



Exploring Embedded UQ Approaches for Improved Scalability and Efficiency

Eric Phipps (etphipp@sandia.gov)
Sandia National Laboratories

Uncertainty Quantification in High Performance
Computing Workshop

May 2-4, 2012

SAND 2012-3582C



Outline

- **Challenges for embedded algorithms**
- **Approach to mitigating these challenges**
 - **Template-based generic programming**
 - **Agile components**
- **Embedded UQ R&D this enables**
 - **Gradient-enhanced sampling**
 - **Embedded sampling**
 - **Stochastic Galerkin**
 - **Multiphysics UQ**
- **Diverse set of**
 - **People**
 - **Projects**
 - **Funding sources**
- **Embedded UQ R&D primarily funded by ASCR Multiscale Math**
 - **Joint project between Sandia and USC (Roger Ghanem)**



Forward UQ

- **UQ means many things**
 - **Best estimate + uncertainty**
 - **Model validation**
 - **Model calibration**
 - **Reliability analysis**
 - **Robust design/optimization**
 - ...
- **A key to many UQ tasks is forward uncertainty propagation**
 - **Given uncertainty model of input data (aleatory, epistemic, ...)**
 - **Propagate uncertainty to output quantities of interest**
- **Key challenges:**
 - **Achieving good accuracy**
 - **High dimensional uncertain spaces**
 - **Expensive forward simulations**



Can These Challenges be (Partially) Met by Embedded Methods?

- **Embedded algorithms leverage simulation structure**
 - **Adjoint sensitivities/error estimates**
 - **Derivative-based optimization/stability analysis**
 - **Stochastic Galerkin or adjoint-based UQ methods**
 - ...
- **Many kinds of quantities required**
 - **State and parameter derivatives**
 - **Various forms of second derivatives**
 - **Polynomial chaos expansions**
 - ...
- **Incorporating directly requires significant effort**
 - **Developers must understand algorithmic requirements**
 - **Limits embedded algorithm R&D and its impact**



A solution

- **Need a framework that**
 - **Allows simulation code developers to focus on complex physics development**
 - **Doesn't make them worry about advanced analysis**
 - **Allows derivatives and other quantities to be easily extracted**
 - **Is extensible to future embedded algorithm requirements**
- **Template-based generic programming**
 - **Code developers write physics code templated on scalar type**
 - **Operator overloading libraries provide tools to propagate needed embedded quantities**
 - **Libraries connect these quantities to embedded solver/analysis tools**
- **Foundation for this approach lies with Automatic Differentiation (AD)**

What is Automatic Differentiation (AD)?

- **Technique to compute analytic derivatives without hand-coding the derivative computation**
- **How does it work -- freshman calculus**
 - **Computations are composition of simple operations (+, *, sin(), etc...) with known derivatives**
 - **Derivatives computed line-by-line, combined via chain rule**
- **Derivatives accurate as original computation**
 - **No finite-difference truncation errors**
- **Provides analytic derivatives without the time and effort of hand-coding them**

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

$$x \leftarrow 2$$

$$t \leftarrow e^x$$

$$u \leftarrow \log x$$

$$v \leftarrow xu$$

$$w \leftarrow t + v$$

$$y \leftarrow \sin w$$

x	$\frac{d}{dx}$
2.000	1.000
7.389	7.389
0.301	0.500
0.602	1.301
7.991	8.690
0.991	-1.188



Sacado: AD Tools for C++ Codes

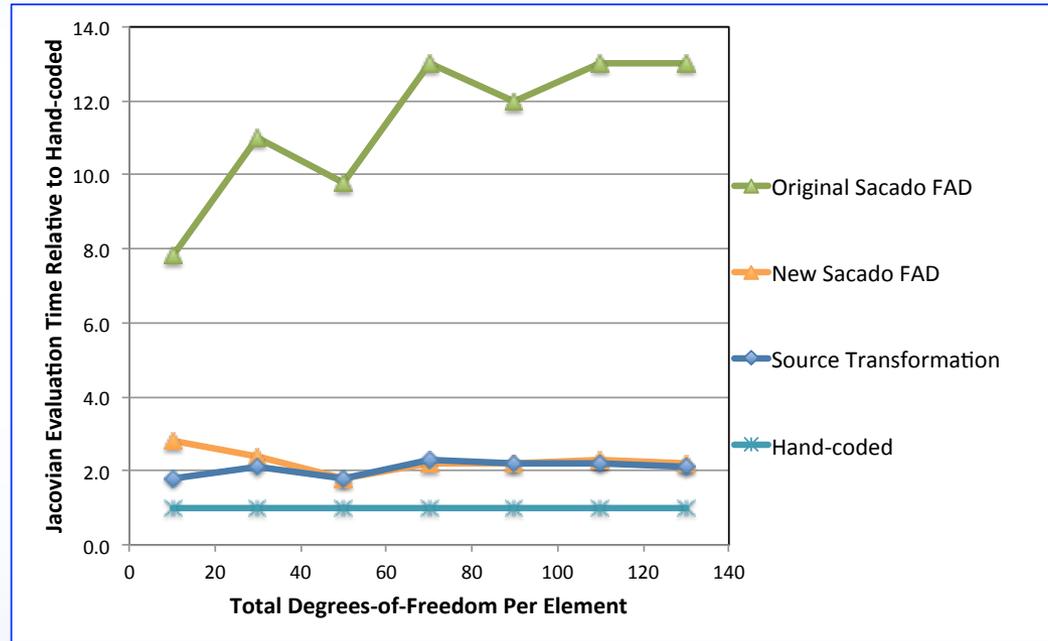
- **Several modes of Automatic Differentiation**
 - Forward
 - Reverse
 - Univariate Taylor series
 - Modes can be nested for various forms of higher derivatives
- **Sacado uses operator overloading-based approach for C++ codes**
 - Phipps, Gay (SNL ASC)
 - Sacado provides C++ data type for each AD mode
 - Replace scalar type (e.g., double) with template parameter
 - Instantiate template code on various Sacado AD types
 - Mathematical operations replaced by overloaded versions
 - Expression templates to reduce overhead



<http://trilinos.sandia.gov>



Our AD Tools Perform Extremely Well



- **Simple set of representative PDEs**
 - **Total degrees-of-freedom = number of nodes x number of PDEs for each element**
- **Operator overloading overhead is nearly zero**
- **2x cost relative to hand-coded, optimized Jacobian (very problem dependent)**

AD to TBGP

- **Benefits of templating**
 - Developers only develop, maintain, test one templated code base
 - Developers don't have to worry about what the scalar type really is
 - Easy to incorporate new scalar types
- **Templates provide a deep interface into code**
 - Can use this interface for more than derivatives
 - Any calculation that can be implemented in an operation-by-operation fashion will work
- **We call this extension Template-Based Generic Programming (TBGP)**
 - **Extended precision**
 - Shadow double
 - Floating point counts
 - Logical sparsity
 - **Uncertainty propagation**
 - Intrusive stochastic Galerkin/polynomial chaos
 - Simultaneous ensemble propagation
 - 2 papers under revision to *Jou. Sci. Prog.*



Sandia Agile Components Strategy

- **Create, use, and improve a common base of software components**
 - **Component = modular, independent yet interoperable software**
 - Libraries, interfaces, software quality tools, demo applications
 - **Leverage software base for new efforts**
 - **Use new efforts to improve software base**
 - **Path to impact for basic research (e.g., ASCR)**
- **Large effort encapsulating much of SNL CIS R&D**
 - **Driven by Andy Salinger**



- **Foundation for new simulation code efforts**
 - **E.g., Drekar for CASL (Pawlowski, Cyr, Shadid, Smith)**

The Components Effort is Large (~100 modular pieces)

Analysis Tools (black-box)

Optimization
UQ (sampling)
Parameter Studies
V&V, Calibration
OUU, Reliability

Analysis Tools (embedded)

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

Multi-Core
Accelerators

Composite Physics

MultiPhysics Coupling
System Models
System UQ

Mesh Tools

Mesh I/O
Inline Meshing
Partitioning
Load Balancing
Adaptivity
Remeshing
Grid Transfers
Quality Improvement
DOF map

Utilities

Input File Parser
Parameter List
Memory Management
I/O Management
Communicators

PostProcessing

Visualization
Verification
Model Reduction

Mesh Database

Mesh Database
Geometry Database
Solution Database

Data-Centric Algs

Graph Algorithms
SVDs
Map-Reduce
Linear Programming
Network Models

Local Fill

Discretizations

Discretization Library
Field Manager

Derivative Tools

Sensitivities
Derivatives
Adjoints
UQ / PCE Propagation

Physics Fill

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

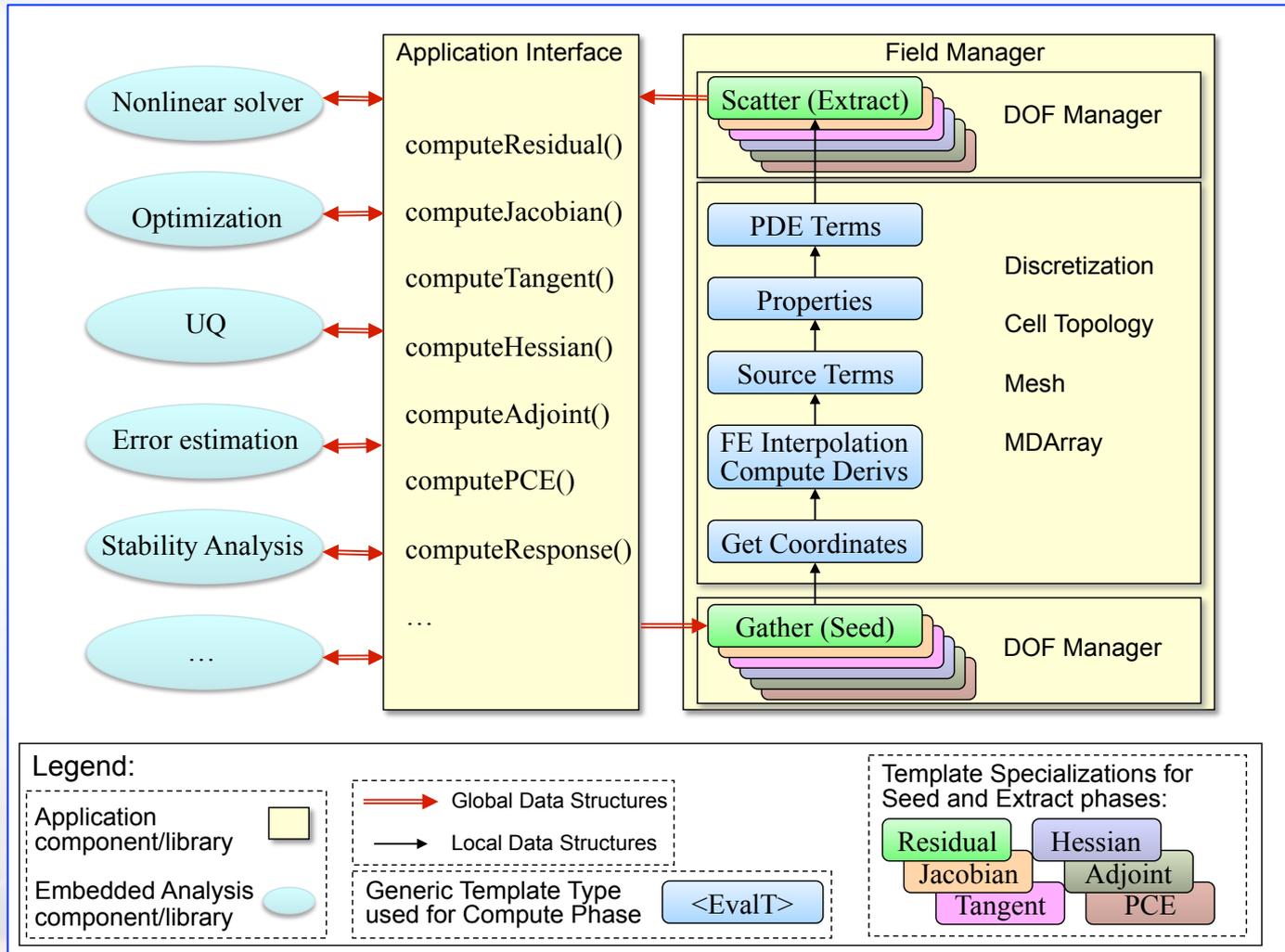
Software Quality

Version Control
Regression Testing
Build System
Backups
Verification Tests
Mailing Lists
Unit Testing
Bug Tracking
Performance Testing
Code Coverage
Porting
Web Pages
Release Process

Templating is a Key Component Software Design Principal

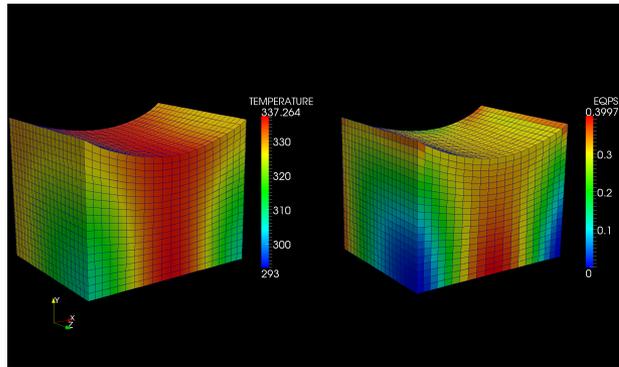
- **C++ templating is a powerful method of abstraction**
 - **STL containers**
 - **Boost MPL**
 - **Trilinos Kokkos node API**
- **Templating components on the scalar type provides API to support embedded algorithms**
 - **Developers focus on component implementation**
 - **Embedded quantities are easily incorporated**
 - **Scalable to many embedded techniques**

Templated Components Orthogonalize Physics and Embedded Algorithm R&D

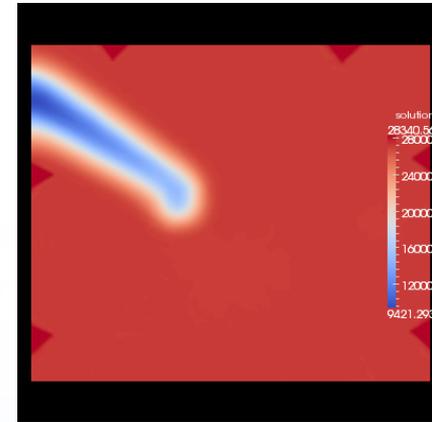


Approach Supports Complex Physics Development

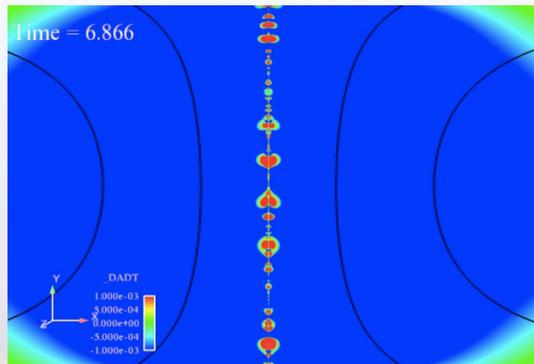
Albany/LCM – Thermo-Elasto-Plasticity
– J. Ostein et al



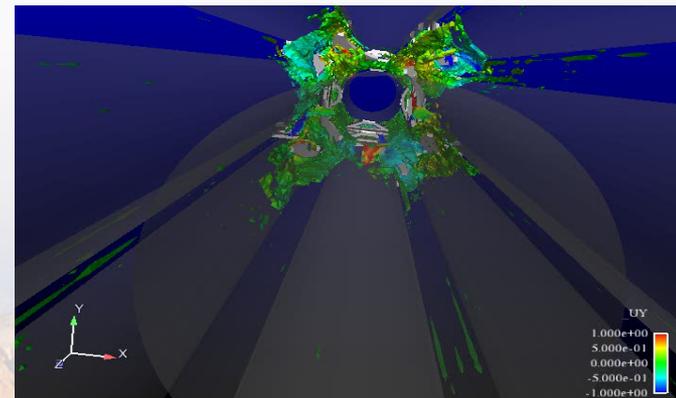
Albany/QCAD – Quantum Device Modeling
– R. Muller et al



Charon/MHD – Magnetic Island Coalescence
– Shadid, Pawlowski, Cyr

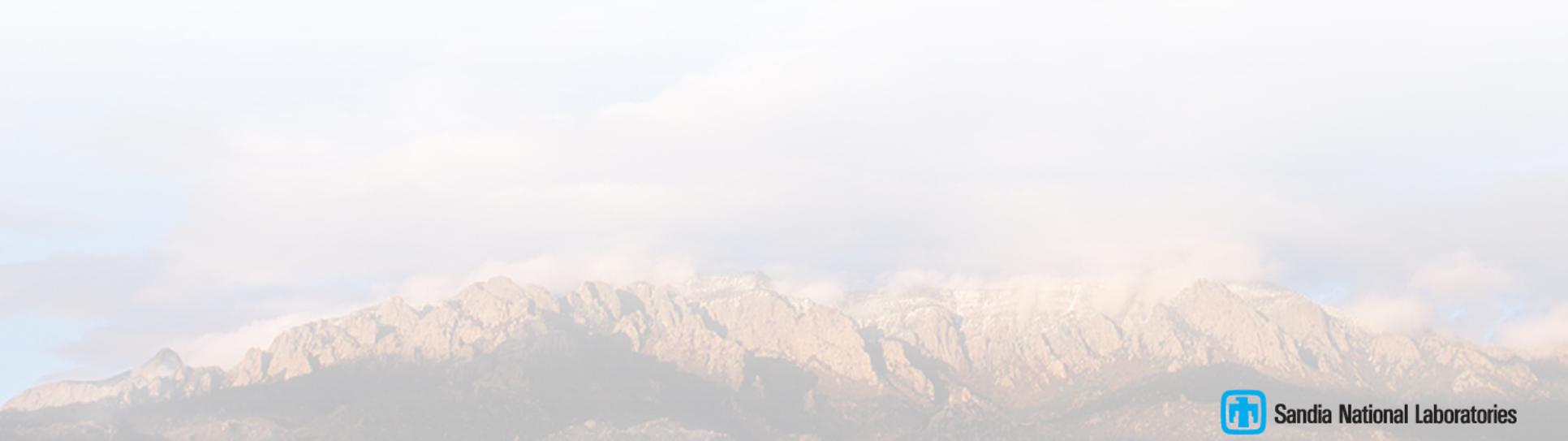


Drekar/CASL – Thermal-Hydraulics
– Pawlowski, Shadid, Smith, Cyr





Embedded Algorithms R&D using TBGP



Polynomial Chaos Expansions (PCE)



- **Steady-state finite dimensional model problem:**

Find $u(\xi)$ such that $f(u, \xi) = 0$, $\xi : \Omega \rightarrow \Gamma \subset R^M$, density ρ

- **(Global) Polynomial Chaos approximation:**

$$u(\xi) \approx \hat{u}(\xi) = \sum_{i=0}^P u_i \Psi_i(\xi), \quad \langle \Psi_i \Psi_j \rangle \equiv \int_{\Gamma} \Psi_i(x) \Psi_j(x) \rho(x) dx = \delta_{ij} \langle \Psi_i^2 \rangle$$

- **Non-intrusive polynomial chaos (NIPC, NISP):**

$$u_i = \frac{1}{\langle \Psi_i^2 \rangle} \int_{\Gamma} \hat{u}(x) \Psi_i(x) \rho(x) dx \approx \frac{1}{\langle \Psi_i^2 \rangle} \sum_{k=0}^Q w_k \bar{u}_k \Psi_i(x_k), \quad f(\bar{u}_k, x_k) = 0$$

- **Regression PCE:**

$$\hat{u}(x_k) = \bar{u}_k \implies \sum_{i=0}^P u_i \Psi_i(x_k) = \bar{u}_k, \quad f(\bar{u}_k, x_k) = 0, \quad k = 0, \dots, Q$$

- **Reduce number of samples by adding derivatives**
 - **Mike Eldred (SNL ASC)**

$$\sum_{i=0}^P u_i \Psi_i(x_k) = \bar{u}_k, \quad k = 0, \dots, Q$$
$$\sum_{i=0}^P u_i \frac{\partial \Psi_i}{\partial x}(x_k) = \frac{\partial \bar{u}_k}{\partial x_k}, \quad k = 0, \dots, Q$$



Computing Accurate Gradients Efficiently in PDE Simulations



- **Steady-state sensitivities**

- **Forward:**

$$f(u^*, p) = 0, \quad s^* = g(u^*, p) \implies$$

$$\frac{ds^*}{dp} = -\frac{\partial g}{\partial u}(u^*, p) \left(\frac{\partial f}{\partial u}(u^*, p) \right)^{-1} \frac{\partial f}{\partial p}(u^*, p) + \frac{\partial g}{\partial p}(u^*, p)$$

- **Adjoint :**

$$\left(\frac{ds^*}{dp} \right)^T = - \left(\frac{\partial f}{\partial p}(u^*, p) \right)^T \left(\frac{\partial f}{\partial u}(u^*, p) \right)^{-T} \left(\frac{\partial g}{\partial u}(u^*, p) \right)^T + \left(\frac{\partial g}{\partial p}(u^*, p) \right)^T$$

- **Single forward/transpose solve for each parameter/response**
 - **Accuracy determined by accuracy of partials, solution to linear systems**

- **Transient sensitivities:**

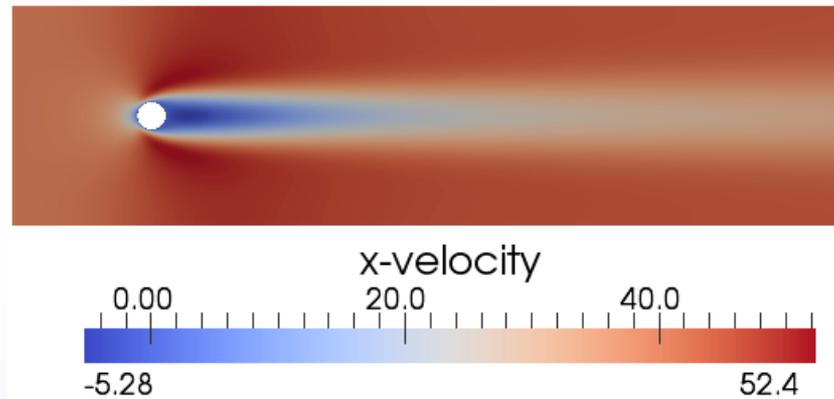
- **Forward:**

$$f(\dot{u}, u, p) = 0,$$
$$\frac{\partial f}{\partial \dot{u}} \frac{\partial \dot{u}}{\partial p} + \frac{\partial f}{\partial u} \frac{\partial u}{\partial p} + \frac{\partial f}{\partial p} = 0$$

- **Transient adjoint sensitivities are possible, but much harder**

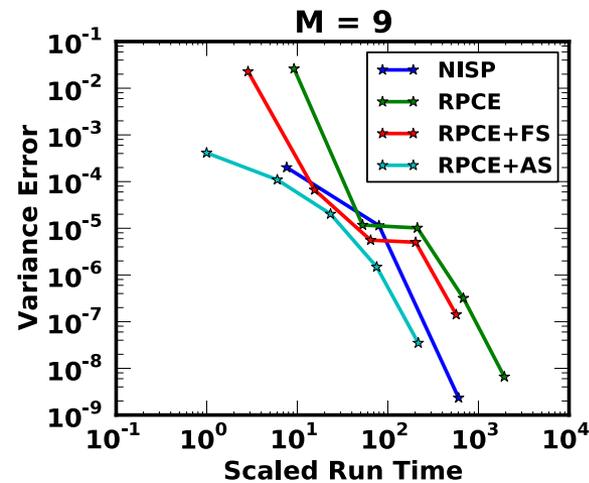
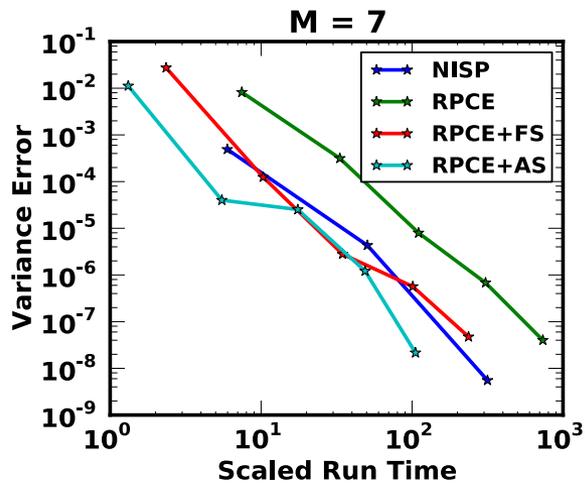
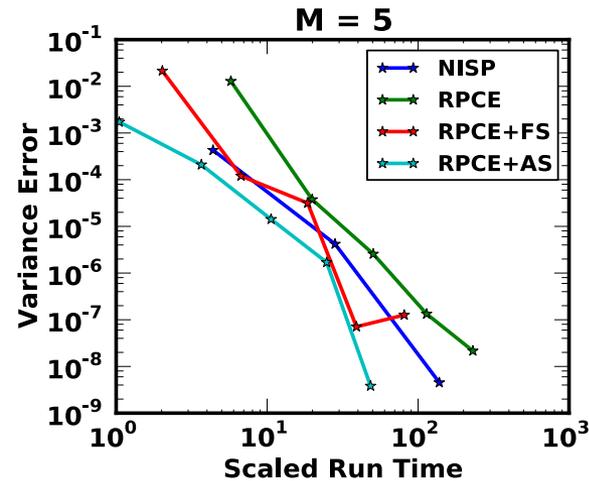
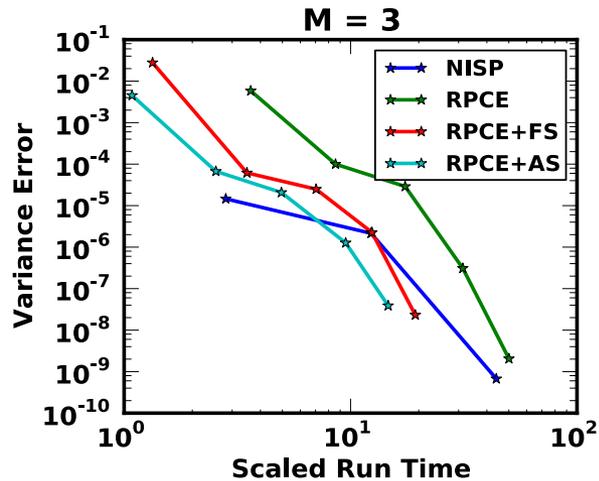
Small Model Problem

- **2-D incompressible fluid flow past a cylinder**



- **Albany code -- stabilized Galerkin FEM**
 - SUPG, PSPG
- **GMRES with RILU(2) preconditioning (Belos, Ifpack)**
- **Uncertain viscosity field**

Comparisons on Model Problem



Solver Reuse for Sampling-based Approaches



- **Sampling method can be viewed as a block-diagonal nonlinear system:**

$$\begin{bmatrix} f_1(u_1, x_1) = 0 \\ \vdots \\ f_N(u_N, x_N) = 0 \end{bmatrix} \implies \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & & \\ & \ddots & \\ & & \frac{\partial f_N}{\partial u_N} \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_N \end{bmatrix} = - \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

- **Leverage reuse**

- **Preconditioner**
- **Krylov basis^{1,2}**



- **Compute multiple residual/Jacobian samples simultaneously**

- **Multi-point TBGP scalar type**

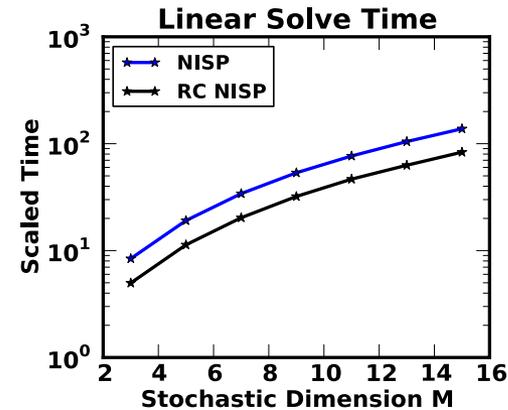
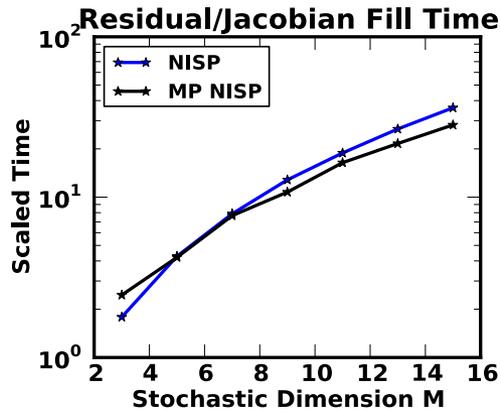
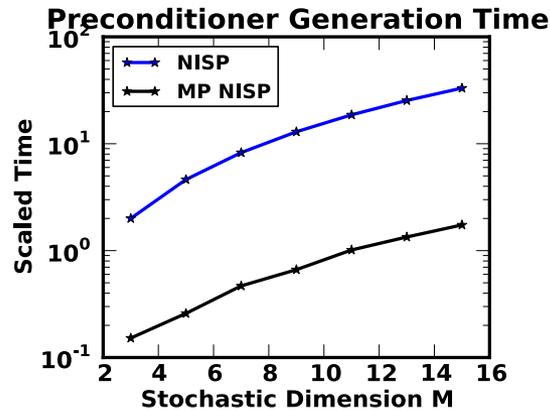
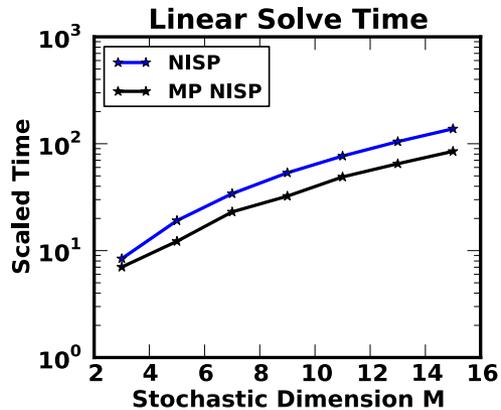
$$a = \{a_1, \dots, a_N\}, \quad b = \{b_1, \dots, b_N\}, \quad c = a \times b = \{a_1 \times b_1, \dots, a_N \times b_N\}$$

- **Improved vectorization, data locality**

¹C. Jin, X-C. Cai, and C. Li, *Parallel Domain Decomposition Methods for Stochastic Elliptic Equations*, SIAM Journal on Scientific Computing, Vol. 29, Issue 5, pp. 2069—2114, 2007.

²Michael L. Parks, Eric de Sturler, Greg Mackey, Duane Johnson, and Spandan Maiti, *Recycling Krylov Subspaces for Sequences of Linear Systems*, SIAM Journal on Scientific Computing, 28(5), pp. 1651-1674, 2006

Multi-point Sampling of Model Problem



- Only real improvement is reusing preconditioner
- Recycling benefits can be had just by recycling between Newton steps

Embedded Stochastic Galerkin UQ Methods

- **Steady-state stochastic problem (for simplicity):**

Find $u(\xi)$ such that $f(u, \xi) = 0$, $\xi : \Omega \rightarrow \Gamma \subset R^M$, density ρ

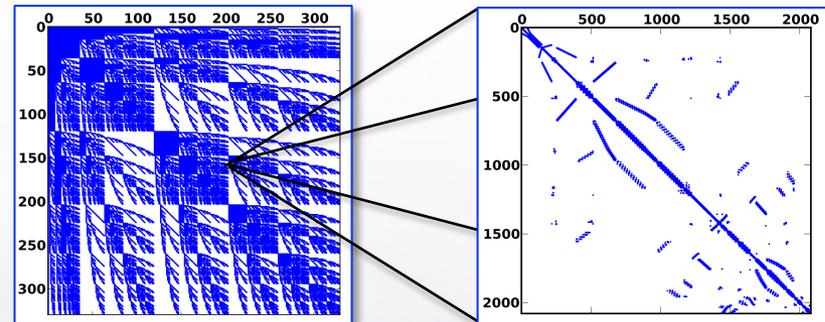
- **Stochastic Galerkin method (Ghanem and many, many others...):**

$$\hat{u}(\xi) = \sum_{i=0}^P u_i \psi_i(\xi) \rightarrow F_i(u_0, \dots, u_P) = \frac{1}{\langle \psi_i^2 \rangle} \int_{\Gamma} f(\hat{u}(y), y) \psi_i(y) \rho(y) dy = 0, \quad i = 0, \dots, P$$

– **Multivariate orthogonal basis of total order at most N – (generalized polynomial chaos)**

- **Method generates new coupled spatial-stochastic nonlinear problem (intrusive)**

$$0 = F(U) = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_P \end{bmatrix}, \quad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_P \end{bmatrix} \quad \frac{\partial F}{\partial U} :$$



Stochastic sparsity

Spatial sparsity

- **Advantages:**

– **Many fewer stochastic degrees-of-freedom for comparable level of accuracy**

- **Challenges:**

– **Computing SG residual and Jacobian entries in large-scale, production simulation codes**
 – **Solving resulting systems of equations efficiently, particularly for nonlinear problems**

Stokhos: Trilinos tools for embedded stochastic Galerkin UQ methods



- Eric Phipps, Chris Miller, Habib Najm, Bert Debuschere, Omar Knio

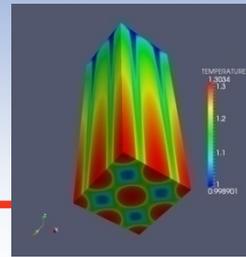


- Tools for describing SG discretization
 - Stochastic bases, quadrature rules, etc...
- C++ operator overloading library for automatically evaluating SG residuals and Jacobians
 - Replace low-level scalar type with orthogonal polynomial expansions
 - Leverages Trilinos Sacado automatic differentiation library

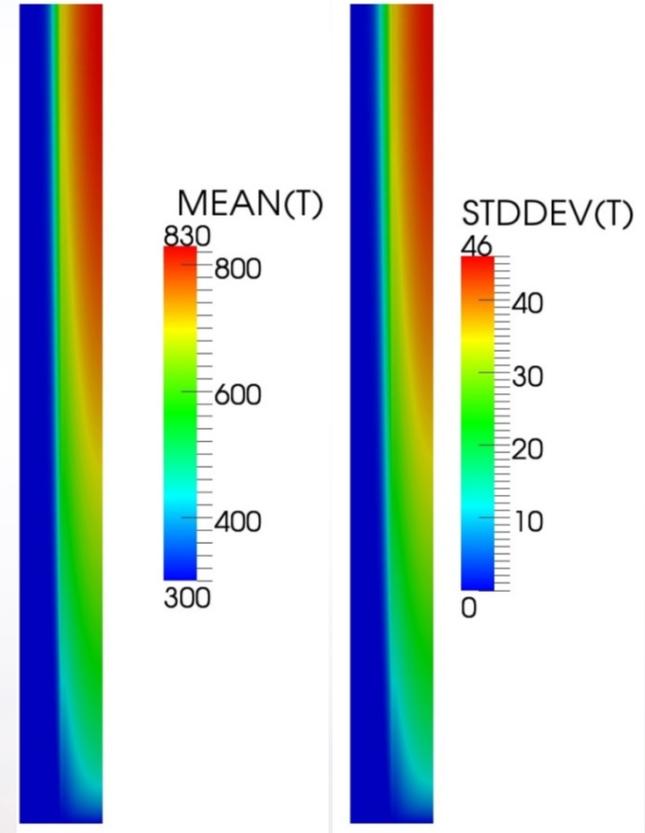
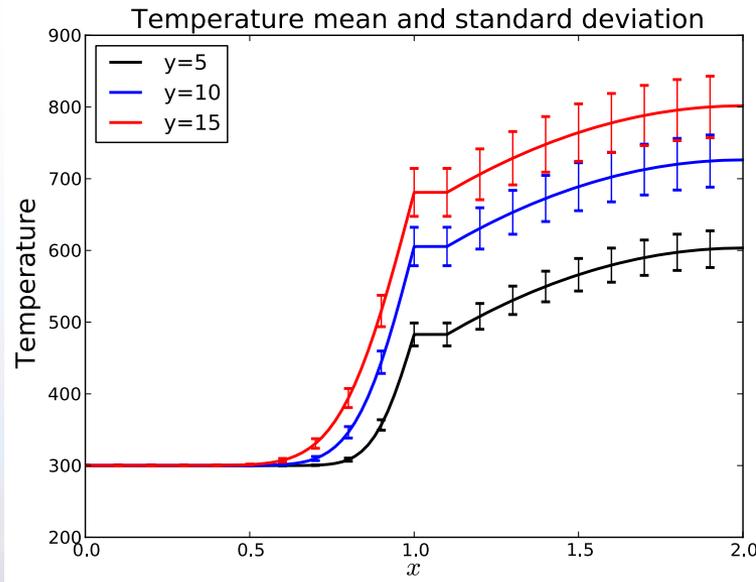
$$a = \sum_{i=0}^P a_i \psi_i, \quad b = \sum_{j=0}^P b_j \psi_j, \quad c = ab \approx \sum_{k=0}^P c_k \psi_k, \quad c_k = \sum_{i,j=0}^P a_i b_j \frac{\langle \psi_i \psi_j \psi_k \rangle}{\langle \psi_k^2 \rangle}$$

- Tools forming and solving SG linear systems
 - SG matrix operators
 - Stochastic preconditioners
 - Hooks to Trilinos parallel solvers and preconditioners
- Nonlinear SG application code interface
 - Connect SG methods to nonlinear solvers, time integrators, optimizers, ...

Embedded UQ in Drekar: Multiphysics: Rod to Fluid Heat Transfer



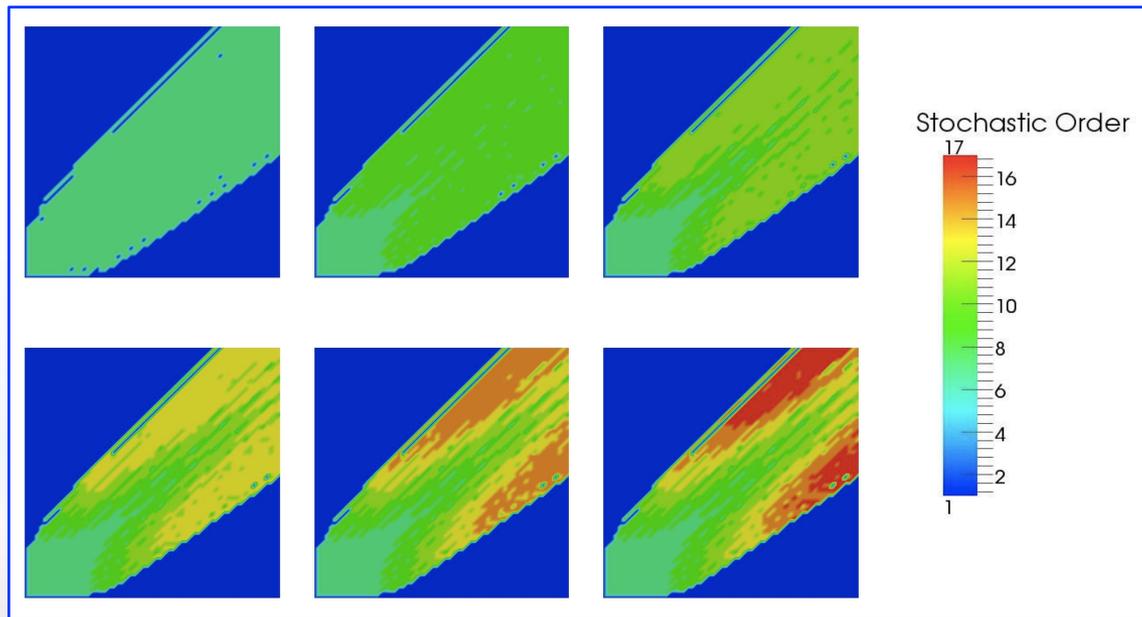
- True multiphysics formulation: conjugate heat transfer demonstrated in Drekar
- Embedded uncertainty quantification demonstration run using TBGP concepts at the 1 year mark
- Agile components significantly decreases the time to import cutting edge research into production applications



Stochastic Galerkin UQ analysis propagating uncertainty in the magnitude of the model fuel source term and the average inflow velocity.

Unique Embedded UQ R&D

- **Spatially adaptive UQ of a strongly convected field in Drekar**
 - Eric Cyr – SNL LDRD
 - 2-D convection-diffusion with stochastically varying inlet angle
 - Intrusive stochastic Galerkin with spatially varying polynomial order



- ***Possible only through embedded approaches***

Comparison between linear and nonlinear PDEs

Linear Problem

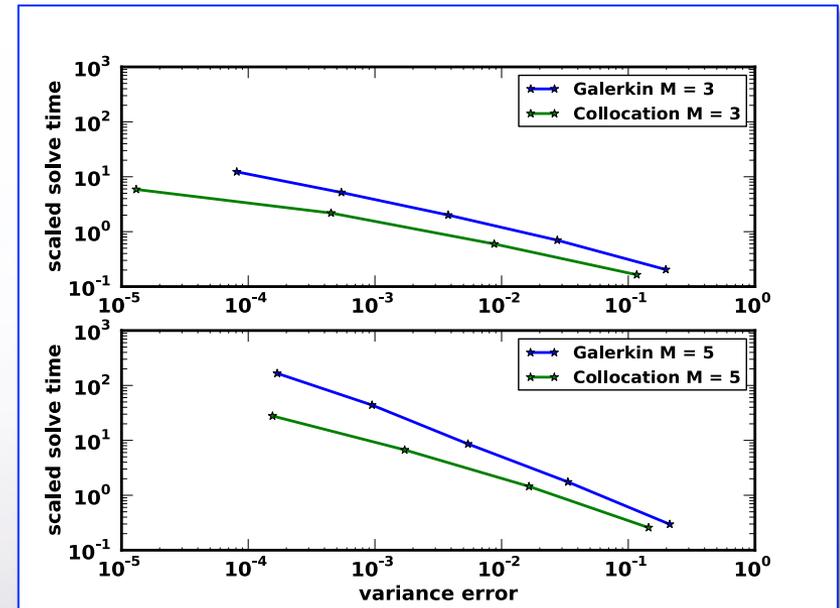
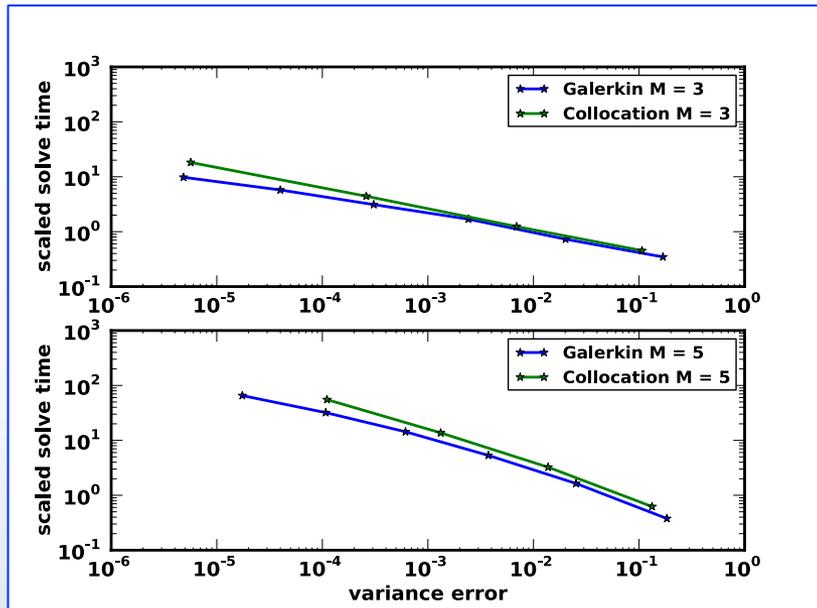
$$-\nabla \cdot (a(x, \xi) \nabla u) = 1, x \in [0, 1]^3$$

$$a(x, \xi) = \mu + \sigma \sum_{k=1}^M \sqrt{\lambda_k} f_k(x) \xi_k, \xi_k \sim U(-1, 1)$$

Nonlinear Problem

$$-\nabla \cdot (a(x, \xi) \nabla u) = \alpha u^2, x \in [0, 1]^3$$

$$a(x, \xi) = \mu + \sigma \sum_{k=1}^M \sqrt{\lambda_k} f_k(x) \xi_k, \xi_k \sim U(-1, 1)$$

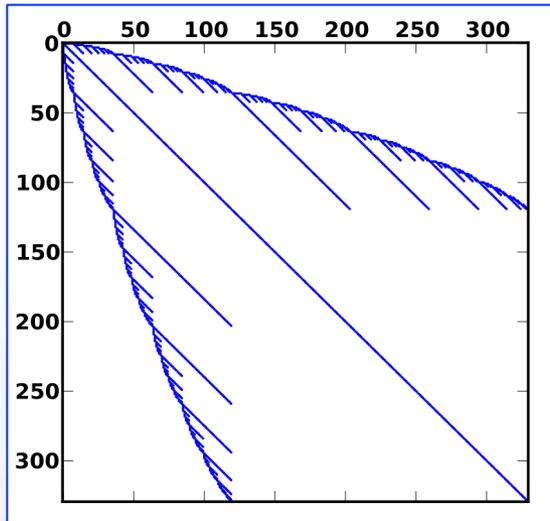


- Albany FEM code
- AztecOO Krylov solver
- ML mean-preconditioner
- Stokhos approximate Gauss-Seidel stochastic preconditioner

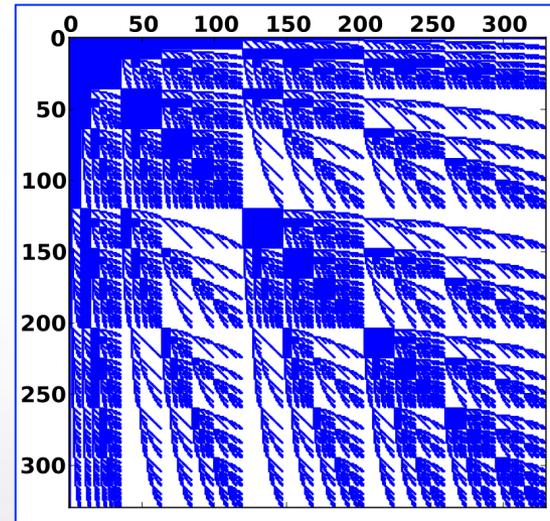
Comparison between linear and nonlinear PDEs

- Difference in performance due to dramatically reduced sparsity of the stochastic Galerkin operator
 - Increased cost of matrix-vector products

Linear Problem



Nonlinear Problem



- On-going R&D
 - Improved stochastic preconditioning
 - Dimension reduction for SG Jacobian operator
 - Multicore acceleration

Emerging Architectures Motivate New Approaches

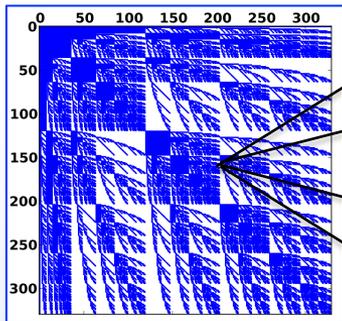
- UQ approaches usually implemented as an outer loop
 - Repeated calls of deterministic solver
- Single-point forward simulations use very little available node compute power (unstructured, implicit)
 - 3-5% of peak FLOPS on multi-core CPUs (P. Lin, Charon, RedSky)
 - 2-3% on contemporary GPUs (Bell & Garland, 2008)
- Emerging architectures leading to dramatically increased on-node compute power
 - Not likely to translate into commensurate improvement in forward simulation
 - Many simulations/solvers don't contain enough fine-grained parallelism
- Can this be remedied by inverting the outer UQ/inner solver loop?
 - Add new dimensions of parallelism through *embedded approaches*

Structure of Galerkin Operator

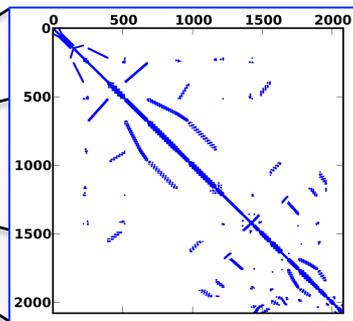
- Operator traditionally organized with outer-stochastic, inner-spatial structure
 - Allows reuse of deterministic solver data structures and preconditioners
 - Makes sense for sparse stochastic discretizations

$$J^{trad} = \sum_{k=0}^P G_k \otimes J_k$$

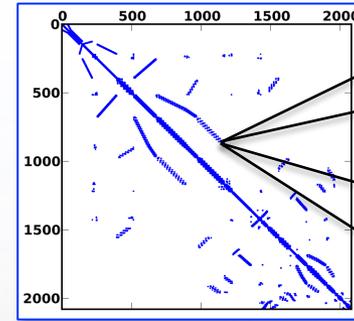
$$J^{com} = \sum_{k=0}^P J_k \otimes G_k$$



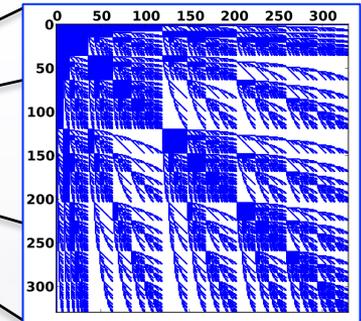
Stochastic sparsity



Spatial sparsity



Spatial sparsity

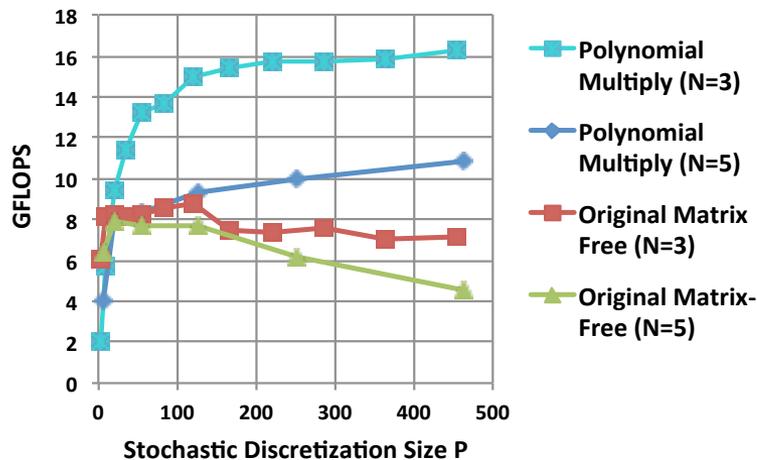


Stochastic sparsity

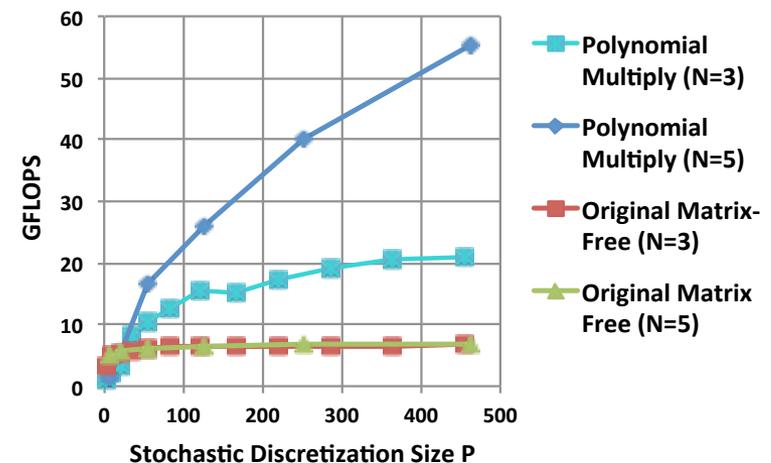
- For nonlinear problems, makes sense to commute this layout to outer-spatial, inner-stochastic
 - Leverage emerging architectures to handle denser stochastic blocks
 - Phipps, Edwards, Hu (SNL LDRD)

SG Mat-Vec Floating-point Rate

**Intel Westmere
(n=32k, 12 threads)**



**NVIDIA C2070
(n=32k)**



- Significant performance improvement, particularly for GPUs



Multiphysics Embedded UQ

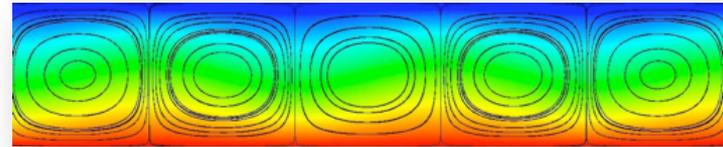
- **SNL**
 - Phipps, Constantine, Eldred, Pawlowski, Red-Horse, Schmidt, Wildey
- **USC**
 - Ghanem, Arnst, Tipireddy

Stochastic Coupled Nonlinear Systems

- Shared-domain multi-physics coupling
 - Equations coupled at each point in domain

$$\mathcal{L}_1(u_1(x), u_2(x), \xi_1) = 0$$

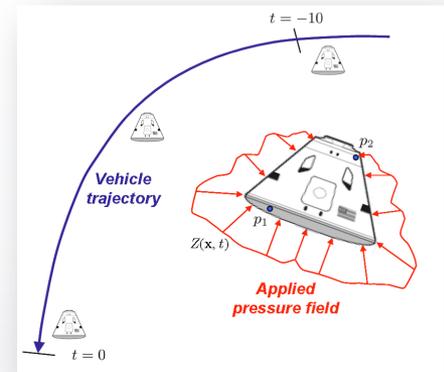
$$\mathcal{L}_2(u_1(x), u_2(x), \xi_2) = 0$$



- Interfacial multi-physics coupling
 - Equations are coupled through boundaries

$$\mathcal{L}_1(u_1(x), v_2(x_2), \xi_1) = 0, \quad v_2(x_2) = \mathcal{G}_2(u_2(x_2)), \quad x_2 \in \Gamma_2$$

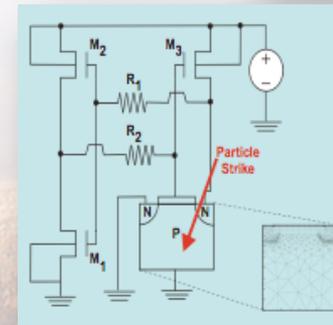
$$\mathcal{L}_2(v_1(x_1), u_2(x), \xi_2) = 0, \quad v_1(x_1) = \mathcal{G}_1(u_1(x_1)), \quad x_1 \in \Gamma_1$$



- Network coupling
 - Equations are coupled through a set of scalars

$$\mathcal{L}_1(u_1(x), v_2, \xi_1) = 0, \quad v_2 = \mathcal{G}_2(u_2)$$

$$\mathcal{L}_2(v_1, u_2(x), \xi_2) = 0, \quad v_1 = \mathcal{G}_1(u_1)$$



Curse of Dimensionality

- All three forms can be written after discretization

$$f_1(u_1, v_2, \xi_1) = 0, \quad u_1 \in \mathbb{R}^{n_1}, \quad v_2 = g_2(u_2) \in \mathbb{R}^{m_2}, \quad f_1 : \mathbb{R}^{n_1+m_2+M_1} \rightarrow \mathbb{R}^{n_1}$$

$$f_2(v_1, u_2, \xi_2) = 0, \quad u_2 \in \mathbb{R}^{n_2}, \quad v_1 = g_1(u_1) \in \mathbb{R}^{m_1}, \quad f_2 : \mathbb{R}^{m_1+n_2+M_2} \rightarrow \mathbb{R}^{n_2}$$

- Because system is coupled, each component must compute approximation over full stochastic space:

$$\hat{u}_1(\xi_1, \xi_2) = \hat{u}_1(\hat{v}_2(\xi_1, \xi_2), \xi_1)$$

- For segregated methods, requires solving sub-problems of larger dimensionality
 - Adding more components, or more sources of uncertainty in other components, increases cost of each sub-problem
- Mitigate curse of dimensionality by defining new random variables

$$\hat{u}_1(\xi_1, \xi_2) = \sum_{j=0}^P u_{1,j} \Psi_j(\xi_1, \xi_2) \longrightarrow \tilde{u}_1(\eta_2, \xi_1) = \sum_{j=0}^{\tilde{P}_1} \tilde{u}_{1,j} \Phi_j(\eta_2, \xi_1), \quad \eta_2 = \hat{v}_2(\xi_1, \xi_2)$$

- Size of each UQ problem now number of uncertain variables + number of interface variables
- Challenges: computing new orthogonal polynomials, associated quadrature rules

Stochastic Dimension Reduction

- Consider simplified problem of composite functions

$$h(x) = g(y), \quad y = f(x), \quad f : \Gamma \subset \mathbb{R}^M \rightarrow \mathbb{R}^L, \quad g : f(\Gamma) \rightarrow \mathbb{R}^S, \quad L \ll M$$

- with discrete inner product $(f_1, f_2) = \sum_{k=0}^Q w_k f_1(x^{(k)}) f_2(x^{(k)})$

- We wish to approximate $\hat{h}(x) = \sum_{i=0}^P h_i \Psi_i(x)$, $h_i = (h, \Psi_i)$, $V = \text{span}\{\Psi_i\}$

- Conceptual basis for dimension reduction:

- Compute subspace W given by span of monomials in y , projected onto V

$$y = f(x) = (y_1, \dots, y_L), \quad W = \text{span} \left\{ \sum_{i=0}^P (y_1^{k_1} \dots y_L^{k_L}, \Psi_i) \Psi_i \right\} \subset V$$

- Compute orthogonal basis for this subspace

$$\text{span}\{\Phi_i : i = 0, \dots, \tilde{P}\} = W, \quad (\Phi_i, \Phi_j) = \delta_{ij}, \quad \tilde{P}+1 = \dim(W), \quad \tilde{P} \ll P$$

- Compute reduced quadrature rule by requiring exactness on this space

$$\sum_{l=0}^{\tilde{Q}} \tilde{w}_l \Phi_i(x^{(k_l)}) \Phi_j(x^{(k_l)}) = \delta_{ij}, \quad \tilde{Q} \ll Q$$

- Compute reduced projection

$$\tilde{h}(x) = \sum_{i=0}^{\tilde{P}} \tilde{h}_i \Phi_i(x), \quad \tilde{h}_i = \sum_{l=0}^{\tilde{Q}} \tilde{w}_l g(y^{(k_l)}) \Phi_i(y^{(k_l)}), \quad y^{(k_l)} = f(x^{(k_l)})$$

- Compute final transformation back to original basis

$$\bar{h}(x) = \sum_{i=0}^P \bar{h}_i \Psi_i(x), \quad \bar{h}_i = \sum_{j=0}^{\tilde{P}} \tilde{h}_j \alpha_{ij}, \quad \alpha_{ij} = (\Psi_i, \Phi_j)$$



Devil is in the Details

- **Computing W , orthogonal basis accurately is challenging**
 - Gram-Schmidt QR
- **Variety of approaches for reduced quadrature**
 - Least-squares
 - Linear program (arXiv: 1112.4772)
- **In 1-D ($L = 1$) this is much easier**
 - Discretized Stieltjes = Lanczos (arXiv 1110.0058)
- **Alternative approach**
 - Apply Lanczos approach to each component of y
 - 1-D orthogonal polynomials, Gauss rules
 - Total order tensor product polynomials, sparse grid quadrature
 - This spans W , but is not an orthogonal basis!
 - Project onto this basis using this quadrature rule/inner product
 - Project onto original basis, using above as a surrogate
 - Relying on point-wise convergence w.r.t. wrong inner product/measure
 - This can fail catastrophically

Coupled neutron-transport and heat transfer demonstration

- 2-D “slab reactor” (H. Stripling):

$$Q - \frac{1}{L^2} \int_D \Phi(x) \Sigma_f(\bar{T}) E_f dx = 0 \quad \text{s.t.} \quad -\nabla \cdot (D(\bar{T}) \nabla \Phi(x)) + (\Sigma_a(\bar{T}) - \nu \Sigma_f(\bar{T})) \Phi(x) = S(x, \xi_1),$$
$$\bar{T} - \frac{1}{L^2} \int_D T(x) dx = 0 \quad \text{s.t.} \quad -\nabla \cdot (k(x, \xi_2) \nabla T(x)) = Q,$$

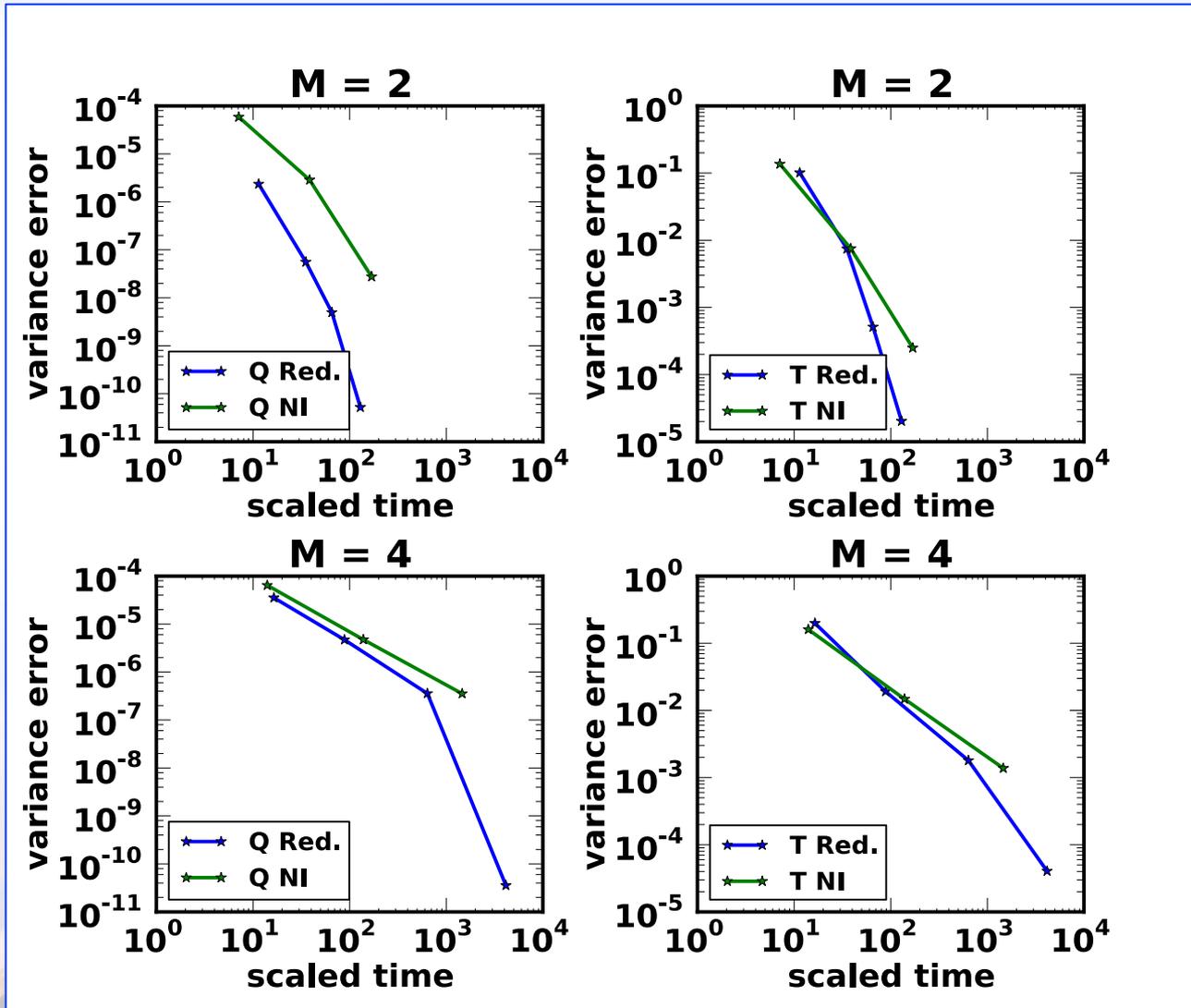
$$x \in D = [0, L]^2, \quad \Sigma(\bar{T}) = \Sigma(T_0) \sqrt{\frac{\bar{T}_0}{\bar{T}}}, \quad D = \frac{1}{3(\Sigma_a + \Sigma_s)}$$

$$S(x, \xi_1) = S_0(x) + \sigma_S \sum_{i=0}^M \sqrt{\lambda_i} a_i(x) \xi_{1,i}, \quad k(x, \xi_2) = k_0(x) + \sigma_k \sum_{i=0}^M \sqrt{\mu_i} b_i(x) \xi_{2,i},$$

$\xi_{1,i}, \xi_{2,i}$ Uniform on $(-1, 1)$

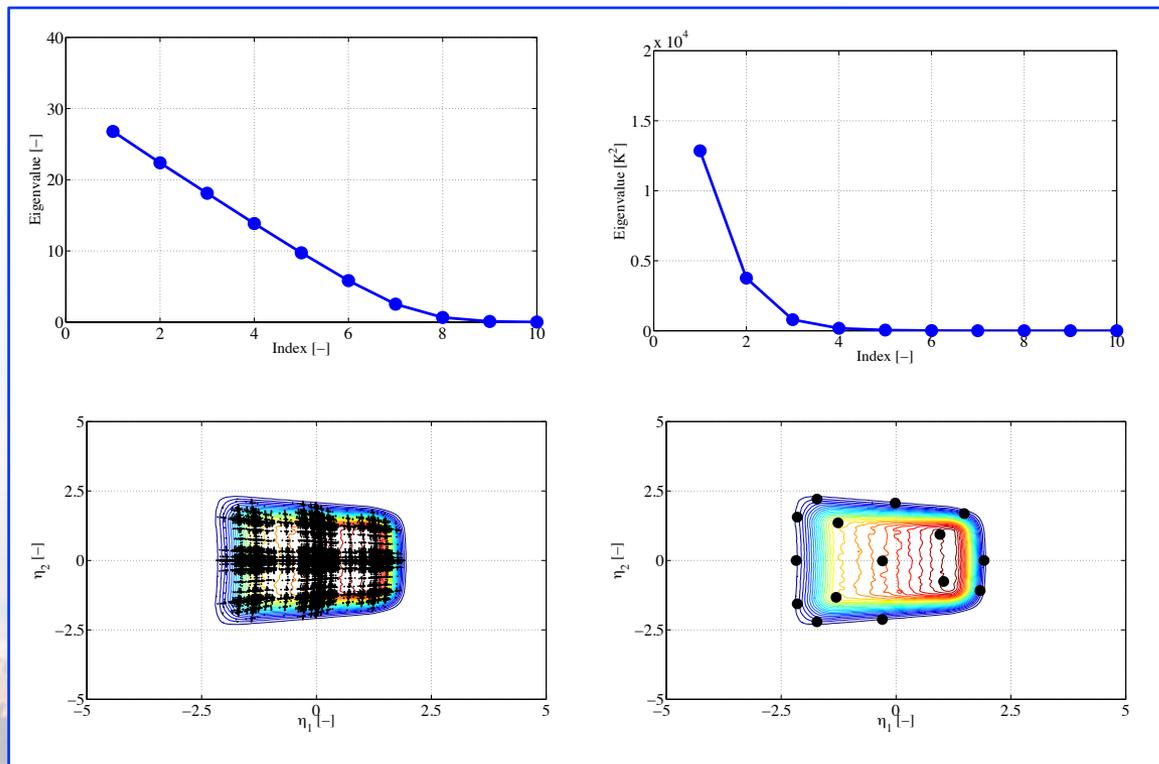
- 2-component network system, nonlinear elimination coupling
 - Non-intrusive: Sparse-grid quadrature provided through Dakota on space of size 2^*M
- Dimension reduction:
 - Tensor-product Lanczos variant of approach outlined previously
 - Intrusive (stochastic Galerkin) at 2×2 network level, non-intrusive for each component
 - Each component UQ problem of size $M+1$

Results



Dimension Reduction in Shared-Domain/ Interfacial Coupling

- Approaches rely on small dimensional interfaces between physics
 - Network coupling – built into the model
 - Shared-domain/interfacial – transfer between physics may live on small dimensional manifold
 - Use KL to parameterize this (arXiv: 1112.4761)

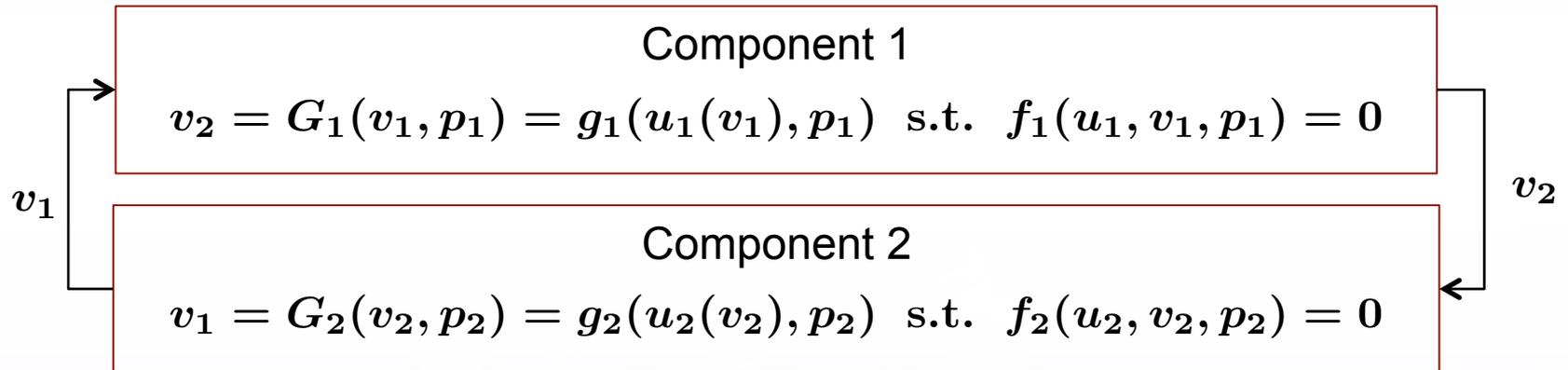




Concluding Remarks

- **Looked at a variety of embedded UQ algorithms leverage structure**
 - **Simulation structure**
 - **Architecture structure**
- **An approach for incorporating them in large-scale codes**
 - **Template-based generic programming**
 - **Agile components**
- **Powerful vehicle for investigating embedded algorithms with path to impact important applications**

Nonlinear Elimination for Network Coupled Systems



Nonlinear elimination

Equations

$$v_2 - G_1(v_1, p_1) = 0$$

$$v_1 - G_2(v_2, p_2) = 0$$

Newton Step

$$\begin{bmatrix} -dG_1/dv_1 & 1 \\ 1 & -dG_2/dv_2 \end{bmatrix} \begin{bmatrix} \Delta v_1 \\ \Delta v_2 \end{bmatrix} = - \begin{bmatrix} v_2 - G_1(v_1, p_1) \\ v_1 - G_2(v_2, p_2) \end{bmatrix}$$

$$\frac{dG_i}{dv_i} = - \frac{\partial g_i}{\partial u_i} \left(\frac{\partial f_i}{\partial u_i} \right)^{-1} \frac{\partial f_i}{\partial v_i}$$