Polynomial chaos expansions part 3: Intrusive Galerkin method

Jonathan Feinberg and Simen Tennøe

Kalkulo AS

March 4, 2015

Relevant links

A very basic introduction to scientific Python programming:
http://hplgit.github.io/bumpy/doc/pub/sphinx-basics/index.html

Installation instructions:
https://github.com/hplgit/chaospy

Repetition of our model problem

We have a simple differential equation

$$\frac{du(x)}{dx} = -au(x), \qquad u(0) = I$$

with the solution

$$u(x) = Ie^{-ax}$$

with two random input variables:

$$a \sim \text{Uniform}(0, 0.1), \qquad I \sim \text{Uniform}(8, 10)$$

Want to compute $\text{E}(u)$ and $\text{Var}(u)$

The Galerkin method is a projection method for approximating functions

Given a function space V and inner product on V
$\langle u, v\rangle_Q = \int_0^L uvdx$

$$u'(x) = g(x)$$
$$\int_0^L u'(x)v(x)dx = \int_0^L g(x)v(x)dx, \quad \forall v \in V$$
$$\left\langle u', v\right\rangle_Q = \langle g, v\rangle_Q \quad \text{(projection)}$$

With $u(x;q) \approx \hat{u}_M(x;q) = \sum_{n=0}^{N} c_n(x)P_n(q)$ this leads to a linear system for the coefficients c_n.

Calculating initial condition using Galerkin

$$\hat{u}_M(0) = I, \qquad \hat{u}_M = \sum_{n=0}^{N} c_n(x) P_n(q)$$

$$\sum_{n=0}^{N} c_n(0) P_n = I$$

$$\left\langle \sum_{n=0}^{N} c_n(0) P_n, P_k \right\rangle_Q = \langle I, P_k \rangle_Q \qquad k = 0, \ldots, N$$

$$\sum_{n=0}^{N} c_n(0) \langle P_n, P_k \rangle_Q = \langle I, P_k \rangle_Q$$

$$c_k(0) \langle P_k, P_k \rangle_Q = \langle I, P_k \rangle_Q$$

$$c_k(0) = \frac{\langle I, P_k \rangle_Q}{\langle P_k, P_k \rangle_Q} = \frac{E(IP_k)}{E(P_k^2)}$$

Galerkin applied to the differential equation

$$\frac{d}{dx}\left(\hat{u}_M\right) = -a\hat{u}_M$$

$$\frac{d}{dx}\left(\sum_{n=0}^{N} c_n P_n\right) = -a\sum_{n=0}^{N} c_n P_n$$

$$\left\langle \frac{d}{dx}\left(\sum_{n=0}^{N} c_n P_n\right), P_k\right\rangle_Q = \left\langle -a\sum_{n=0}^{N} c_n P_n, P_k\right\rangle_Q \qquad k = 0, \dots, N$$

$$\frac{d}{dx}\sum_{n=0}^{N} c_n \left\langle P_n, P_k\right\rangle_Q = -\sum_{n=0}^{N} c_n \left\langle aP_n, P_k\right\rangle_Q$$

$$\frac{d}{dx}c_k \left\langle P_k, P_k\right\rangle_Q = -\sum_{n=0}^{N} c_n \left\langle aP_n, P_k\right\rangle_Q$$

$$\frac{d}{dx}c_k = -\sum_{n=0}^{N} c_n \frac{\left\langle aP_n, P_k\right\rangle_Q}{\left\langle P_k, P_k\right\rangle_Q} = -\sum_{n=0}^{N} c_n \frac{\mathrm{E}(aP_nP_k)}{\mathrm{E}(P_k^2)}$$

The Galerkin Projection results in a coupled $(N+1) \times (N+1)$ system of differential equations

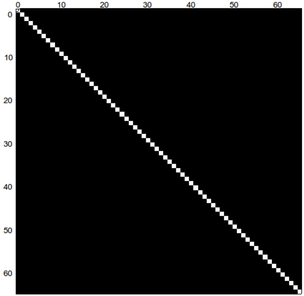
$$\frac{d}{dx} c_k(x) = -\sum_{n=0}^{N} c_n(x) \frac{E(aP_nP_k)}{E(P_k^2)} \qquad k = 0, \ldots, N$$

$$c_k(0) = \frac{E(IP_k)}{E(P_k^2)}$$

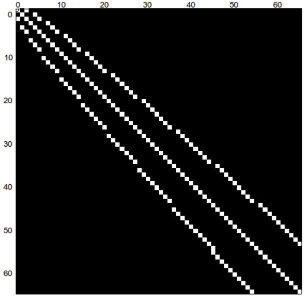
$$\frac{d}{dx}\mathbf{c} = -\mathbf{M}\mathbf{c}, \quad M_{kn} = \frac{E(aP_nP_k)}{E(P_k^2)}$$

The differential equation system is very sparse (mostly zeros)

$E(P_n P_k)$



$E(a P_n P_k)$



Intrusive Galerkin usually converges faster

- Original problem: one scalar differential equation
- Stochastic UQ problem: system of differential equations
- The method is called *intrusive Galerkin*
- The original solver cannot be reused

Solving the set of differential equations numerically

```
import chaospy as cp
import numpy as np
import odespy

dist_a = cp.Uniform(0, 0.1)
dist_I = cp.Uniform(8, 10)
dist = cp.J(dist_a, dist_I) # joint multivariate dist

P, norms = cp.orth_ttr(n, dist, retall=True)
variable_a, variable_I = cp.variable(2)
```

Solving the set of differential equations numerically

```
PP = cp.outer(P, P)
E_aPP = cp.E(variable_a*PP, dist)
E_IP = cp.E(variable_I*P, dist)

def right_hand_side(c, x):                # c' = right_hand_sid
  return -np.dot(E_aPP, c)/norms      # -M*c

initial_condition = E_IP/norms

solver = odespy.RK4(right_hand_side)
solver.set_initial_condition(initial_condition)

x = np.linspace(0, 10, 1000)
c = solver.solve(x)[0]

u_hat = cp.dot(P, c)
```

Intrusive Galerkin usually converges faster