Embedded Stochastic Galerkin Projection and Solver Methods via Template-based Generic Programming

Eric Phipps (etphipp@sandia.gov), Roger Pawlowski, Andy Salinger, Sandia National Laboratories,
Roger Ghanem, Ramakrishna Tipireddy, University of Southern California

http://trilinos.sandia.gov

http://www.cs.sandia.gov/DAKOTA

Goal: Understand costs vs. benefits of intrusive methods

Embedded Stochastic Galerkin UQ Methods

- Steady-state stochastic problem:

$$\text{Find } u(\xi) \text{ such that } f(u,\xi) = 0 \text{ a.e., } \xi : \Omega \to \Gamma \subset \mathbb{R}^M, \text{ density } \rho,\ \langle\cdot\rangle \equiv \int_\Gamma \cdot\, \rho(y)dy$$

- Stochastic Galerkin (SG) method, a.k.a., Intrusive Polynomial Chaos (PC):

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

- Basis polynomials are *usually* tensor products of 1-D orthogonal polynomials of total degree at most N (assumes independence):

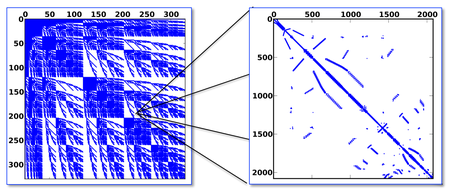
$$\psi_i(y) = \psi_{i_1}^1(y_1)\dots\psi_{i_M}^M(y_M),\quad i_1+\dots+i_M \le N,\quad i = 0,\dots,P,\quad P = \frac{(M+N)!}{M!N!} - 1$$

- Method generates new coupled spatial-stochastic nonlinear problem:

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

$$\left(\frac{\partial F}{\partial U}\right)_{i,j} = \frac{\partial F_i}{\partial u_j} = \frac{1}{\langle\psi_i^2\rangle}\int_\Gamma \frac{\partial f}{\partial u}(\hat{u}(y),y)\psi_i(y)\psi_j(y)\rho(y)dy \approx \sum_{k=0}^{P} J_k\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle},$$

$$\frac{\partial f}{\partial u}(\hat{u}(y),y) \approx \sum_{k=0}^{P} J_k\psi_k(y),\ J_k = \frac{1}{\langle\psi_k^2\rangle}\int_\Gamma \frac{\partial f}{\partial u}(\hat{u}(y),y)\psi_k(y)\rho(y)dy$$



Stochastic sparsity — Spatial sparsity

Challenges: Computing SG residual/Jacobian coefficients and solving resulting SG systems

Stokhos: Trilinos Tools for Embedded SG Methods

- Eric Phipps, Ramakrishna Tipireddy (USC), Chris Miller (Maryland), Habib Najm (SNL), Bert Debusschere (SNL), Omar Knio (Johns Hopkins)
- Overloaded operators for TBGP-based embedded PC propagation (see below)
  - –Sacado: Trilinos AD tools for C++ applications
  - –Hooks to UQToolkit of Najm *et al* for Taylor, integration-based approaches
  - –Hooks to Burkardt sparse grid quadrature libraries through DAKOTA/PECOS
- Tools for forming/solving SG linear/nonlinear systems
  - –Block product vectors, multi-vectors, and matrices
  - –"Matrix free" block SG operator

$$\frac{\partial F_i}{\partial u_j} \approx \sum_{k=0}^{P} J_k\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle} \implies \left(\frac{\partial F}{\partial U}V\right)_i = \sum_{j=0}^{P}\sum_{k=0}^{P} J_k v_j\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle}$$

  - –Distributed memory parallelism (MPI) over spatial and stochastic degrees-of-freedom
  - –Hooks to Trilinos parallel, iterative linear solvers & preconditioners
  - –Customized SG block solver, preconditioning methods (see below)
- Nonlinear SG application code interface
  - –Interface to nonlinear solver, time integrator, optimizer, …

Enables study of intrusive methods in complex simulation codes

Embedded PC Propagation through Template-Based Generic Programming

- Foundation in automatic differentiation (AD) techniques
  - –AD relies on known derivative formulas for all intrinsic operations plus chain rule:

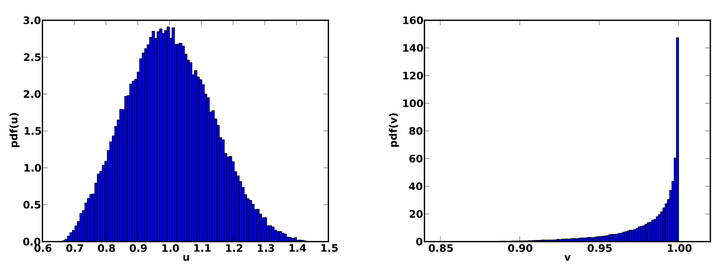
$$c = a * b \implies dc = da * b + a * db$$

  - –Principle applies to all kinds of calculations:
    - Jacobians, directional derivatives, Hessians, adjoints, Taylor series, …
    - Sparsity patterns, extended precision, flop counts, bounding intervals, …
  - –Similar approach possible for SG expansion, e.g.,

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

  - –Transcendental operations more difficult (see Debusschere *et al*, 2004)
    - Taylor series (simple but not robust), line integration (robust but expensive)
    - Sparse grid quadrature (simple and robust)
- Implement in C++ codes through classing, operator overloading, and templating
  - –Sacado: Trilinos AD tools for C++ applications
  - –Pawlowski, *et al*, 2011

```
#include "Stokhos_Sacado.hpp"

// The function to compute the polynomial chaos expansion of, written as a template function
template <class ScalarType>
inline ScalarType simple_function(const ScalarType& u) {
  return 1.0/(std::pow(std::log(u),2.0) + 1.0);
}

// Typename of PC expansion type
typedef Sacado::PCE::OrthogPoly<double, Stokhos::StandardStorage<int,double> > pce_type;

int main(int argc, char **argv) {

    // Basis of dimension 3, order 5
    const int d = 3; const int p = 5;
    Teuchos::Array< Teuchos::RCP<const Stokhos::OneDOrthogPolyBasis<int,double> > > bases(d);
    for (int i=0; i<d; i++) {
      bases[i] = Teuchos::rcp(new Stokhos::LegendreBasis<int,double>(p));
    }
    Teuchos::RCP<const Stokhos::CompletePolynomialBasis<int,double> > basis =
      Teuchos::rcp(new Stokhos::CompletePolynomialBasis<int,double>(bases));

    // Quadrature method
    Teuchos::RCP<const Stokhos::Quadrature<int,double> > quad =
      Teuchos::rcp(new Stokhos::TensorProductQuadrature<int,double>(basis));

    // Triple product tensor
    Teuchos::RCP<Stokhos::Sparse3Tensor<int,double> > Cijk =
      basis->computeTripleProductTensor(basis->size());

    // Expansion method
    Teuchos::RCP<Stokhos::QuadOrthogPolyExpansion<int,double> > expn =
      Teuchos::rcp(new Stokhos::QuadOrthogPolyExpansion<int,double>(
                                        basis, Cijk, quad));

    // Polynomial expansions
    pce_type u(expn);
    u.term(0,0) = 1.0;
    for (int i=0; i<d; i++) {
      u.term(i,1) = 0.4 / d; u.term(i,2) = 0.06 / d; u.term(i,3) = 0.002 / d;
    }

    // Compute PCE expansion of function
    pce_type v = simple_function(u);
```



Simplifies incorporation of SG methods in complex code

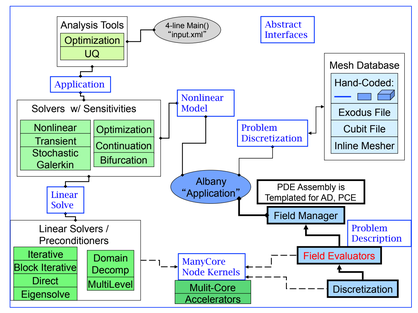
References

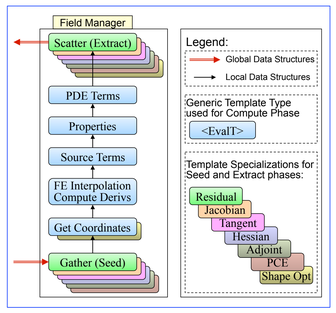
1. B.J. Debusschere, H.N. Najm, P.P. Pebay, O.M. Knio, R.G. Ghanem and O.P. Le Maitre, "Numerical challenges in the use of polynomial chaos representations for stochastic processes," SIAM J. Sci. Comput., vol. 26 (2), 2004.
2. H.C. Elman, C.W. Miller, E.T. Phipps, and R.S. Tuminaro, "Assessment of Collocation and Galerkin Approaches to Linear Diffusion Equations with Random Data", International Journal for Uncertainty Quantification, vol. 1 (1), 2010.
3. **R.P. Pawlowski, E.T. Phipps, and A.G. Salinger, "Automating embedded analysis capabilities using template-based generic programming," submitted to Journal of Scientific Programming, 2011.**
4. **R.P. Pawlowski, E.T. Phipps, and A.G. Salinger, S.J. Owen, C. Seifert and M.L. Staten, "Applying template-based generic programming to the simulation and analysis of partial differential equations," submitted to Journal of Scientific Programming, 2011**
5. **R. Tipireddy, E.T. Phipps, and R.G. Ghanem, "A comparison of solution methods for stochastic partial differential equations," submitted to International Journal of Computer Mathematics, 2011.**
6. E. Ullman, "A Kronecker Product Preconditioner for Stochastic Galerkin Finite Element Discretizations," SIAM J. Sci. Comput., Vol. 32 (2), 2010.

Albany Agile Components Demonstration Application

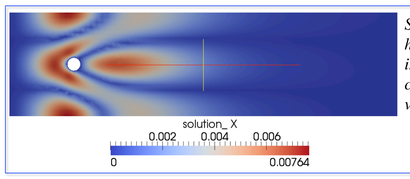
(Salinger – PI)

- Brings together numerous Trilinos tools in fully functional application code
  - –Finite element discretizations of 1, 2, and 3D PDEs on unstructured meshes
- Fully integrates template-based generic programming approach
  - –Templated finite element residual calculation
  - –Analytic derivatives, sensitivities with Sacado
  - –Intrusive polynomial chaos expansions with Stokhos
  - –Pawlowski, *et al*, 2011





- Rapid development of complex physics
  - –Incompressible Navier-Stokes (Salinger)
  - –Neutronics (Phipps)
  - –Linear and nonlinear elasticity (Ostien)
  - –Quantum device modeling (Muller)



Standard deviation of horizontal fluid velocity for incompressible flow around a cylinder with uncertain viscosity field

Rapid development of complex physics supporting advanced analysis

New Block Preconditioning Methods for SG Discretizations

$$\bar{P}X = B \implies X = \bar{P}^{-1}B$$

- Traditional mean-based preconditioning (e.g., Elman *et al*, 2010)

$$P_0 \approx J_0,\quad \bar{P} = I \otimes P_0 \implies x_i = (\bar{P}^{-1}B)_i = P_0^{-1}b_i,\quad i = 0,\dots,P$$

- Kronecker-product precondtioner (Ullman, 2010)

$$\bar{P} = G\otimes P_0,\quad G = \sum_{k=0}^{P}\frac{\mathrm{tr}(J_k^T J_0)}{\mathrm{tr}(J_0^T J_0)}G_k,\quad G_k(i,j) = \frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle},\quad i,j,k = 0,\dots,P$$

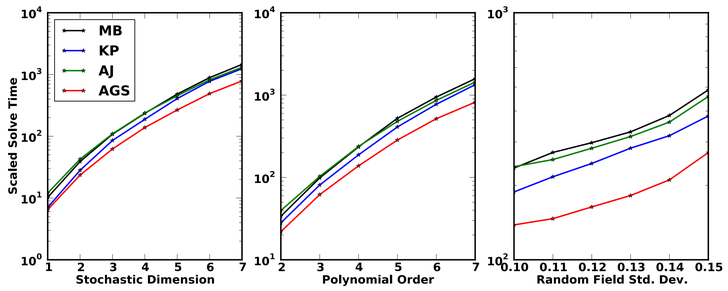
- Approximate Jacobi (Tipireddy *et al*, 2011)

$$x_i^{(l+1)} = P_0^{-1}\left(b_i - \sum_{j=0}^{P}\sum_{k=1}^{P} J_k x_j^{(l)}\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle}\right),\quad i = 0,\dots,P,\quad l = 1,2$$

- Approximate Gauss-Seidel (Tipereddy *et al*, 2011)

$$x_i^{(l+1)} = P_0^{-1}\left(b_i - \sum_{j=0}^{l}\sum_{k=1}^{P} J_k x_j^{(l+1)}\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle} - \sum_{j=l+1}^{P}\sum_{k=1}^{P} J_k x_j^{(l)}\frac{\langle\psi_i\psi_j\psi_k\rangle}{\langle\psi_i^2\rangle}\right),$$
$$i = 0,\dots,P,\quad l = 1$$

- 2-D linear diffusion equation with uniform random field (exponential covariance) diffusion coefficient:



Simple ideas, but significant improvement over existing approaches

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.