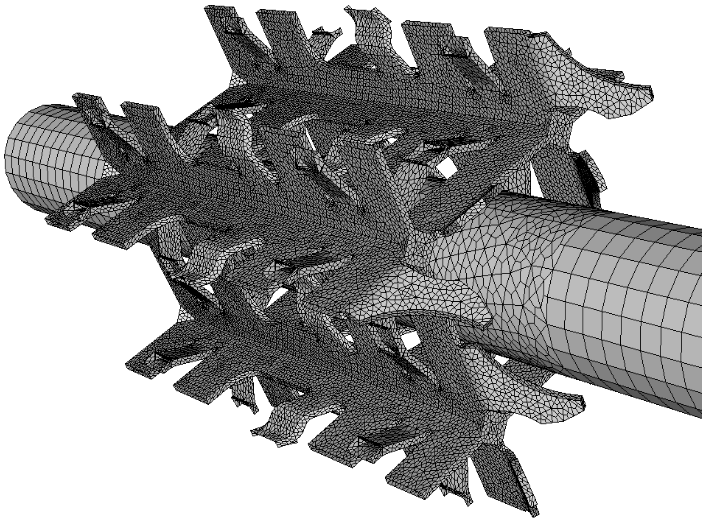
$$\mathbf{T} \cdot \mathbf{n} = 0$$

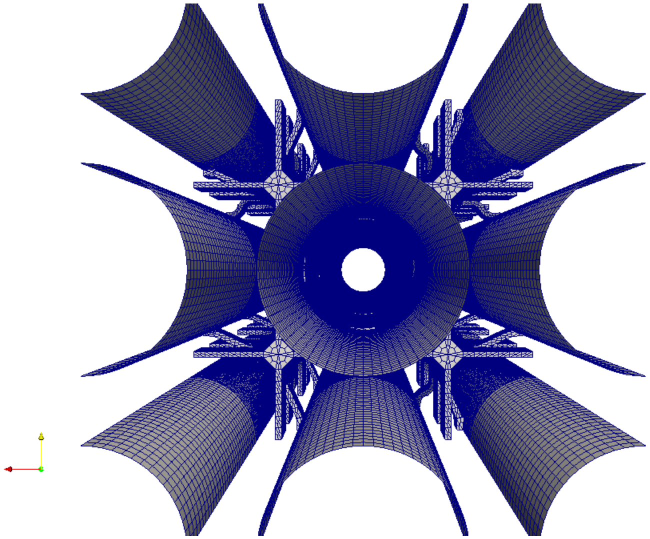


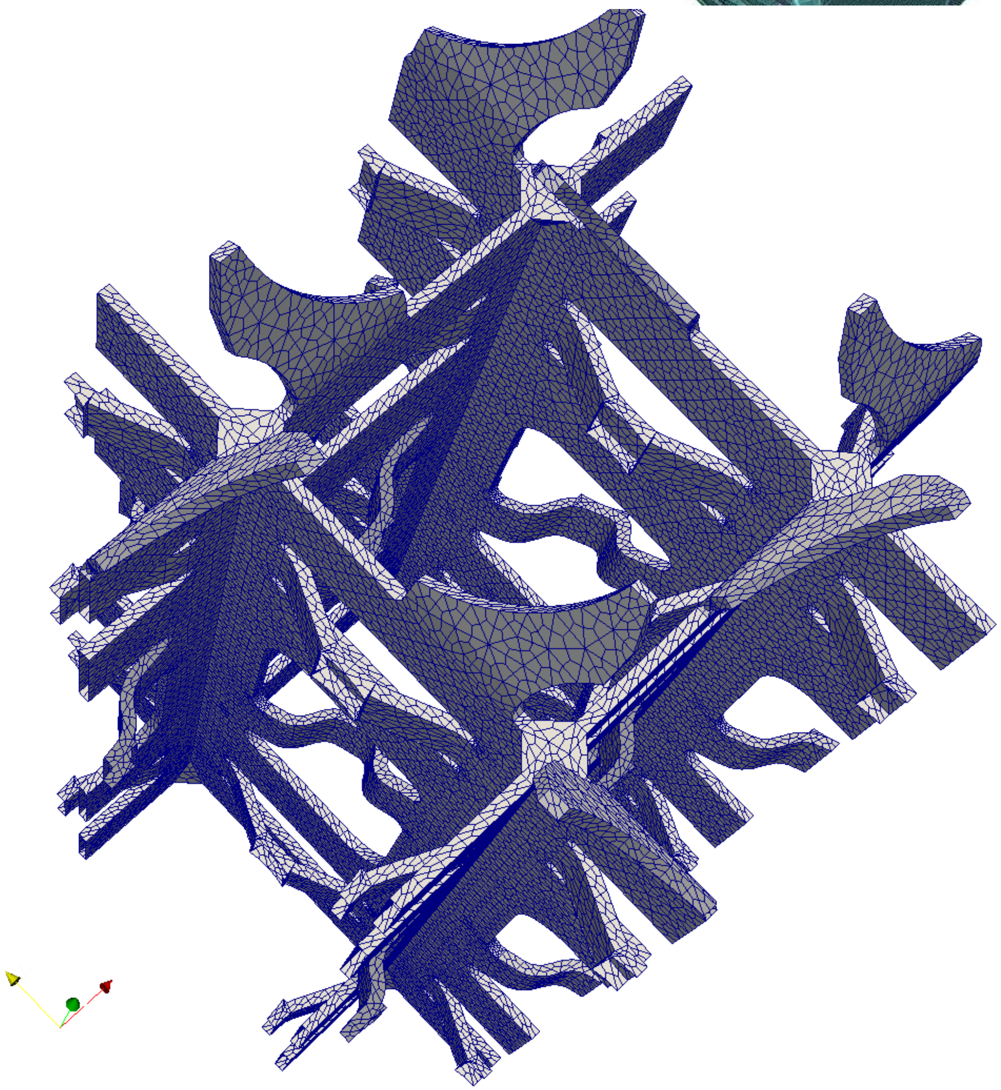




Geometry
(Hexahedral Elements)







Code Design

Agile Components (A. Salinger): Trilinos has a coordinated integration effort (ASC) to support all aspects of a simulation!

Analysis Tools (*non-invasive*)
- Optimization
- Parameter Studies
- UQ (non-invasive)
- V&V, Calibration
- OUU, Reliability
- Computational Steering

Analysis Tools (*invasive*)
- Nonlinear Solver
- Time Integration
- Continuation
- Sensitivity Analysis
- Stability Analysis
- Constrained Solves
- Optimization
- UQ Solver

Linear Algebra
- Data Structures
- Iterative Solvers
- Direct Solvers
- Eigen Solver
- Preconditioners
- Matrix Partitioning
- Architecture-Dependent Kernels

Mesh Tools
- Mesh I/O
- Inline Meshing
- Partitioning
- Load Balancing
- Adaptivity
- Remeshing
- Grid Transfers
- Mesh Quality

Mesh Database
- Mesh Database
- Geometry Database
- Solution Database

Local Fill

Discretizations
- Discretization Library
- Variable Manager

Derivative Tools
- UQ / PCE Propagation
- Derivatives
- Sensitivities

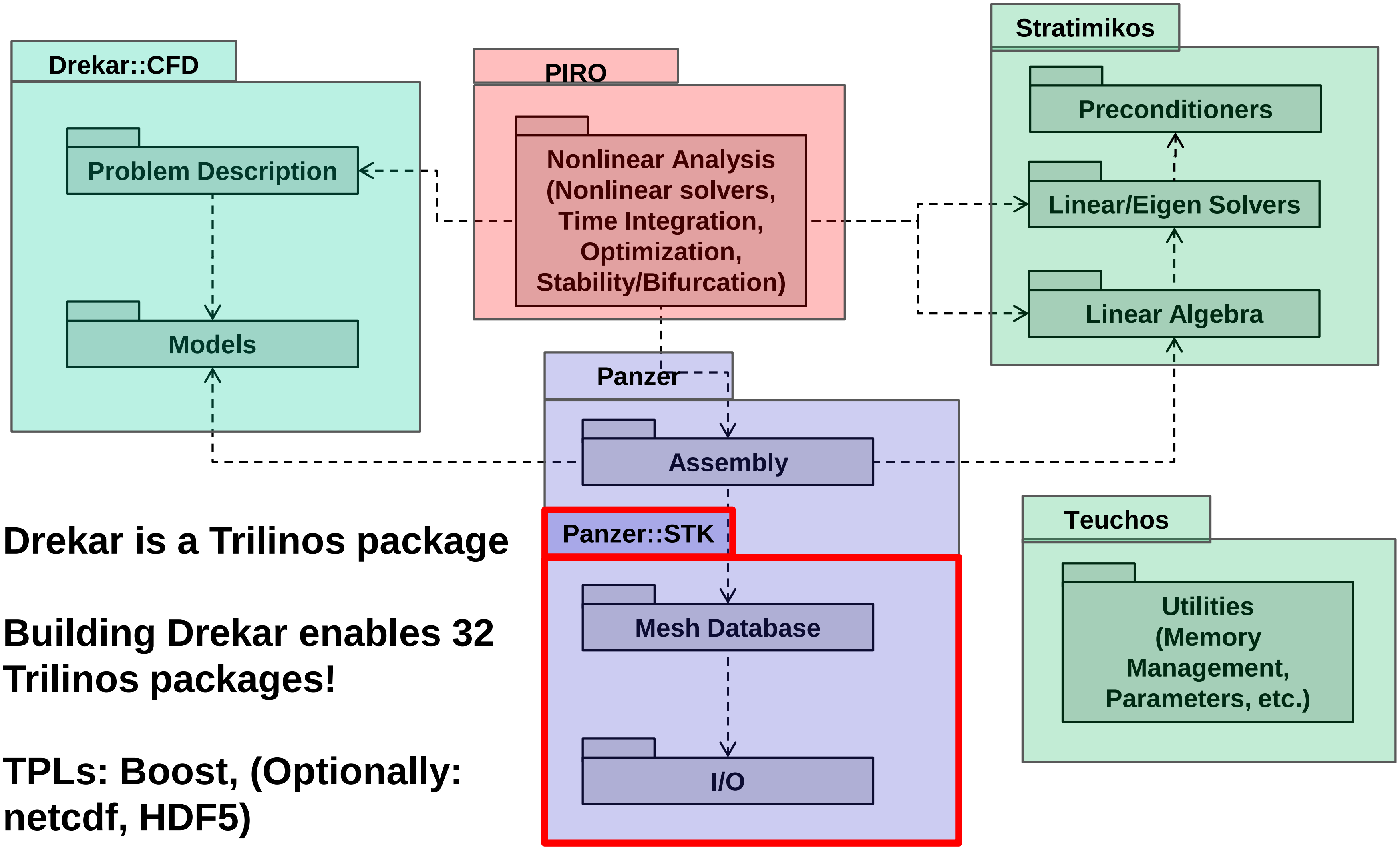
Physics Fill
- Element Level Fill
- Material Models
- Objective Function
- Constraints
- Error Estimates
- MMS Source Terms

Composite Physics
- MultiPhysics Coupling
- Solution Control
- System Models
- System UQ

Utilities
- Input File Parser
- Parameter List
- I/O Management
- Memory Management
- Communicators
- Runtime Compiler
- MultiCore Parallelization Tools

PostProcessing
- Visualization
- Verification Tools
- Feature Extraction
- Data Reduction
- Model Reduction

Software Design
(Composition of Trilinos Packages)



- **Drekar is a Trilinos package**
- **Building Drekar enables 32 Trilinos packages!**
- **TPLs: Boost, (Optionally: netcdf, HDF5)**

Sandia National Laboratories

Introducing Drekar

(Named for the Viking Longship)

- A **light-weight front end** "Trilinos package" that provides Stabilized Galerkin CFD and MHD physics
- Provides mathematical kernels to evaluate the discretized PDEs using TBGP concepts
- Panzer/Drekar package dependencies:
  - 10 required
  - 9 optional
- Indirect dependencies: 32 enabled packages (including Drekar itself)

Panzer and Drekar

Trilinos Discretization Tool Stack

(Pawlowski,Cyr, Shadid, Smith)

PIRO (Solvers)
NOX
Rythmos
MOOCHO
LOCA
LIME

Thyra Model Evaluator

Drekar::CFD

- Input ParameterList
- Equation Set Factory
- BC Factory
- Evaluator Factory

Phalanx (TBGP)

Intrepid (FE Basis/IR)

Shards (Cell Topology)

Stokhos (UQ)

Sacado (AD)

Panzer

- **Multiphysics Assembly Engine:**
  - Fully coupled Multiphysics
  - Compatible discretizations
  - Multiple Equation sets
  - Arbitrary BCs
- **DOF Manager**
  - Mapping DOFs
  - ConnectionManager

FEI (DOF Mapping Strategy)

STK (Mesh Database)

SEACAS (I/O, Partitioning)

Assembly Engine

- The assembly engine is the core of the multiphysics support
  - Supplies the residuals, Jacobians, etc. required for fully-coupled/implicit solution methods
  - Supplies physics specific preconditioner operators
- This was discussed in the talk on Panzer this morning!

Graph-based Assembly Process

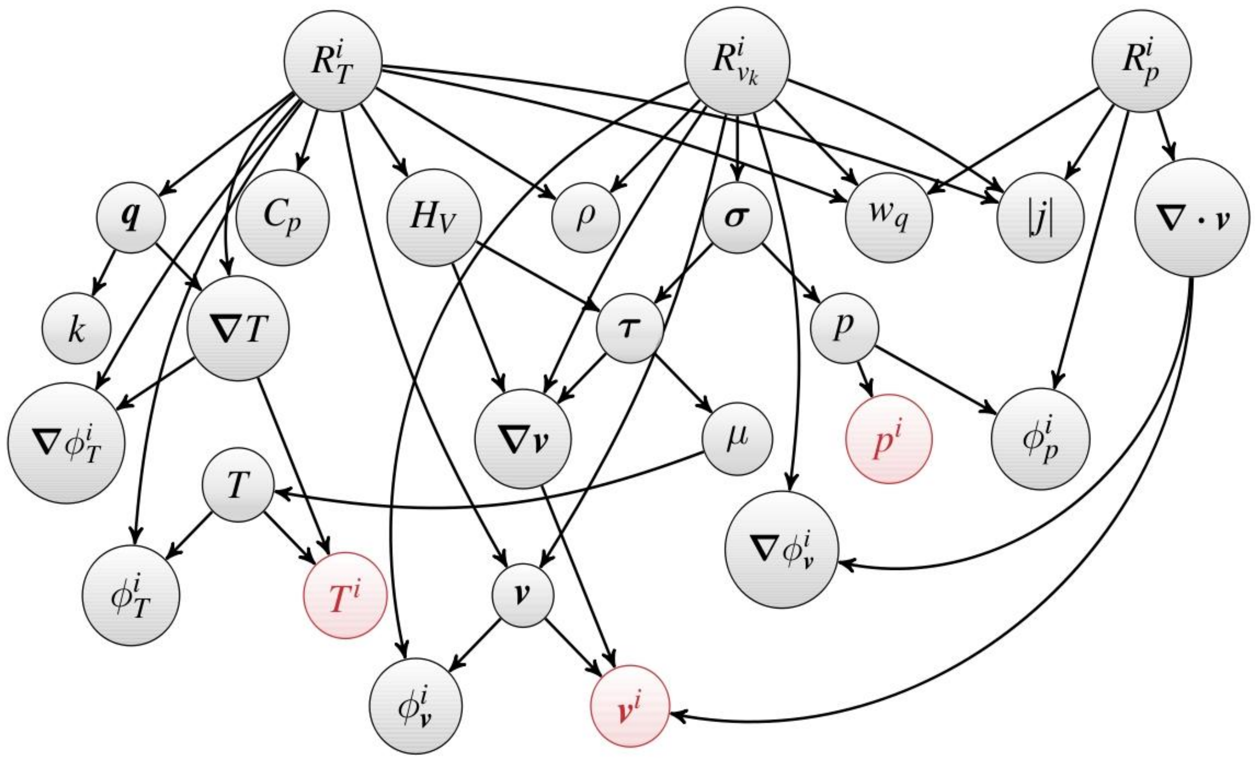
(Notz, Pawlowski, Sutherland; submitted to TOMS)

- Phalanx package
- Graph-based equation description
  - Automated dependency tracking (Topological sort to order the evaluations)
  - Each node is a point of extension that can be swapped out
  - Easy to add equations
  - Easy to change models
  - Easy to test in isolation
- Ideal for multiphysics
  - More equations → adding more nodes to graph
  - Reuse fields
- Multi-core research:
  - Spatial decomposition (Kokkos::MDArray)
  - Algorithmic decomposition

$$R_T^i = \sum_{e=1}^{N_e}\sum_{q=1}^{N_q}\left[\left(\rho C_p \boldsymbol{v}\cdot\nabla T - H_V\right)\phi_T^i - \boldsymbol{q}\cdot\nabla\phi_T^i\right] w_q|j| = 0$$

$$R_{v_k}^i = \sum_{e=1}^{N_e}\sum_{q=1}^{N_q}\left[\rho\boldsymbol{v}\cdot\nabla\boldsymbol{v}\phi_{\boldsymbol{v}}^i + \boldsymbol{\sigma} : \nabla\left(\phi_{\boldsymbol{v}}^i\boldsymbol{e}_k\right)\right] w_q|j| = 0$$

$$R_p^i = \sum_{e=1}^{N_e}\sum_{q=1}^{N_q}\nabla\cdot\boldsymbol{v}\phi_p^i w_q|j| = 0$$



Solution Algorithm Complexity

- Forward solves using implicit methods (Newton-based):

Find $x_* \in \mathbb{R}^n$ **such that** $F(x_*) = 0$ **where** $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$

  – Requires Jacobian or Jacobian vector product (JFNK):

$$J(x)_{ij} = \frac{\partial F_i}{\partial x_j} \qquad J(x) \in \mathbb{R}^{n \times n} \qquad Jv$$

- SAND: PDE Const. Optimization, Stability Analysis, Bifurcation Analysis:
  – Parameter sensitivities
  – Hessian

$$\frac{\partial F}{\partial p} \qquad \frac{\partial^2 F}{\partial x_i \partial x_j}$$

- Concept: Template-Based Generic Programming
  – Decouple the physics description from the requirements of the solution/analysis capabilities
  – The key: in c++, template the *scalar type and overload math operators using expression templates*

Sandia National Laboratories

Generic Programming Example: Templating the Scalar Type

$$f_0 = 2x_0 + x_1^2$$
$$f_1 = x_0^3 + sin(x_1)$$

```
// double version
void computeF(double* x, double* f)
{
  f[0] = 2.0 * x[0] + x[1] * x[1];
  f[1] = x[0] * x[0] * x[0] + sin(x[1]);
}
```

```
void computeJ(double* x, double* J)
{
  // J(0,0)
  J[0] = 2.0;
  // J(0,1)
  J[1] = 2.0 * x[1];
  // J(1,0)
  J[2] = 3.0 * x[0] * x[0];
  // J(1,1)
  J[3] = cos(x[1]);
}
```

Writing derivatives in the context of multiphysics systems with changing dependency chains is difficult and error prone!

```
// ad version
template <typename ScalarT>
void computeF(ScalarT* x, ScalarT* f)
{
  f[0] = 2.0 * x[0] + x[1] * x[1];
  f[1] = x[0] * x[0] * x[0] + sin(x[1]);
}
```

ScalarT → double	Residual
ScalarT → Dfad<double>	Jacobian

Machine precision accuracy: No FD involved!

Generic Programming

(using data types from Trilinos/Sacado: E. Phipps)

Field Manager is templated on Evaluation Type

Concept: Evaluation Types

- **Residual** $F(x,p)$ — `double`
- **Jacobian** $J = \frac{\partial F}{\partial x}$ — `DFad<double>`
- **Hessian** $\frac{\partial^2 F}{\partial x_i \partial x_j}$ — `DFad< DFad<double> >`
- **Parameter Sensitivities** $\frac{\partial F}{\partial p}$ — `DFad<double>`
- **Jv** Jv — `DFad<double>`
- **Stochastic Galerkin Residual** — `Sacado::PCE::OrthogPoly<double>`
- **Stochastic Galerkin Jacobian** — `Sacado::Fad::DFad< Sacado::PCE::OrthogPoly<double> >`

Scalar Types

(listed beside each evaluation type above)

NOTES:

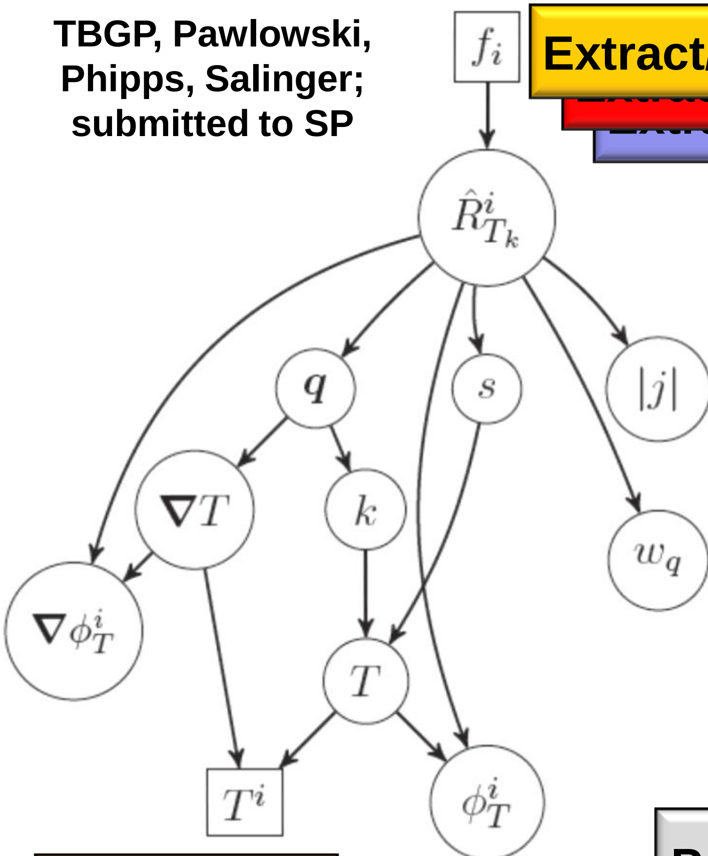
1. Not tied to double (can do arbitrary precision)
2. Not tied to any one scalar type can use multiple scalar types in any evaluation type!

Sandia National Laboratories

Phalanx Handles Multiphysics Complexity using Template-based Generic Programming

TBGP, Pawlowski, Phipps, Salinger; submitted to SP

Extract/Scatter



Gather/Seed

$$f(x) = \sum_{k=1}^{N_w} f_k = \sum_{k=1}^{N_w} Q_k^T \hat{R}^i_{T_k}(P_k x)$$

$$\hat{R}^i_T = \sum_{e=1}^{N_e} \sum_{q=1}^{N_q} \left[-\nabla \phi^i_T \cdot q + \phi^i_T s \right] w_q |j| = 0$$

Evaluation Type	Scalar Type
$f(x,p)$	`double`
$J = \frac{\partial f}{\partial x}$	`DFad<double>`
$\frac{\partial^2 f}{\partial x_i \partial x_j}$	`DFad< DFad<double> >`

Param. Sens., Jv, Adjoint, PCE (SGF, SGJ), Arb. Prec.

`PCE::OrthogPoly<double>`

`DFad<PCE::OrthogPoly<double> >`

Take Home Message:
Reuse the same code base!
Equations decoupled from algorithms!
Machine precision accuracy!

Sandia National Laboratories

Flexibility of Evlauation Types

```
struct Traits : public PHX::TraitsBase  {

  // ***************************************************************
  // *** Scalar Types
  // ***************************************************************
  typedef double RealType;
  typedef Sacado::Fad::DFad<RealType> FadType;

  #ifdef HAVE_STOKHOS
    typedef Stokhos::StandardStorage<int,RealType> SGStorageType;
    typedef Sacado::PCE::OrthogPoly<RealType,SGStorageType> SGType;
    typedef Sacado::Fad::DFad<SGType> SGFadType;
  #endif

  // ***************************************************************
  // *** Evaluation Types
  // ***************************************************************
  struct Residual { typedef RealType ScalarT; };
  struct Jacobian { typedef FadType ScalarT;  };

  #ifdef HAVE_STOKHOS
    struct SGResidual { typedef SGType ScalarT; };
    struct SGJacobian { typedef SGFadType ScalarT;  };
  #endif
```

```
// ***************************************************************
// *** MPL Vector of Evaluation Types
// ***************************************************************

typedef Sacado::mpl::vector<Residual, Jacobian
                     #ifdef HAVE_STOKHOS
                       , SGResidual, SGJacobian
                     #endif
                    > EvalTypes;
```

Jacobian-Free Newton-Krylov (JFNK)

$$x_{k+1} = x_k + \alpha \Delta x$$

$$J_k \triangle x = -F_k$$

<u>**Iterative Linear Solver – GMRES**</u>

Krylov Subspace of the form:

$$\mathcal{K}(A, v) \equiv \text{span}\{v, Av, A^2 v, ..., A^{m-1} v\}$$

In the inner iteration of the linear solve, we only need the action of the Jacobian on a vector:

$$Jv \approx \frac{F(x + \delta v) - F(x)}{\delta}$$

Only require an explicit matrix for preconditioning – does NOT have to be exact!

Advantages:

- <u>**Same as Newton, but no Jacobian is required**</u>**!**
- **Residual Based!**

Disadvantage:

- **Accuracy/convergence issues due to scalar perturbation factor:**

$$\delta = \lambda \left(\lambda + \frac{||\mathbf{x}||}{||\mathbf{v}||} \right)$$

- **Solution vector scaling is critical**

Sandia National Laboratories

Example: JFNK
(2D Diffusion/Rxn System: 2 eqns)

- **JFNK (FD)**

$$Jv \approx \frac{F(x+\delta v) - F(x)}{\delta}$$

$$t \approx (\text{num\_Its}) * cost(F)$$

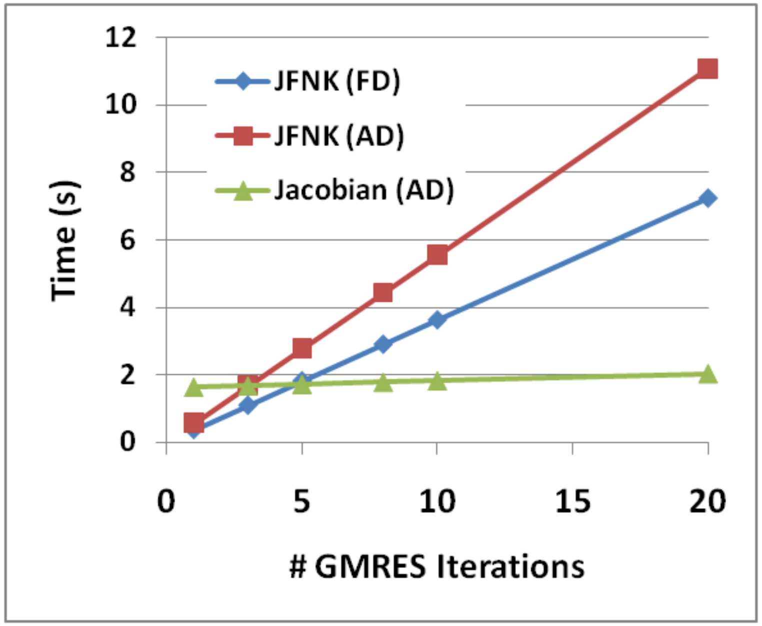
- **JFNK (AD)**
  - **Machine precision accurate**
  - **Ex: Solution varies 10^12 over domain**

$$Jv \leq 2.5 * cost(F)$$
$$t \approx 1.53 * (num\_Its) * cost(F)$$

- **Explicit Jacobian (AD generated)**
  - **Machine precision accurate**
  - **Complexity ideas allow for storing individual operators for preconditioning!**
  - **Larger memory requirements**

$$J(x) \leq 13 * cost(F)$$
$$t \approx 4.45 + (num\_Its) * cost(Mv)$$

	Relative times	
JFNK (AD)	F(x)	1.00
Explicit J (AD)	J(x)	4.45
JFNK (AD)	Jv (AD)	1.53
Explicit J (AD)	Mv (matvec)	0.06



Sandia National Laboratories

Build System

- Panzer and Drekar are both Trilinos packages
  - Panzer will be released as official Trilinos package
- Reside in separate git repository
  - Uses external Trilinos package support
- Leverage the build and testing infrastructure
- Under CI, nightly testing as part of CASL

Utilities

- Memory Management
  - **Teuchos** memory management classes: RCP, ArrayRCP, …
  - Has some important features missing in boost/tr1
    - Array view semantics, extra data
  - Common look and feel with Trilinos packages

- Input Objects are constructed with Teuchos ParameterLists
  - Validation is critical for robust usage
  - Common look and feel with Trilinos packages

- Boost (Phalanx Assembly)
  - Template metaprogramming library (MPL)
  - Graph library
  - Tokenizer (for one input path)
  - Hash Table

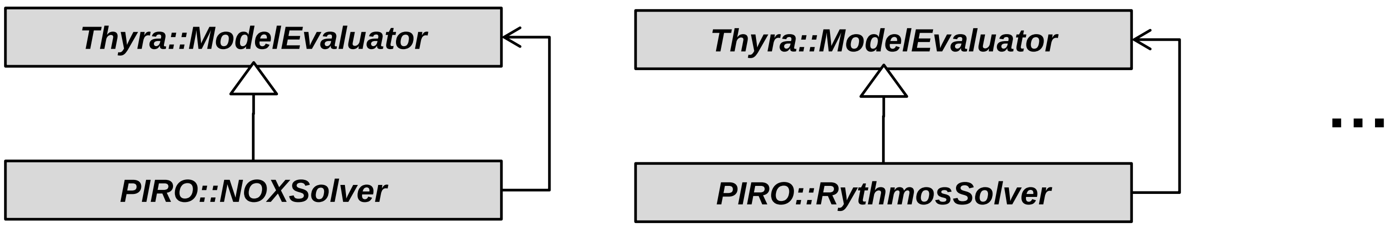
Problem Description

- Driven by Teuchos::ParameterList
  - Runtime configurable database
  - Let developers write their own front end to populate:
    - XML (automatically supported)
    - GUI (Trilinos/Optika/QT)
    - Traditional text parsing (boost::tokenizer)

- Issues
  - ParameterList XML reader does not preserve ordering (not standards compliant)
  - Validation doesn't quite support our use case
    - Dynamic (user defined) trees: sublist names are designated by the user at runtime

Nonlinear Analysis

- We use the PIRO package to interface to nonlinear analysis tools
  - Gives access to NOX, LOCA, Rythmos, and MOOCHO through a single interface
  - Model evaluator driven
  - **ParameterList + ModelEvaluator (+ optional observers) = Solver**



Building a Nonlinear Analysis Model Evaluator is this Simple!

```
RCP<Teuchos::ParameterList> piro_params =
  Teuchos::rcp(new Teuchos::ParameterList(solncntl_params));

RCP<Thyra::ModelEvaluatorDefaultBase<double> > piro;

if (solver=="NOX") {
  piro = Teuchos::rcp(new Piro::NOXSolver<double>(piro_params, drekar_me));
}
else if (solver=="Rythmos") {
  piro = Teuchos::rcp(new Piro::RythmosSolver<double>(piro_params, drekar_me,
            observer_factory->buildRythmosObserver(mesh,dofManager,ep_lof)));
}
else {
```

Why Newton-Krylov Methods?

Newton-Krylov

Direct-to-steady-state

Fully-implicit transient

Convergence properties

- Strongly coupled multi-physics often requires a strongly coupled nonlinear solver
- Quadratic convergence near solutions (backtracking, adaptive convergence criteria)
- Often only require a few iterations to converge, if close to solution, independent of problem size

$$\mathbf{F}(\mathbf{x}, \lambda_1, \lambda_2, \lambda_3, ..) = \mathbf{0}$$

Inexact Newton-Krylov

Solve $\mathbf{J}\mathbf{p}_k = -\mathbf{F}(\mathbf{x}_k)$; until $\dfrac{\|\mathbf{J}\mathbf{p}_k + \mathbf{F}(\mathbf{x}_k)\|}{\|\mathbf{F}(\mathbf{x}_k)\|} \le \eta_k$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Theta \mathbf{p}_k$$

Jacobian Free N-K Variant

$$\mathbf{M}\mathbf{p}_k = \mathbf{v}$$

$$\mathbf{J}\mathbf{p}_k = \frac{\mathbf{F}(\mathbf{x} + \delta \mathbf{p}_k) - \mathbf{F}(\mathbf{x})}{\delta}\ ;\ \text{or by AD}$$

See e.g. Knoll & Keyes, JCP 2004

Why Newton-Krylov Methods?

Newton-Krylov

Direct-to-steady-state

Fully-implicit transient

$$\mathbf{F}(\dot{\mathbf{x}},\mathbf{x},\lambda_1,\lambda_2,\lambda_3,..)=\mathbf{0}$$

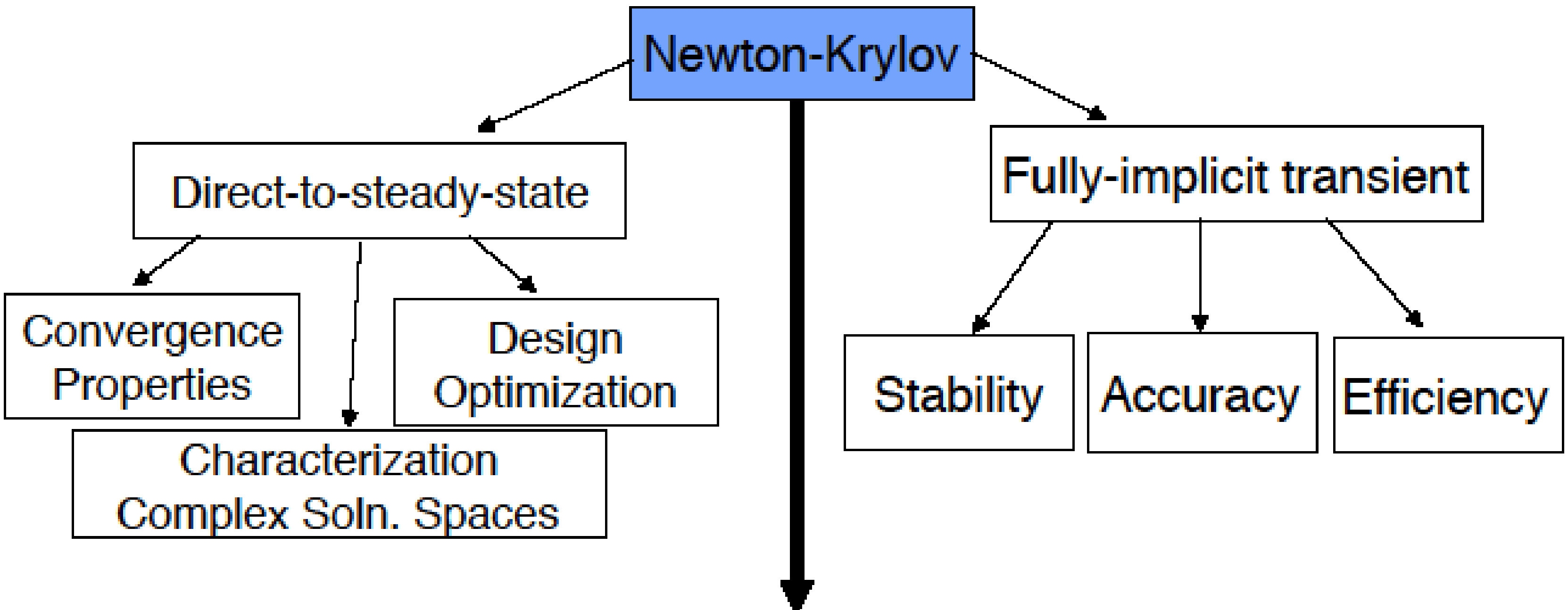
e.g.

$$\left.\frac{\partial c}{\partial t}\right|^{n+1}+\nabla\bullet\left(\left[\rho c\mathbf{u}\right]^{n+1}\right)-\nabla\bullet\left[D^{n+1}\nabla c^{n+1}\right]+S_c^{n+1}=0$$

<u>Stability and Accuracy Properties</u>

- Stable (stiff systems)
- High order methods
- Variable order techniques
- Local and global error control possible
- Can be stable and accurate run at the dynamical time-scale of interest in multiple-time-scale systems (e.g. Knoll et. al., Brown & Woodward., Chacon and Knoll, Ropp & S.)

Why Newton-Krylov Methods?

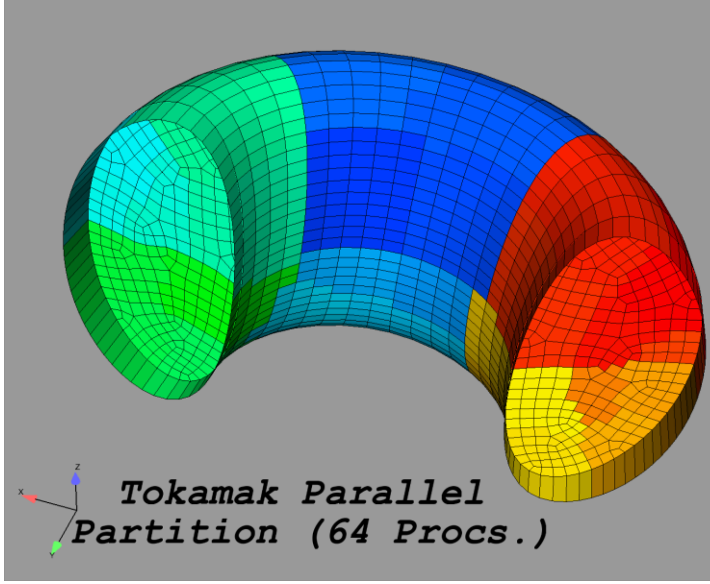


Very Large Problems -> Parallel Iterative Solution of Sub-problems

Krylov Methods - Robust, Scalable and Efficient Parallel Preconditioners

- Approximate Block Factorizations
- Physics-based Preconditioners
- Multi-level solvers for systems and scalar equations

Preconditioning



Three variants of preconditioning

1. Domain Decomposition (Trilinos/Aztec & IFPack)

- ILU(k) Factorization on each processor (with overlap)
- High parallel efficiency, non-optimal algorithmic scalability

2. Multilevel methods: Trilinos/ML Package

Fully-coupled Algebraic Multilevel methods

- Consistent set of DOF at each node (stabilized FE)
- Aggregation rate chosen to fix coarse grid size
- Jacobi, GS, ILU(k) as smoothers
- Can provide optimal algorithmic scalability

Aggregation based Multigrid:
- Vanek, Mandel, Brezina, 1996
- Vanek, Brezina, Mandel, 2001
- Sala, Formaggia, 2001

3. Approximate Block Factorization / Physics-based: Trilinos/Teko package

- Applies to mixed interpolation (FE) or staggered (FV) discretization approaches
- Applied to systems where AMG is difficult or might fail
- Can provide optimal algorithmic scalability

Sandia National Laboratories

Brief Overview of Block Preconditioning Methods for Navier-Stokes:
(A Taxonomy based on Approximate Block Factorizations, JCP – 2008)

Discrete N-S	Exact LDU Factorization	Approx. LDU
$\begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} \Delta \mathbf{u}_k \\ \Delta p_k \end{pmatrix} = \begin{pmatrix} \mathbf{g}_\mathbf{u}^\mathbf{k} \\ g_p^k \end{pmatrix}$	$\begin{pmatrix} I & 0 \\ \hat{B}F^{-1} & I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & F^{-1}B^T \\ 0 & I \end{pmatrix}$ $S = C + \hat{B}F^{-1}B^T$	$\begin{bmatrix} I & 0 \\ \hat{B}H_1 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & -\hat{S} \end{bmatrix} \begin{bmatrix} I & H_2B^T \\ 0 & I \end{bmatrix}$

Precond. Type	H_1	H_2	$\hat{S}$	References
Pres. Proj; 1st Term Nuemann Series	$\mathbf{F}^{-1}$	$(\mathbf{\Delta t I})^{-1}$	$\mathbf{C} + \mathbf{\Delta t \hat{B} B^T}$	Chorin(1967);Temam (1969); Perot (1993): Quateroni et. al. (2000) as solvers
SIMPLEC	$\mathbf{F}^{-1}$	$(\mathbf{diag}(\sum \lvert\mathbf{F}\rvert))^{-1}$	$\mathbf{C} + \hat{\mathbf{B}}(\mathbf{diag}(\sum \lvert\mathbf{F}\rvert))^{-1}\mathbf{B^T}$	Patankar et. al. (1980) as solvers; Pernice and Tocci (2001) smothers/MG
Pressure Convection / Diffusion	0	$\mathbf{F}^{-1}$	$\mathbf{A_p F_p^{-1}}$	Kay, Loghin, Wathan, Silvester, Elman (1999 - 2006); Elman, Howle, S., Shuttleworth, Tuminaro (2003,2008)

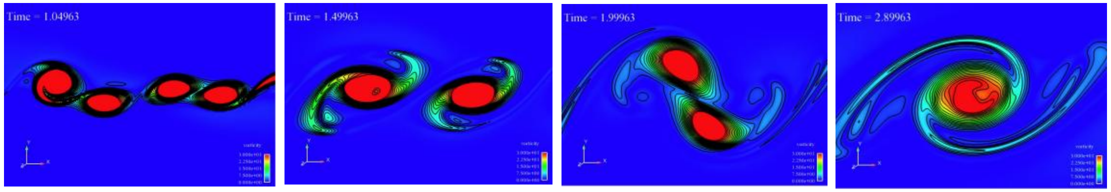
Now use AMG type methods on sub-problems.

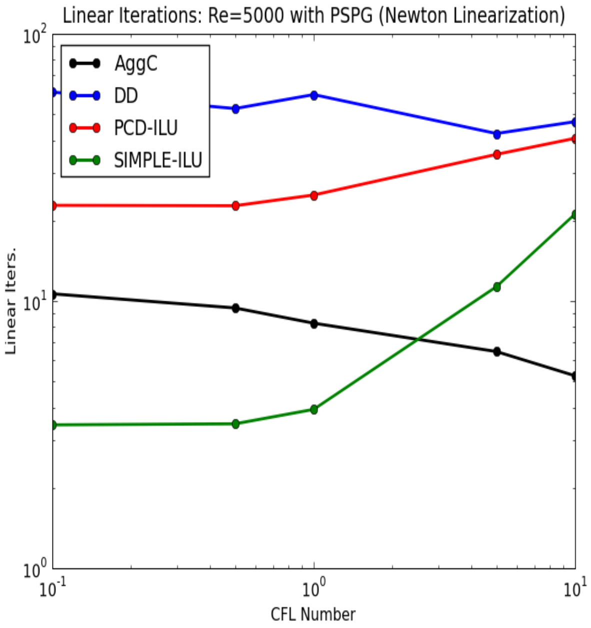
Momentum transient convection-diffusion: $F\Delta \mathbf{u} = \mathbf{r_u}$

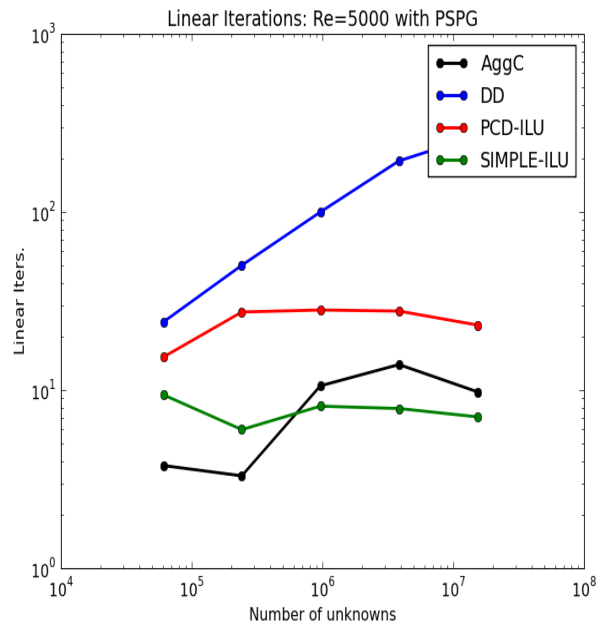
Pressure – Poisson type: $-\hat{S}\Delta p = \mathbf{r}_p$

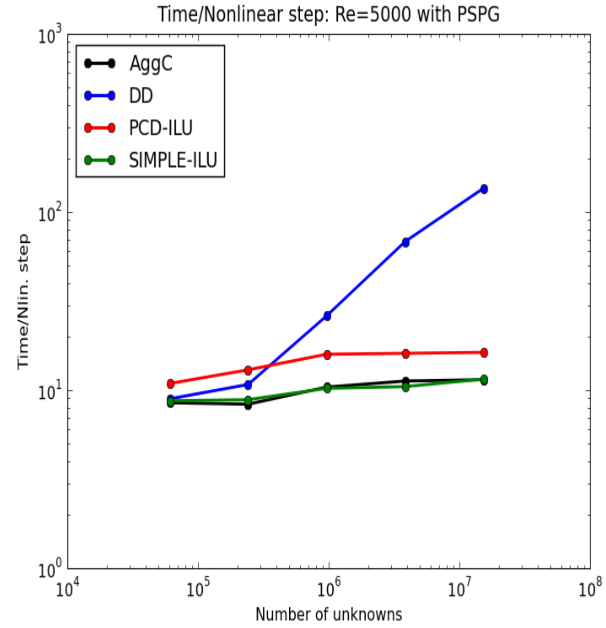
Sandia National Laboratories

Transient Kelvin-Helmholtz









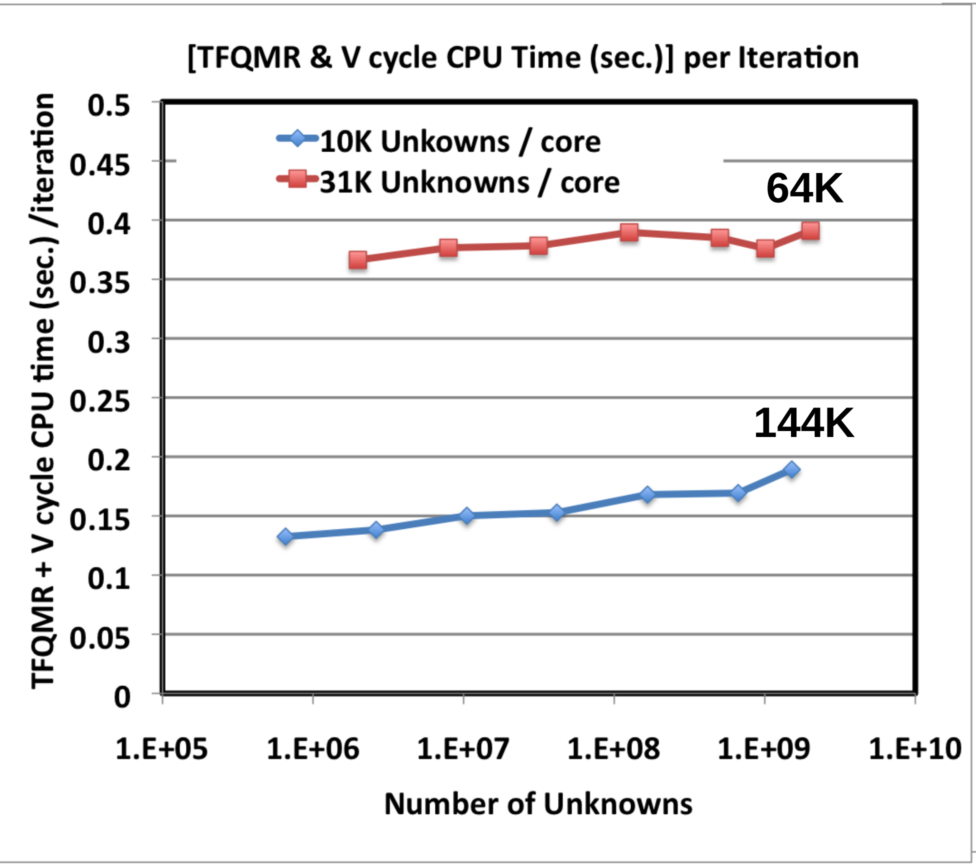
Kelvin Helmholtz: Re=5000, Weak scaling at CFL=2.5

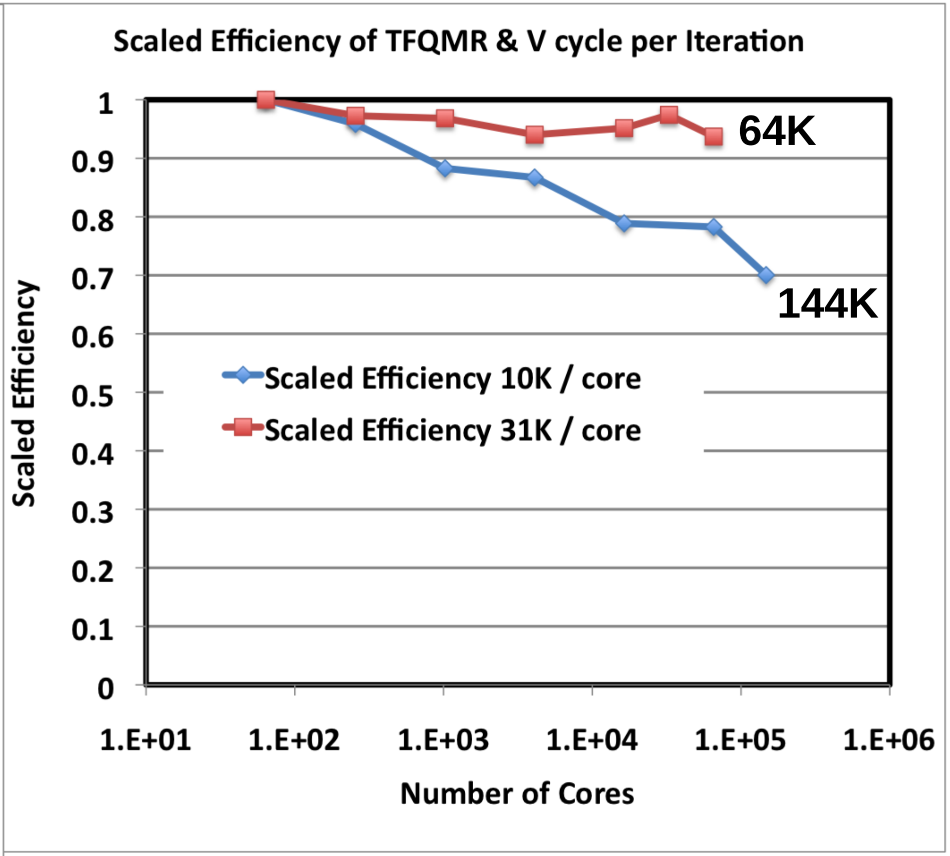
- **Run on 1 to 256cores**
- **Pressure - PSPG, Velocity - SUPG (residual and Jacobian)**

1. **SIMPLEC strongly dependent on CFL**
2. **Block methods scale as well as AggC and do not require non-zero C matrix**

Sandia National Laboratories

Weak Scaling Uncoupled Aggregation Scheme: Time/iteration on BlueGene/P (Drift – Diffusion BJT: P. Lin)





- TFQMR: used to look at time/iteration of multilevel preconditioners.
- W-cyc time/iteration not doing well due to significant increase in work on coarse levels (not shown)
- Good scaled efficiency for large-scale problems on larger core counts for 31K Unknowns / core

Sandia National Laboratories

Linear and Eigen Solvers

- We use the Stratimikos package for linear solvers
  - ParameterList driven assembly of all linear solvers and eigensolvers in Trilinos
    - AztecOO, Belos, Amesos
    - "Linear Operator with Solve" (LOWS)
    - Preconditioner support included (Ifpack, ML, Teko, …)
  - Assembles Thyra objects:
- Anasazi (Eigensolver) is accessed
  - Indirectly via LOCA stability analysis

Building a linear solver is this simple!

```
Stratimikos::DefaultLinearSolverBuilder linearSolverBuilder;

linearSolverBuilder.setParameterList(strat_params);

RCP<Thyra::LinearOpWithSolveFactoryBase<double> > lowsFactory =
    createLinearSolveStrategy(linearSolverBuilder);

RCP<Thyra::ModelEvaluatorDefaultBase<double> > thyra_me =
  Thyra::epetraModelEvaluator(ep_me,lowsFactory);
```

Sandia National Laboratories

Linear Algebra Layer

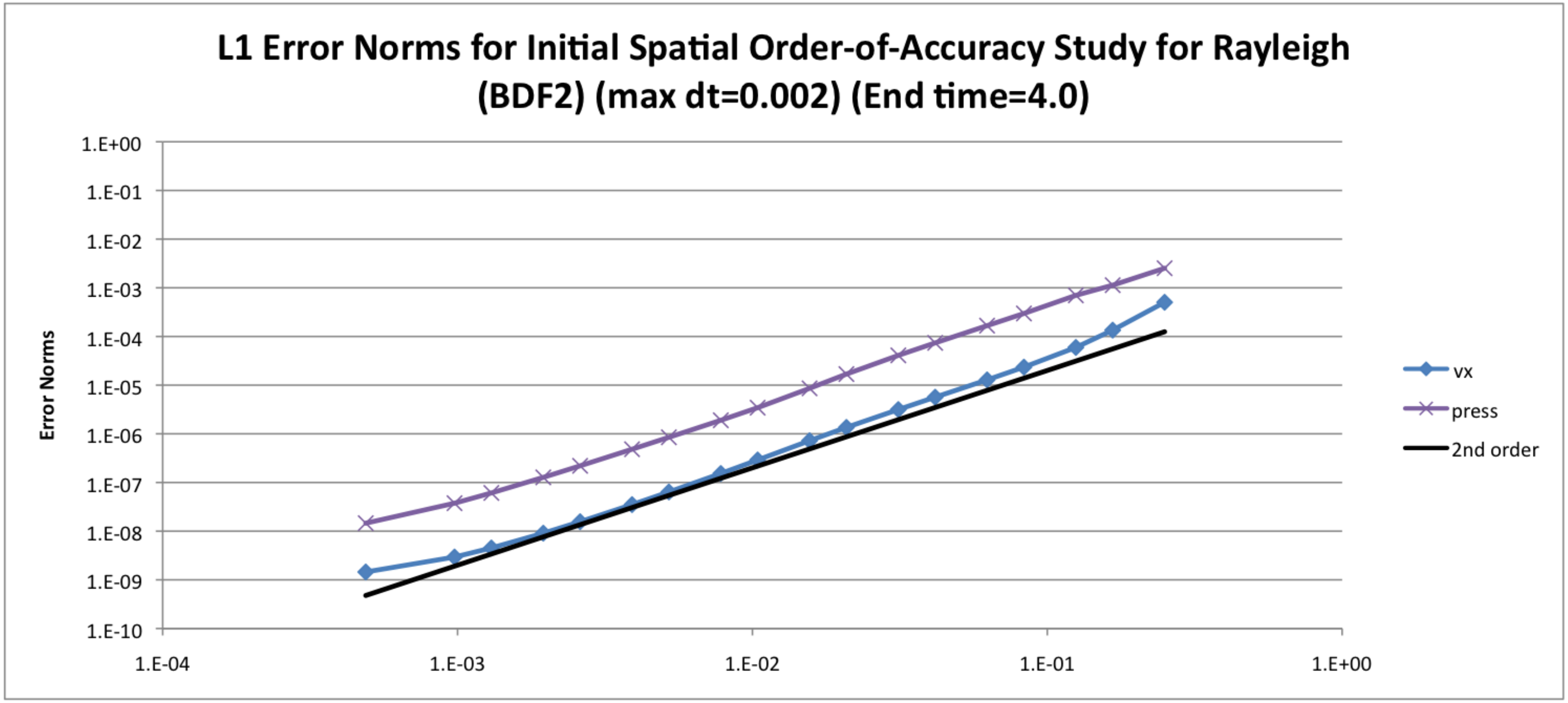
- Pass Thyra vector and operator objects throughout code
  - At the lowest level we cast to concrete type to fill object
  - Plan to use Tpetra, but not all pieces are present in the “2nd Generation” stack
    - MueLu
    - NOX/LOCA
  - For now we use Epetra for concrete LA type

Mesh Database and I/O

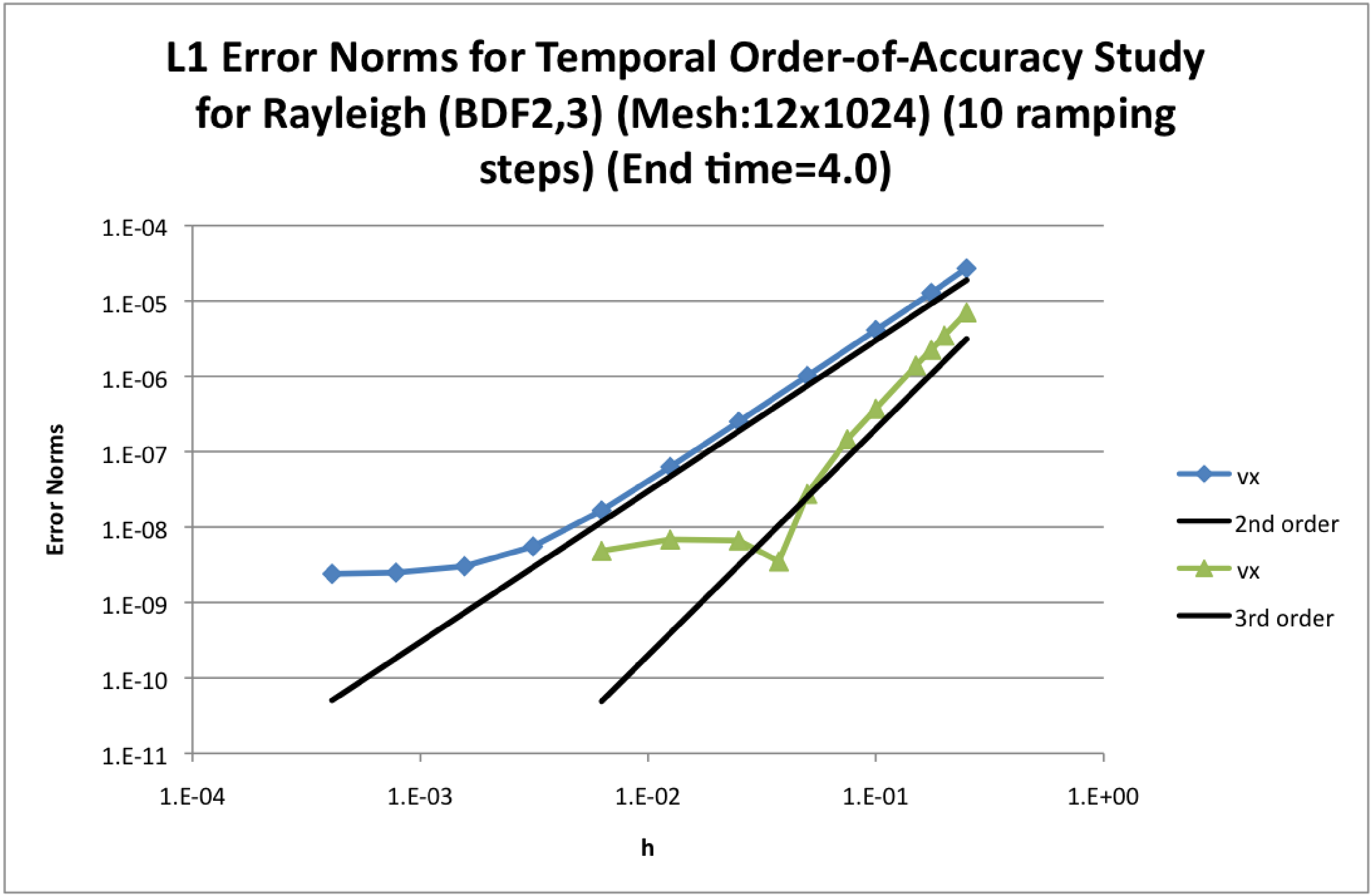
- Any mesh database or I/O format can be used, must implement a Panzer::ConnectionManager to use with the assembly engine
- Panzer conatins a concrete Implementation to **STK::mesh**
  - A ConnectionManager implementation to STK for DOF mapping
- SEACAS
  - I/O uses Exodus format
  - Apps are now in Trilinos (exodiff)
- Panzer provides Observers for NOX and Rythmos to write Exodus file on triggered events (e.g. successful time steps)

Results

Spatial Error Verification



Temporal Error Verification

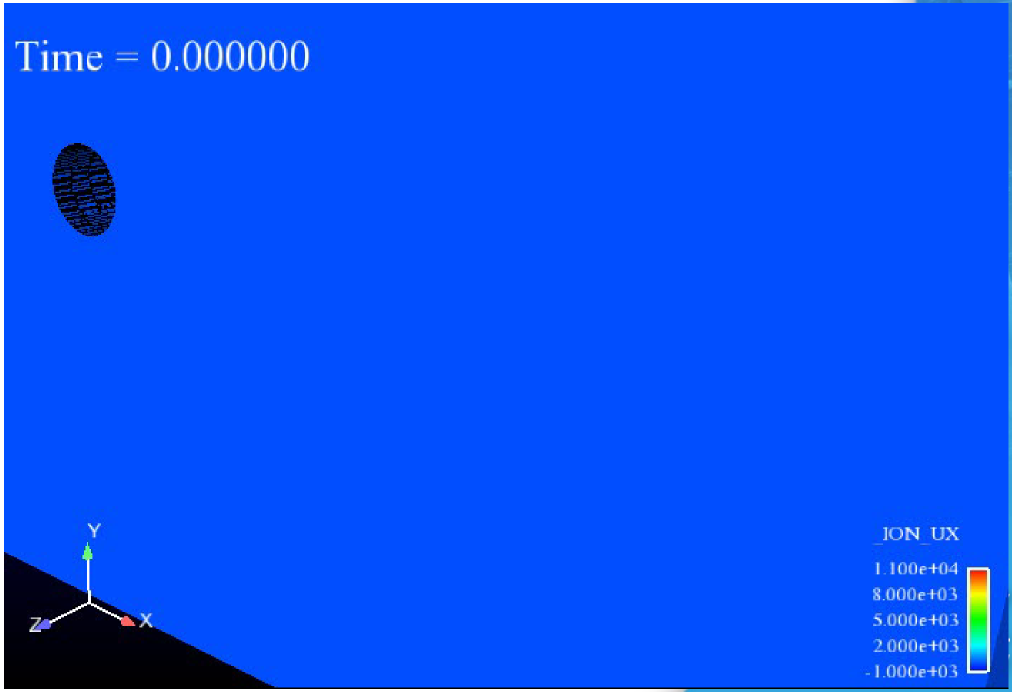


Sandia National Laboratories

VUQ Coordination for solution verification and validation (with B. Rider)

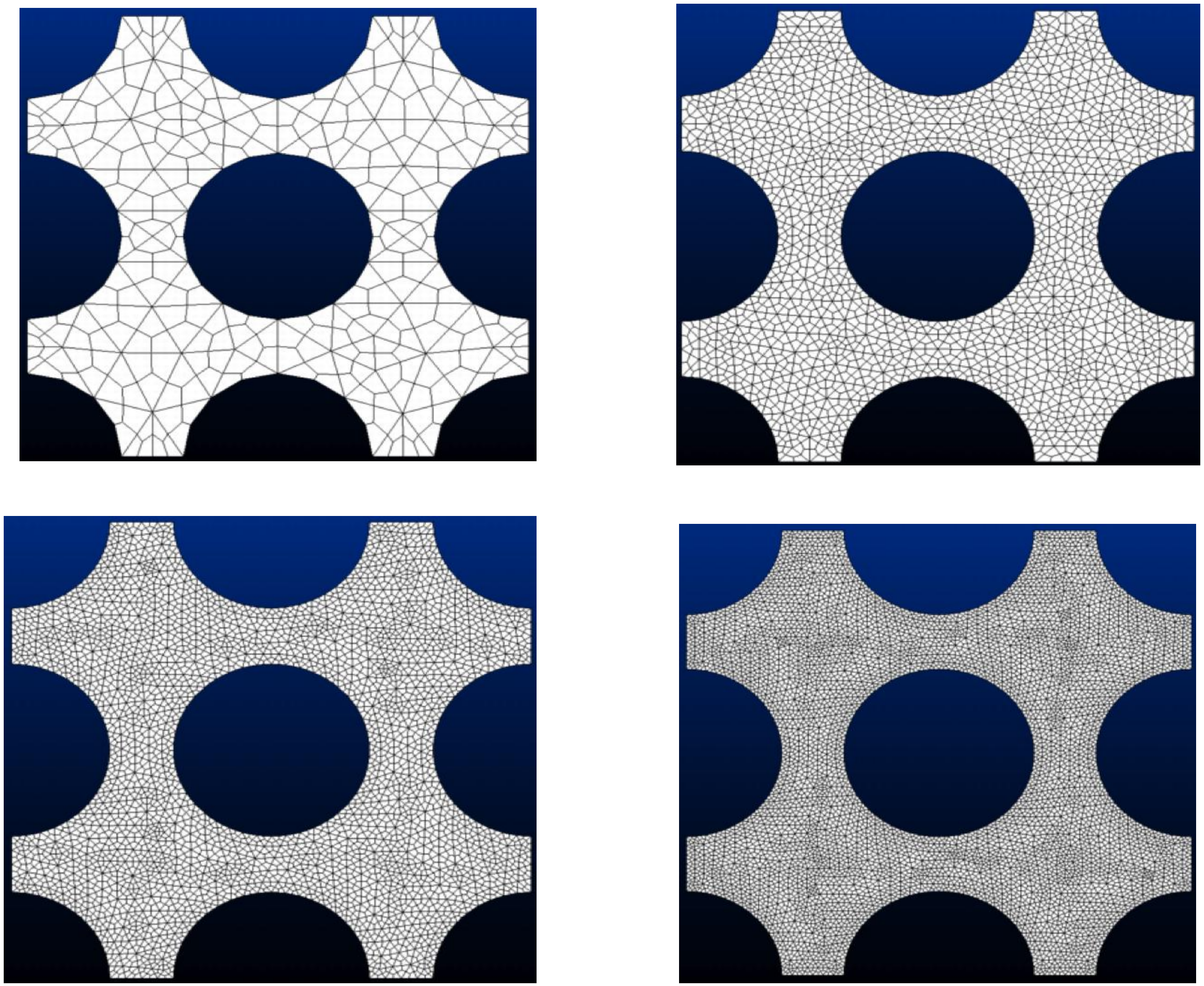
Vortex shedding over cylinder

Method	Experimental Data	Salsa Data	Drekar Data
Re=60	$.136 \pm .006$	0.132	$.134 \pm .004$
Re=100	$.164 \pm .005$	0.163	$.163 \pm .005$
Re=200	$.185 \pm .007$	0.189	$.189 \pm .004$
Re=400	$.205 \pm .005$		$.209 \pm .005$
Re=600	$.210 \pm .006$	0.218	$.218 \pm .006$
Re=1000	$.212 \pm .006$		$.222 \pm .007$
Re=5000	$.212 \pm .006$		$.220 \pm .010$



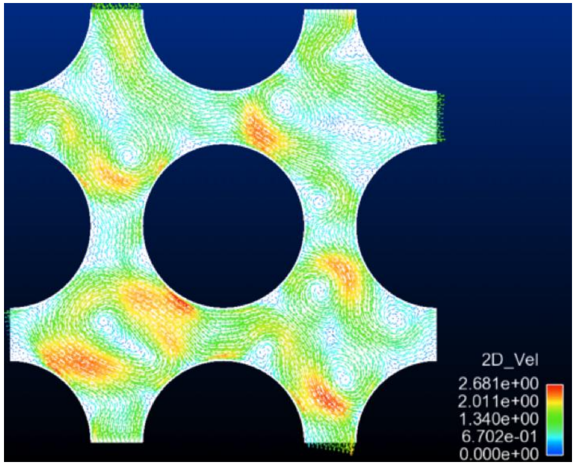
Verification on Multiple Meshes

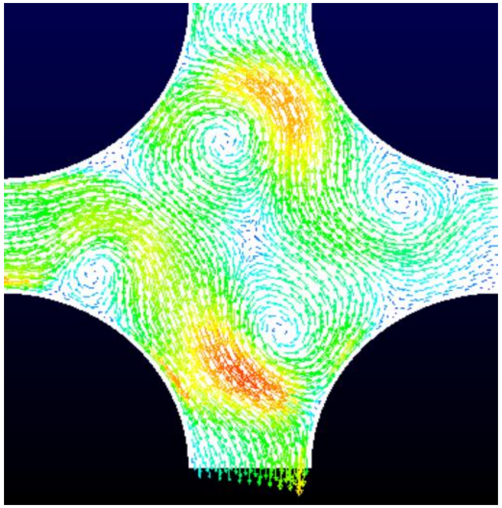
(Hex mesh ranging from 700K to 6M elements)

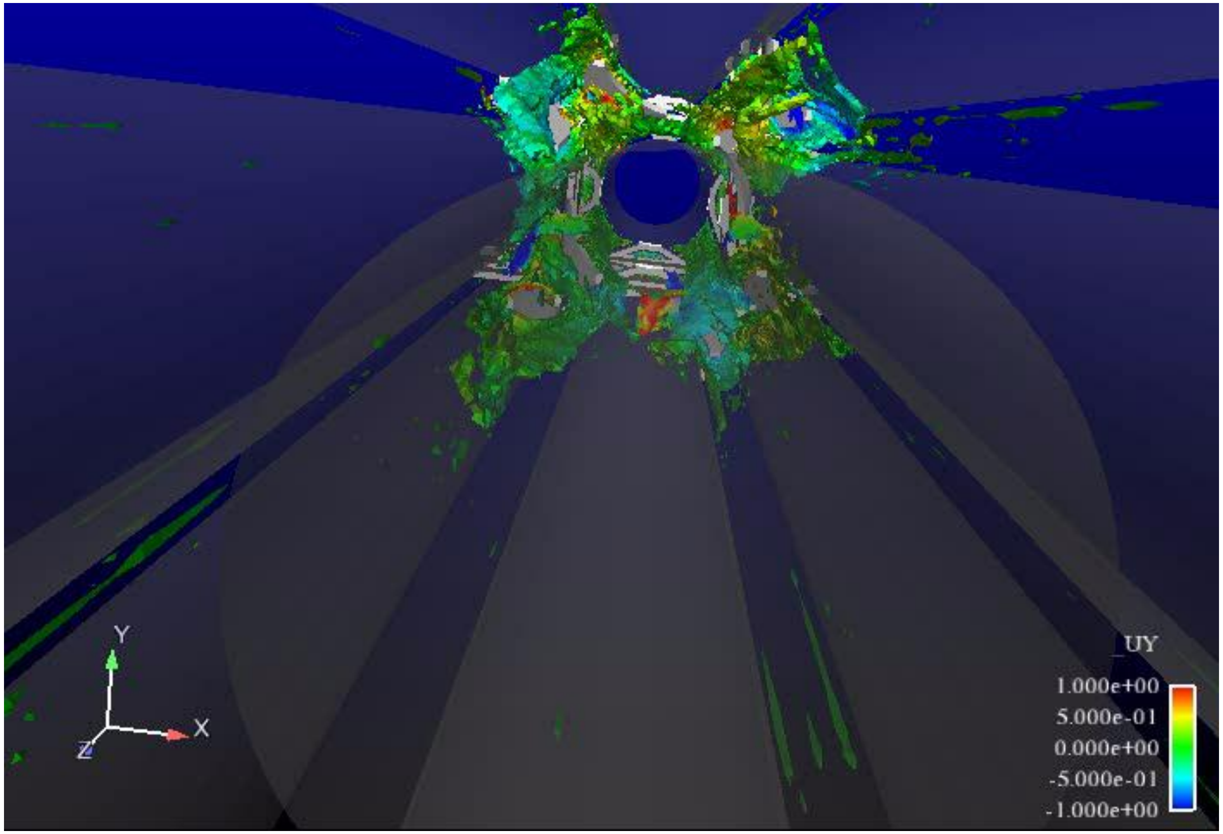


Solution Profiles

- Sandia Redsky platform, 256-1024 processes
- Oak Ridge Jaguar platform, 1200-9600 processes
- 2^{nd} order BFD2 time integration
- Linear Lagrange elements (2^{nd} order in space)



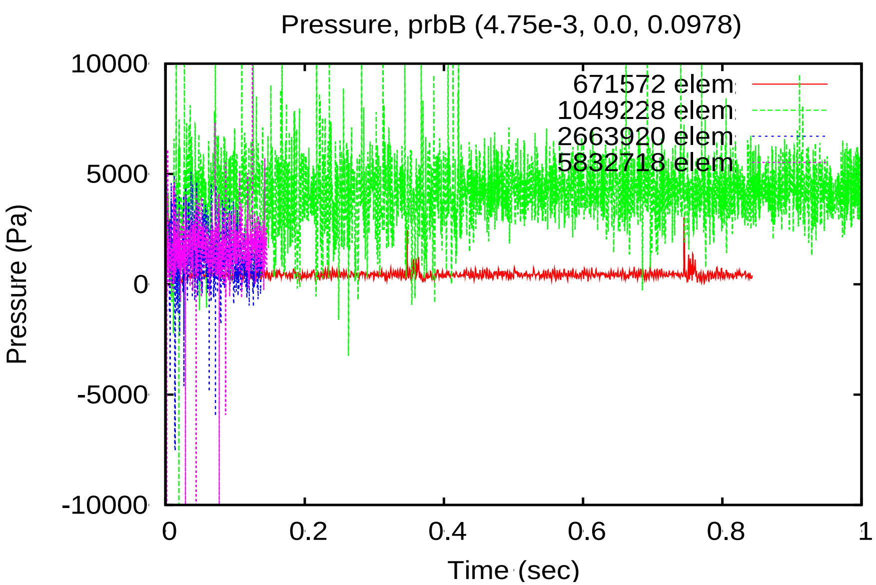


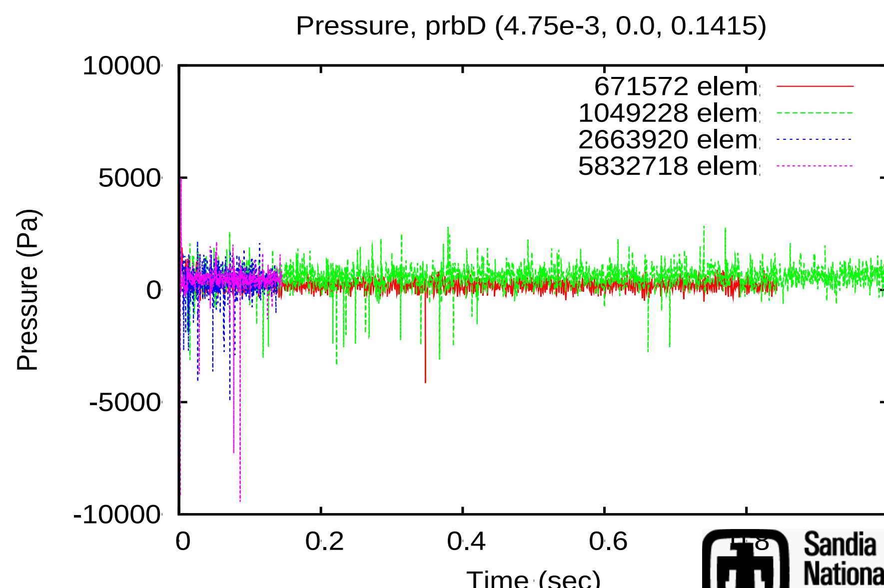


Sandia National Laboratories

Transient Pressure Profiles

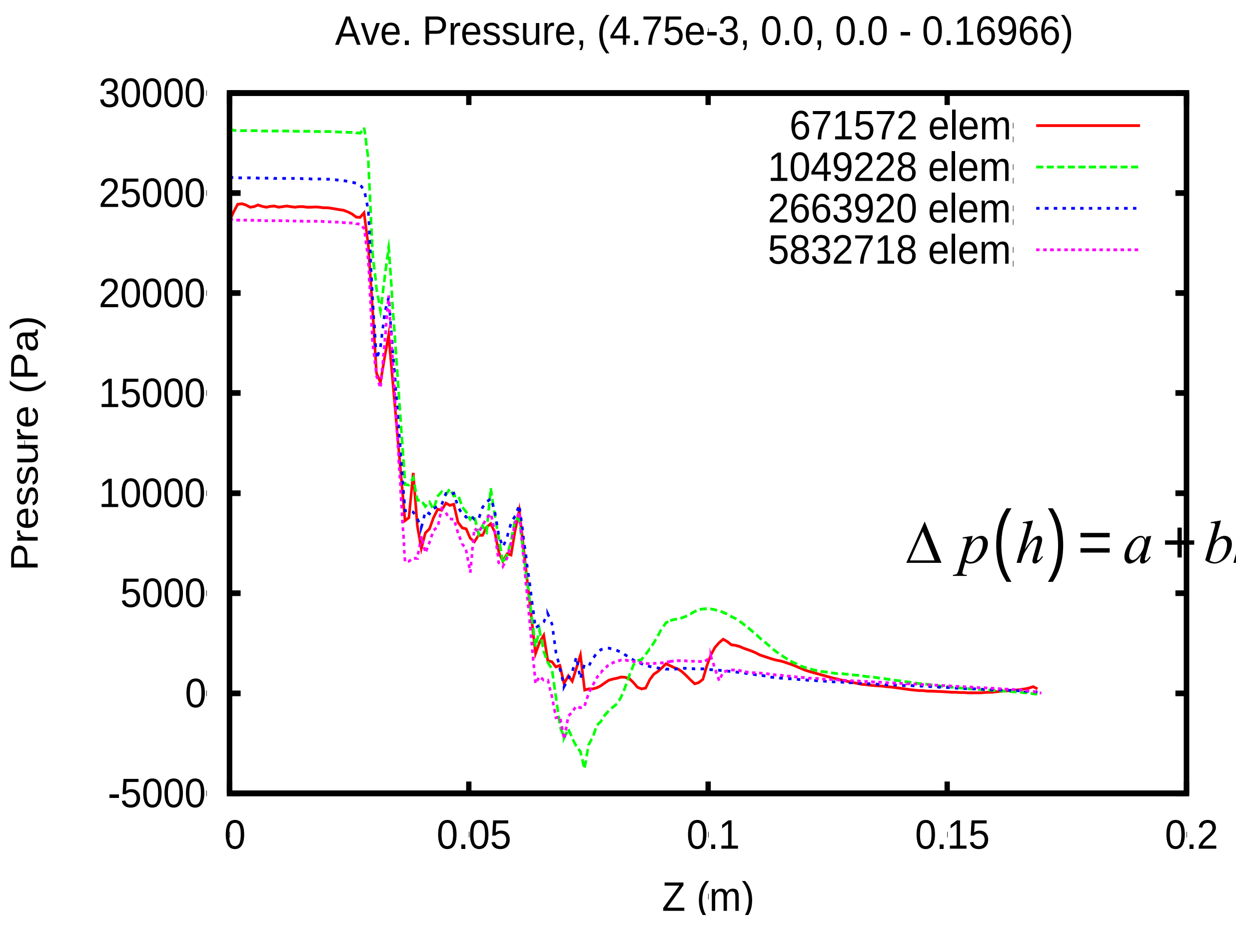
Mesh	# of Elements	Simulated Time (sec.)	Flow through times	Δt	CFL ave. (section1/section3)	Y+ ave.
672K (Red)	671,572	0.88	26	5.0×10^{-5}	9.6/1.3	26
1M (Green)	1,049,288	1.11	33	5.0×10^{-5}	11.1/2.9	19
3M (Blue)	2,663,920	0.2	6	2.0×10^{-5}	5.9/2.1	16
6M (Purple)	5,832,718	0.32	9	2.0×10^{-5}	7.9/2.2	13





Time Averaged Pressure Profiles Across Spacer Grid

- Coarsest mesh is inadequate for simulation
- 3 finer meshes show signs of convergence

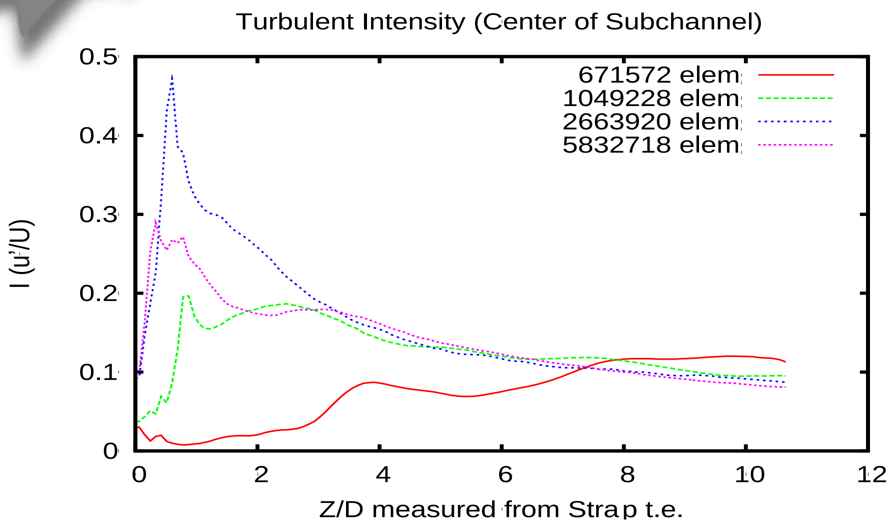


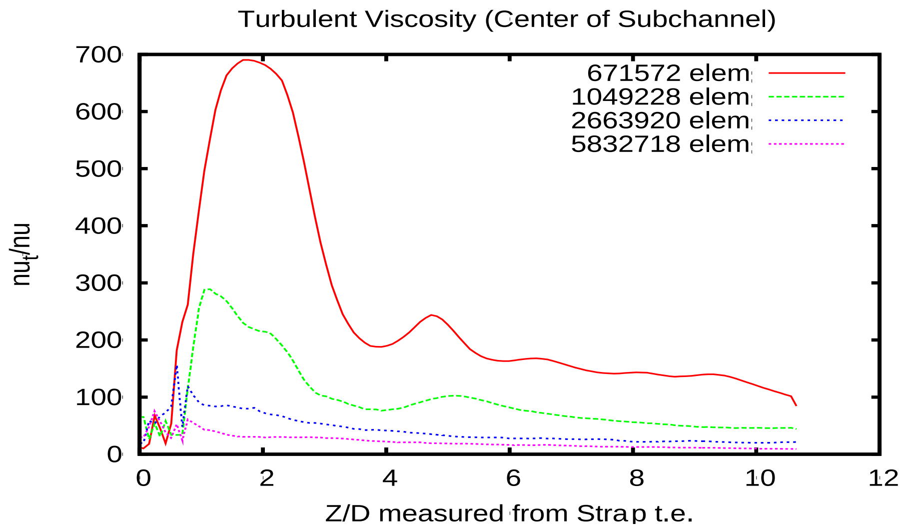
Mesh	Mean Pressure Drop (Pa)
677K	23,400
1M	26,780
3M	23,800
6M	22,042

$$\Delta p(h) = a + bh^c = 17.42 - 163.7h^{1.234}$$

- **Pressure drop value**
- **Coarse mesh solutions not yet in asymptotic range**
- **Already > 1st order!**

Turbulence Characteristics

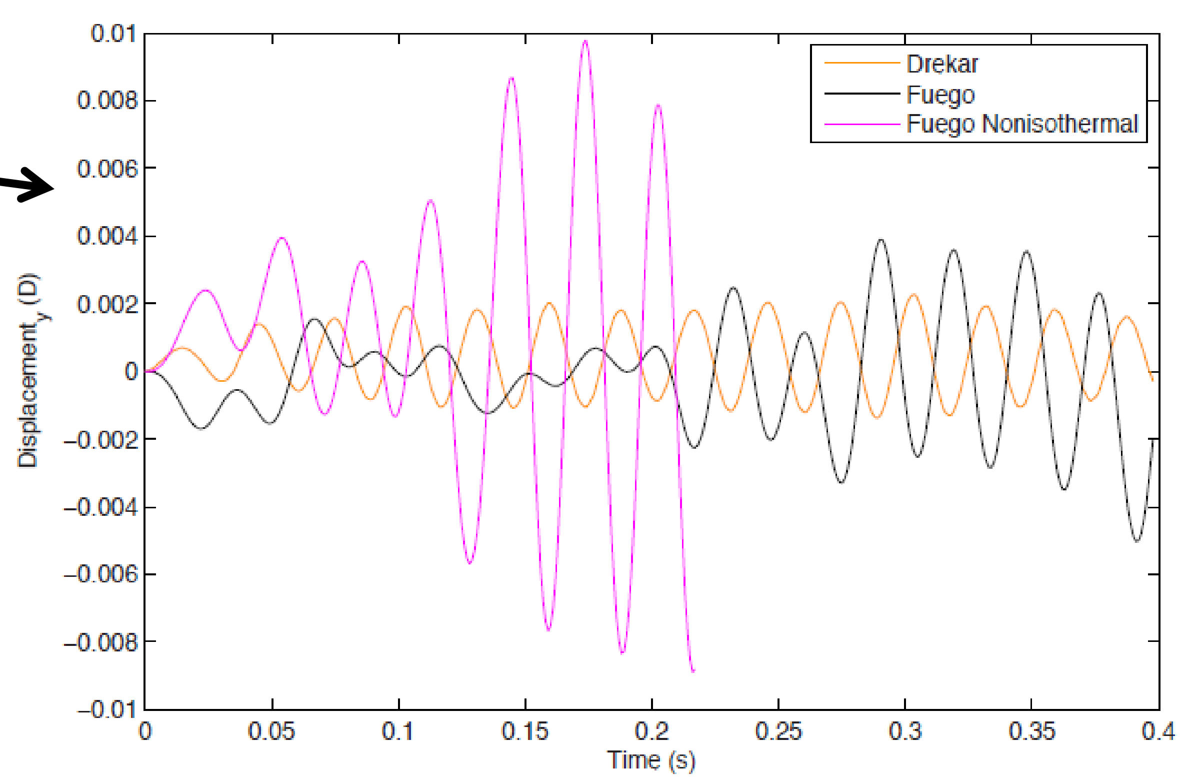


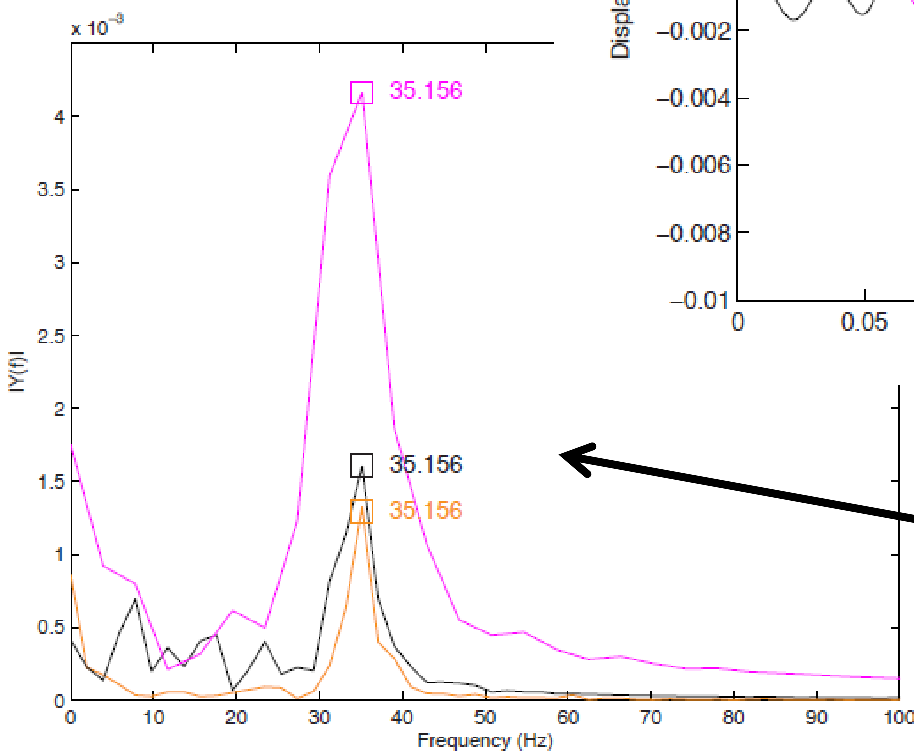


- 672K mesh is inadequate to resolve behavior near spacer grid.
- Non-monotonic behavior in turbulent intensity:
  - Converges 3 diameters downstream as turbulence subsides
  - Suggests under-resolution of mesh near spacer grid
- Turbulent intensity comparison to Benhamadouche *et al.* 2009:
  - At 3 diameters downstream, Drekar: 18%, Ben.: 11%
  - At 10 diameters downstream, Drekar: 10%, Ben.: 8.5%
- Monotonic decrease in amount of eddy-viscosity with refined mesh shows correct behavior (reduced contribution of subgrid model).

Demo of Forcing of Structural Vibration (Sierra/Presto). Solid Zircaloy Rod

Rod displacement in y-direction produced by fluid loads





FFT on displacement yields same frequency.

Impact

- TBGP w/ Phalanx and Panzer allows for rapid development and integration of advanced physics (Using Agile Components)
  - **Drekar went from drawing board to milestone completion in 6 months**
  - **Implementation + verification + validation + production runs!**
- New capabilities to propel ASCR research:
  - Multiphysics: Mixed equation sets (e.g. conjugate heat transfer)
  - Mixed basis and higher order methods (compatible disc. For MHD)
- Drekar has been adopted as one of two TH/CFD codes for CASL.
  - Integrated into VERA simulation suite.
  - First official release to CASL in FY12/Q2
- Community Adoption:
  - ASC/QASPR: Next-generation Charon now under development using Panzer (Suzey Gao)
  - Phalanx is the backbone of FE assembly in Albany(~10 different physics).
- Being considered for “Joule metric” code (ASCR OMB metric)

Sandia National Laboratories

A Users Perspective

- Trilinos is not a uniform code base:
  - Many developers with varying opinions on software quality
  - Good documentation vs good tests (sometimes both or neither)
- Confusion in a package maturity level:
  - PS and SS do not mean **hardened** code! Is there a better way to describe a code's status?
  - Additional capabilities required: PIRO and Rythmos
- Pushing new interfaces/capabilities through Thyra layers can be difficult.
  - The answer can't always be go to Ross!
- Would not have advanced this far this fast without expertise that spans a large part of Trilinos.
  - High entry bar to Trilinos

Conclusions

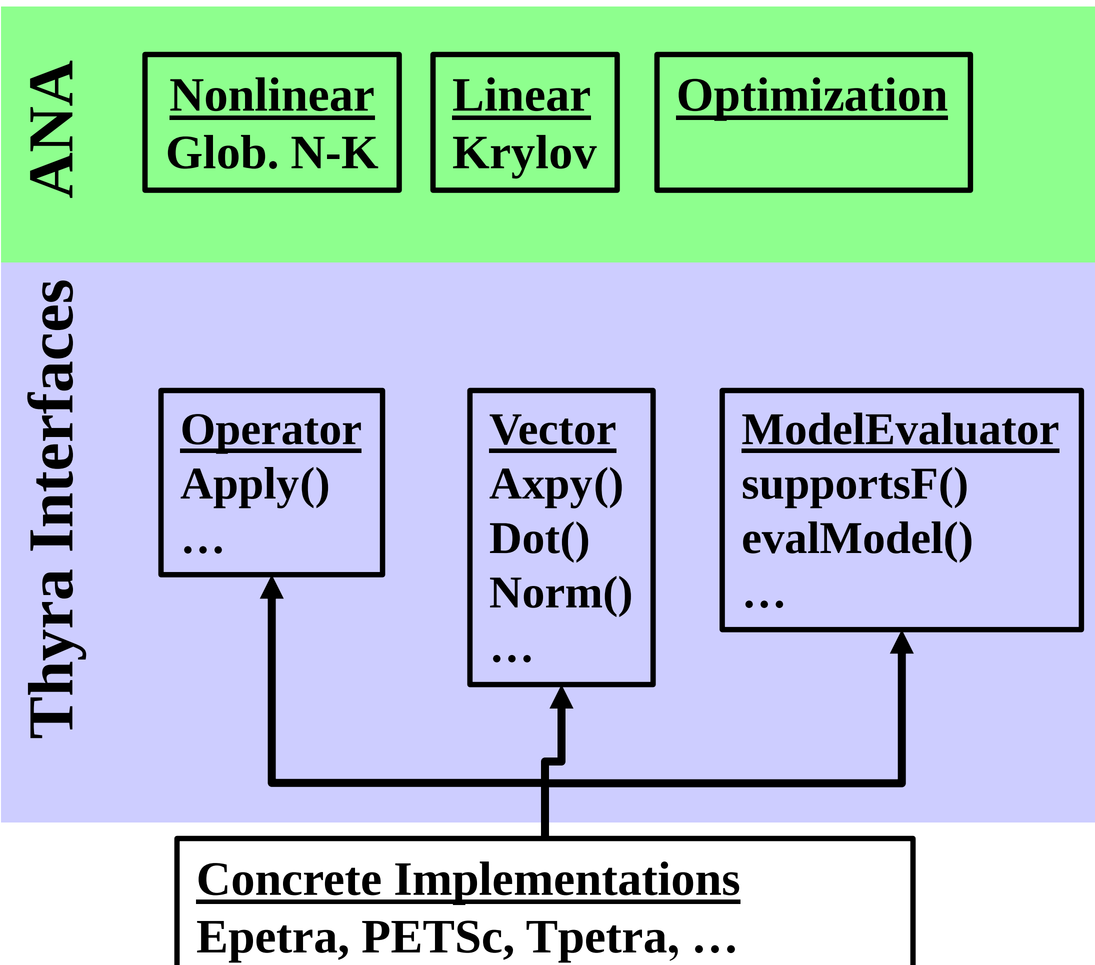
- Successfully stood up a new code in 6 months!
- Leveraged a significant portion of Trilinos
- Not all packages fit perfectly, but it is preferable to rolling your own (technical debt)
- Lightweight front end physics description
- Decoupled from solution/analysis procedures

Extra Slides

Introducing Abstract Numerical Algorithms

What is an abstract numerical algorithm (ANA)?

An ANA is a numerical algorithm that can be expressed abstractly solely in terms of vectors, vector spaces, linear operators, and other abstractions built on top of these *without general direct data access or any general assumptions about data locality*



Block composition operators and vectors:

$$\begin{bmatrix} \mathbf{J}_{TT} & \mathbf{J}_{T\mathrm{W}} \\ \mathbf{J}_{\mathrm{W}T} & \mathbf{J}_{\mathrm{WW}} \end{bmatrix} \begin{bmatrix} \Delta T \\ \Delta \mathrm{W} \end{bmatrix} = - \begin{bmatrix} \mathbf{F}_T \\ \mathbf{F}_{\mathrm{W}} \end{bmatrix}$$

Block Factorization Preconditioners:

$$\begin{bmatrix} I & 0 \\ \hat{B}H_1 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & -\hat{S} \end{bmatrix} \begin{bmatrix} I & H_2 B^T \\ 0 & I \end{bmatrix}$$

$$S = c + \hat{B} F^{-1} B^T$$

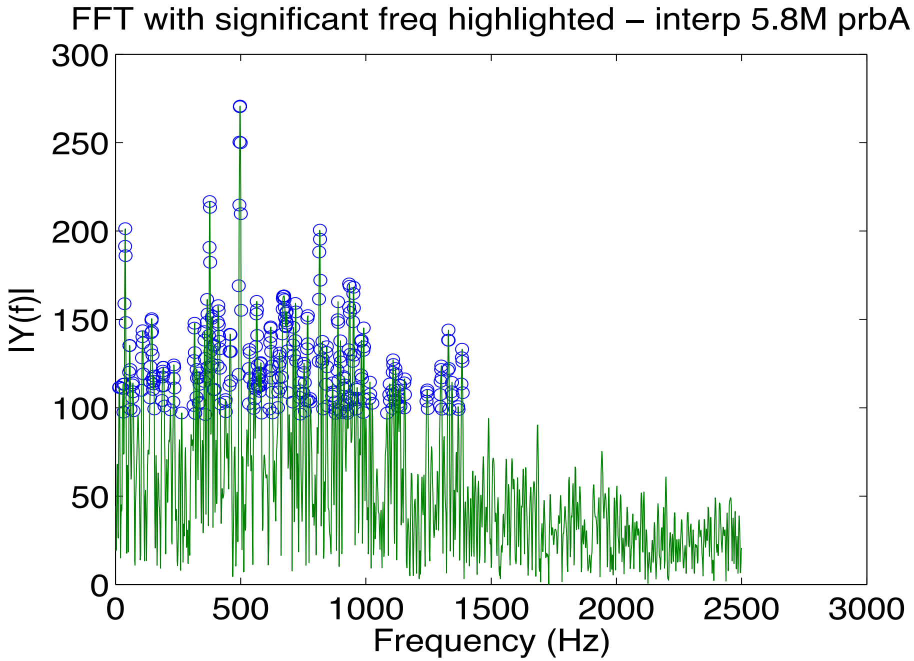
Wish List

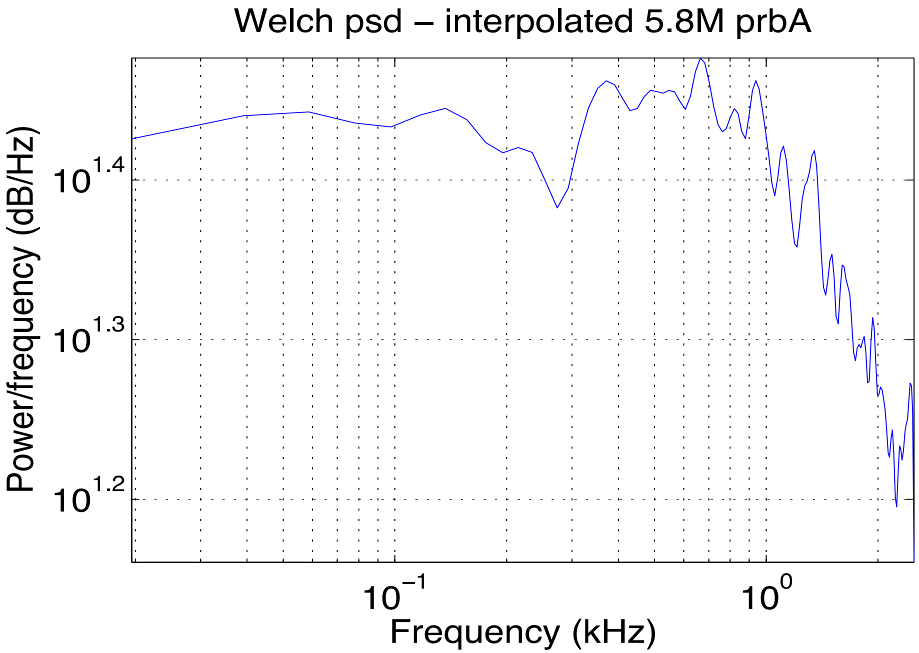
- Order preserving ParameterList xml parser
- ParameterList Validators for our use case
- Improved support for Thyra

- Rythmos support for better startup and ramping
- Intrepid:
  - 0D quadratures need to be supported for 1D code!
  - Cylindrical coordinates (3D physics on a 2D mesh)
- Improved TPL support: versioning!
  - STK, ML, Zoltan: Parmetis 3 → 4

- When code should be made into a general Trilinos package?

FFT and PSD on Center Rod





- Maximum frequency of the FFT is roughly the Nyquist frequency.
- Characteristic turbulent cascade is evident in the PSD (just starting to analyze this).

Drekar::CFD

trilinos.sandia.gov

- **Massively Parallel: MPI**
- **2D & 3D Unstructured Stabilized FE**
- **Fully Coupled Globalized Newton-Krylov solver**
  - **Sensitivities: Template-based Generic Programminc for Automatic Differentiation (Sacado), UQ, Arb. Prec.**
  - **GMRES (AztecOO, Belos)**
  - **Additive Schwarz DD w/ Var. Overlap (Ifpack, AztecOO)**
  - **Aggressive Coarsening Graph Based Block Multi-level [AMG] for Systems (ML w/Amesos for coarse solve)**
  - **Physics-based/Block Factorization (Teko)**
- **Fully-implicit: 1st-5th variable order BDF (Rythmos) & TR**
- **Direct-to-Steady-State (NOX), Continuation, Linear Stability and Bifurcation (LOCA / Anasazi), PDE Constrained Optimization (Moocho)**