An Unintrusive Implementation of Generalized Polynomial Chaos
Sanjay R. Mathur and Aarti Chigullapalli, Purdue University

Galerkin-based Generalized Polynomial Chaos (gPC)

Consider:
\[ \nabla \cdot (k(\varepsilon) \nabla T) = 0 \]
\[ k = f(\varepsilon); \quad \varepsilon \text{ is random} \]

Expand variables in polynomial basis
\[ k = \sum_{i} k_i(x) P_i(\varepsilon) \quad T = \sum_{i} T_i(x) P_i(\varepsilon) \]

- Substitute in pde; perform Galerkin projection
- Derive separate pdes for unknown coefficients; may be non-linear and coupled
- Discretize and solve pdes
- Post-process polynomial expansion of outputs to find means, standard deviations and higher moments

Discrete Stochastic Galerkin gPC: Let the compiler do it!

Exploit C++ features:
- Templated class PC through user-defined data types
- Operator overloading
- Templates and template meta-programming

Original C++ code

```cpp
void myfunc (const double& x, const double& y, double& p, double& q) {
    p = 3 * x * x + sin(y);
    q = p / y;
}
```

Templated C++ code

```cpp
Template <class T>
void myfunc (const T& x, const T& y, T& p, T& q) {
    p = 3 * x * x + sin(y);
    q = p / y;
}
```

- PC class T contains gPC coefficients
- Number of random variables and order of gPC expansion chosen at compile time.
- Operators overloaded to perform Galerkin projections
- UQ Toolkit from Sandia employed

Verification: 1D Heat Conduction with Random Thermal Conductivity

\[ \nabla \cdot (k(\varepsilon) \nabla T) = 0 \]
\[ k = 1 + \varepsilon x \]
\[ \varepsilon: \text{Gaussian} \]
\[ \varepsilon = 0, \quad \sigma_\varepsilon = 0.1 \]

- UQTK compared with analytical solution
- 3rd-order gPC expansion in Hermite polynomials
- Excellent agreement with exact solution.

Advantages and Disadvantages of Galerkin Approach

- Spectral convergence
- Intrusive – must code in new pdes for each new physical model
- New pdes are not conservation equations; may be non-linear and coupled
- Parallelization needed for each new pde

Advantages and Disadvantages of Collocation Approach

- Unintrusive; can be used with legacy code
- Embarassingly parallel
- However, determination of coefficients requires quadrature formula of twice the polynomial order - number of collocation points may become very large for large number of random inputs

Collocation gPC

\[ T(\varepsilon) = \sum_{i=0}^{\infty} T_i(x) P_i(\varepsilon) \]
\[ T_i(x) = \int \frac{T(x,\varepsilon) P_i(\varepsilon) d\varepsilon}{P_i(\varepsilon) d\varepsilon} = \frac{1}{h_i} \int T(x,\varepsilon) P_i(\varepsilon) d\varepsilon \]

- Expand output variables in polynomial basis
- Coefficients found by exploiting orthogonality
- Approximate resulting integration with quadrature formula
- Employ sparse-grid collocation to find coefficients of expansion:
  \[ T_i(x) = \frac{1}{h_i} \sum_{i=1}^{\infty} T_i(x,\varepsilon) \]

Advantages and Disadvantages of Collocation Approach

- Unintrusive; can be used with legacy code
- Embarassingly parallel
- However, determination of coefficients requires quadrature formula of twice the polynomial order - number of collocation points may become very large for large number of random inputs

Future Work

- More testing for non-linear problems, greater number of independent random variables; establish timing
- Speed-up of operator overloading algorithms for basic operations

Laminar Flow in Driven Cavity

Re=100

Viscosity is a Gaussian random variable

Mean of u, v on vertical centerline

Standard deviation of u, v on vertical centerline

- Viscosity is Gaussian, with mean of 1.0 and standard deviation of 0.1 kg/ms
- 3rd-order gPC in Hermite polynomials for UQTK
- 2nd-order gPC in Legendre polynomials for collocation
- Monte Carlo with 500 and 1000 samples

Conclusions

- UQ Toolkit approach gives accurate solutions.
- Simple extension of MEMOSA’s sensitivity framework
- Applies to all physical models in code
- Parallel framework carries over automatically

Future Work

- More testing for non-linear problems, greater number of independent random variables; establish timing
- Speed-up of operator overloading algorithms for basic operations