Stokhos: Trilinos Tools for Embedded Stochastic-Galerkin Uncertainty Quantification Methods

Eric Phipps
etphipp@sandia.gov
Optimization & Uncertainty Quantification Department

Trilinos User Group 2008

SAND 2008-7239 C
Models of Uncertainty

• Predictive simulation means making a rigorous statement about the world based on computational simulation
  – Must understand uncertainty in simulation input data and its effects on simulations

• Two broad classes of uncertainty
  – Aleatory or irreducible uncertainty: “inherent randomness”
    • Probabilistic representations (random variables, stochastic processes)
  – Epistemic or reducible uncertainty: “lack of knowledge”
    • Set/knowledge representations (intervals, fuzzy sets, evidence theory)
    • Probabilistic representations (Bayesian point of view)

• Two basic sources of uncertainty
  – Finite-dimensional simulation parameters
    • Represented through random variables/vectors with given distributions
  – Infinite-dimensional correlated random fields/stochastic processes
    • Realizations often smoothly varying in time/space; allows representation with small number of random variables/vectors (e.g., Karhunen-Loeve representation)

• Assume a finite dimensional representation given by random variables/vectors with known probability distributions
  – Generating such models is a research area in its own right
Stochastic Galerkin Uncertainty Quantification Methods
(More Commonly Known as Polynomial Chaos Methods)

- Deterministic, steady-state problem (possibly after spatial discretization):
  \[ u(p) \text{ such that } F(u; p) = 0, \quad p \in \Gamma \subset \mathbb{R}^M \]

- Stochastic problem:
  \[ u(\xi) \text{ such that } F(u; \xi) = 0, \quad \xi : \Omega \rightarrow \Gamma, \text{ density } \rho \]

- Let \( Z \) be a finite-dimensional subspace of \( L^2_\rho(\Gamma) \) with basis \( \{ \psi_i : i = 0, \ldots, N_{PC} \} \)
  orthogonal with respect to inner product
  \[ \langle fg \rangle \equiv \int_\Gamma f(y)g(y)\rho(y)dy \]

- Stochastic Galerkin method (Ghanem, …):
  \[ \hat{u}(\xi) = \sum_{i=0}^{N_{PC}} u_i\psi_i(\xi) \in Z \rightarrow F_i(u_0, \ldots, u_{N_{PC}}) = \int_\Gamma F(\hat{u}(y); y)\psi_i(y)\rho(y)dy = 0, \quad i = 0, \ldots, N_{PC} \]

- Typically \( Z \) is the complete polynomial space of total degree \( P \), basis polynomials are
tensor products of 1-D polynomials orthogonal with respect to 1-D density of each random
parameter
  - Assumes independence of random parameters
  - Gaussian random variables -- Hermite polynomials
  - Uniform random variables -- Legendre polynomials, …

- Exponential convergence in \( P \) when solution is sufficiently smooth
Stochastic Galerkin Nonlinear System

- Method generates new coupled spatial-stochastic nonlinear problem

\[ 0 = \tilde{F}(\tilde{u}) = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{N_{PC}} \end{bmatrix}, \quad \tilde{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_{PC}} \end{bmatrix} \]

- Dimension grows rapidly with degree or dimension

\[ N_{PC} = \frac{(M + P)!}{M!P!} \]

- System Jacobian:

\[ \hat{J}(\xi) = \sum_{k=0}^{N_{PC}} J_k \psi_k(\xi) \rightarrow J_k = \frac{1}{\langle \psi_k^2 \rangle} \int_{T} \frac{\partial F}{\partial u}(\tilde{u}(y); y)\psi_k(y)\rho(y)dy \]

\[ \frac{\partial F_i}{\partial u_j} \approx \sum_{k=0}^{N_{PC}} J_k \langle \psi_i \psi_j \psi_k \rangle \]
Implementing SG Methods in Nonlinear Applications is Challenging

• Code transformation from deterministic code to SG code
  – Need tools/libraries to automate computation of SG residual and Jacobian entries

\[ F(u; p) = 0 \rightarrow \bar{F}(\bar{u}) = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{NPC} \end{bmatrix} = 0 \]

• Data structures & interfaces for forming block SG systems
  – Linear, nonlinear, transient, optimization, stability, …

• Solver algorithms for block SG systems
  – Exploit new dimensions of parallelism
  – “Jacobian-free” methods are key

\[ \frac{\partial F_i}{\partial u_j} \approx \sum_{k=0}^{NPC} J_k \langle \psi_i \psi_j \psi_k \rangle \quad \rightarrow \quad \left( \frac{\partial \bar{F}}{\partial \bar{u}} \right)_i = \sum_{j,k=0}^{NPC} J_{kj} v_j \langle \psi_i \psi_j \psi_k \rangle \]

• Trilinos provides powerful capabilities here
SG Code Transformation Through Automatic Differentiation (AD)

- Trilinos package Sacado provides AD capabilities to C++ codes
  - AD data types & overloaded operators
  - Replace scalar type in application with Sacado AD data types

- AD relies on known derivative formulas for all intrinsic operations plus chain rule

- AD infrastructure provides deep interface into application code
  - Access to scalar-level computations in application

- Similar approach can be used for any computation that can be done in an operation by operation manner
  - Assume inductively that SG expansions for two intermediate variables $a$ and $b$ have been computed, and we wish to compute a third $c = \varphi(a, b)$

\[
\hat{a}(\xi) = \sum_{i=0}^{N_{PC}} a_i \psi_i(\xi), \quad \hat{b}(\xi) = \sum_{i=0}^{N_{PC}} b_i \psi_i(\xi), \quad \text{Find } \hat{c}(\xi) = \sum_{i=0}^{N_{PC}} c_i \psi_i(\xi) \text{ such that}
\]

\[
\int_{\Gamma} (\hat{c}(y) - \varphi(\hat{a}(y), \hat{b}(y))) \psi_i(y) \rho(y) dy = 0, \quad i = 0, \ldots, N_{PC}
\]
SG Projections of Intermediate Operations

- Addition/subtraction

\[ c = a \pm b \Rightarrow c_i = a_i \pm b_i \]

- Multiplication

\[ c = a \times b \Rightarrow \sum_i c_i \psi_i = \sum_i \sum_j a_i b_j \psi_i \psi_j \rightarrow c_k = \sum_i \sum_j a_i b_j \frac{\langle \psi_i \psi_j \psi_k \rangle}{\langle \psi_k^2 \rangle} \]

- Division

\[ c = a/b \Rightarrow \sum_i \sum_j c_i b_j \psi_i \psi_j = \sum_i a_i \psi_i \rightarrow \sum_i \sum_j c_i b_j \langle \psi_i \psi_j \psi_k \rangle = a_k \langle \psi_k^2 \rangle \]

- Several approaches for transcendental operations
  - Implemented in Fortran library by Najm, Debusschere, Ghanem, Knio

- These ideas allow the implementation of Sacado “AD” types for intrusive stochastic Galerkin methods
  - Easy transition once code is setup to use AD
Other Trilinos Tools Useful for SG Methods

- Epetra -- MPI-based vector/matrix data structures & operator interfaces
  - Used by application codes to form FE residuals, Jacobians
- Thyra -- Abstract vector, operator, and nonlinear interfaces
  - Product vectors for representing block SG solution/residual vectors
  - Operators implementing SG matrix-vector-product in “matrix-free” fashion
  - Nonlinear interface transforming deterministic Thyra interface into SG
- AztecOO, Belos, Ifpack, ML -- Linear solvers and preconditioners
  - Advanced linear solver techniques optimized for block SG structure
- Zoltan, Isorropia -- Graph partitioning & reordering
  - Partitioning, reordering of block SG linear system
- NOX, LOCA, Rythmos -- Nonlinear solver & time integration algorithms
  - Use nonlinear Thyra SG interface to solve steady & transient SG problems
Trilinos Package Stokhos

• These ideas form the basis for a new Trilinos package called Stokhos
  – Collaborative effort among the SG/PCE community to develop tools for large-scale codes

• Initial thoughts are Stokhos will provide (or push development of)
  – SG vector/operator interfaces
  – Nonlinear SG application interface
  – Solver/preconditioner algorithms
  – Intrusive propagation methods

• Currently it only has
  – General facilities for intrusive propagation
  – Wrappers around UQLib library of Najm, Debusschere, Ghanem & Knio
  – Sacado wraps these for AD SG/PCE
    • Sacado::PCE::OrthogPoly<double>

• Sacado::FEApp
  – Example 1D finite element code demonstrating AD
  – Initial implementation of SG interfaces using Epetra & EpetraExt
Sacado::FEApp Demonstration of Intrusive SG using Stokhos & Sacado

- 1-D Bratu problem:
  \[ \frac{\partial^2 u}{\partial x^2} + e^{(\alpha_1 + \cdots + \alpha_M)/M} e^u = 0, \quad 0 \leq x \leq 1 \]

- Linear finite element discretization, 100 elements

- Uniform random variables for nonlinear factor over [-1, 1] using Legendre polynomials

- SG residual/Jacobian entries computed through Sacado

- “Jacobian free” linear solver method using Ifpack RILU(0) of mean block for preconditioner

- Solution mean used as quantity of interest
Intrusive SG Compared to Non-Intrusive Polynomial Chaos (NIPC)

Non-Intrusive Polynomial Chaos
- Dakota
- Sparse-grid quadrature

\[ \hat{u}(\xi) = \sum_{i=0}^{N_{PC}} u_i \psi_i(\xi) \]

\[ u_i = \frac{1}{\langle \psi_i^2 \rangle} \int_{\Gamma} u(y) \psi_i(y) \rho(y) dy \]

\[ \approx \frac{1}{\langle \psi_i^2 \rangle} \sum_{k=0}^{N_Q} w_k u_k \psi_i(y_k) \]

\[ F(u_k; y_k) = 0, \ k = 0, \ldots, N_Q \]
Results Motivate Quadrature Approach for Element-Based Codes

\[ F(u; p) = \sum_{j=0}^{N_E} Q_j^T f_j(P_k u; p) \]

• Evaluate via quadrature for globally assembled residual

\[ F_i = \int_{\Gamma} F(\hat{u}(y); y) \psi_i(y) \rho(y) dy \approx \sum_{k=0}^{N_Q} w_k F(\hat{u}(y_k); y_k) \psi_i(y_k) \]

  – Requires parallel quadrature routines but only interface to global residual

• Apply quadrature for each element residual then assemble

\[ F_i = \sum_{j=0}^{N_E} Q_j^T \int_{\Gamma} f_j(P_j \hat{u}(y); y) \psi_i(y) \rho(y) dy \approx \sum_{j=0}^{N_E} Q_j^T \left( \sum_{k=0}^{N_Q} w_k f_j(P_j \hat{u}(y_k); y_k) \psi_i(y_k) \right) \]

  – Requires only serial quadrature routines but needs element-level interface

  – Boundary conditions add complexity

• Jacobian decomposes similarly

• Speed of quadrature residual/Jacobian fills + benefits of intrusive solve
What We’re Working on

• SG residual/Jacobian fills
  – Sparse quadrature (Dakota)
  – Can the AD approach be improved?

• Linear solver/preconditioner methods for block SG linear systems
  – Multi-level, incomplete factorization methods?

• Stokhos software tools
  – Trilinos/Dakota integration
  – Thyra interfaces
  – Coordination with Najm, et al on UQLib development

• Demonstration/investigation in real apps
  – Charon, Albany

• Release Stokhos with Trilinos 10