Technische Universität München
Lehrstuhl für Informatik V
Dr. Tobias Neckel
M. Sc. Ionuț Farcaș

SS 2017

June 21, 2017

Algorithms for Uncertainty Quantification

Tutorial 8: Sparse pseudo-spectral approximation

In this worksheet, we focus on the application of sparse grid quadrature in uncertainty quantification. More specifically, we apply sparse grid quadrature for the pseudo-spectral approach, i.e. the evaluation of the polynomial chaos expansion coefficients via numerical quadrature.

Sparse quadrature

In the lecture, we discussed the sparse grid idea and we saw how to apply sparse grid quadrature for a five dimensional stochastic version of our model problem, i.e. the second order damped oscillator.
In `chaospy`, it is very easy to use sparse grid quadrature. Since the sparse grid idea applies to problems with dimensionality greater than two, the first step is to create a multivariate probability distribution. For simplicity, in the following code snippet we assume that we have a three-variate uniform distribution

```
import chaospy as cp

# create three uniform distribution objects
distr_d1 = cp.Uniform()
distr_d2 = cp.Uniform()
distr_d3 = cp.Uniform()

# create the joint three dimensional distribution
distr_3D = cp.J(distr_1, distr_2, distr_3)
```

After we created the multivariate distribution, the next step is to generate (sparse) quadrature nodes and weights based this distribution. Referring to the previous example, the syntax is

```
import chaospy as cp

...

# create the joint three dimensional distribution
distr_3D = cp.J(distr_1, distr_2, distr_3)

# create sparse quadrature nodes and weights
# this is obtained by simply adding an extra argument to the
    generate_quadrature function discussed in previous tutorials
# for example, we generate sparse Gaussian nodes and weights
    starting from a univariate quadrature rule of degree
    quad_deg_1D

nodes, weights = cp.generate_quadrature(quad_deg_1D, distr_3D,
    rule='G', sparse=True)
```

To generate multi-variate orthogonal polynomials, the syntax is the same as for 1D, i.e.

```
import chaospy as cp

...

# create the joint three dimensional distribution
distr_3D = cp.J(distr_1, distr_2, distr_3)

# generate orthogonal polynomials w.r.t. to the distribution
    distr_3D
# the first argument of the orth_ttr function is the maximal
    degree of the 1D polynomials, used to generate the multi-
    variate polynomials

poly = cp.orth_ttr(poly_deg_1D, distr_3D, normed=True)
# normed=True/False depending whether you prefer to have
    orthonormal/orthogonal polynomials, respectively
```

Assignment 1

Let $f : \mathbb{R}^5 \to \mathbb{R}^5, \quad f(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 + \sin(x_2 + x_4) - \exp(-x_5) x_1 + x_5^2 - x_1$. Approximate $\int_{[0,1]^5} f(x_1, x_2, x_3, x_4, x_5) dx_1 dx_2 dx_3 dx_4 dx_5$ via numerical quadrature. Use both the sparse and non-sparse version based on 1D Gaussian points of order $K = [3, 4, 5]$. What do you observe?

Assignment 2

Consider the model problem, the linear damped oscillator

$$\begin{cases} \frac{d^2y}{dt^2}(t) + c\frac{dy}{dt}(t) + ky(t) = f\cos(\omega_O t) \\ y(0) = y_0 \\ \frac{dy}{dt}(0) = y_1. \end{cases} \tag{1}$$

Let $t \in [0, 20]$, $\Delta t = 0.01, c \sim \mathcal{U}(0.08, 0.12)$, $k \sim \mathcal{U}(0.03, 0.04)$, $f \sim \mathcal{U}(0.08, 0.12)$, $y_0 \sim \mathcal{U}(0.45, 0.55)$, $y_1 \sim \mathcal{U}(-0.05, 0.05)$ and $\omega_O = 1.0$. The output of interest is $y(10)$. Write a `python` + `chaospy` program to propagate the uncertainty in (c, k, f, y_0, y_1) through the model in Eq. (1) using the generalized polynomial chaos expansion method and asses the expansion coefficients using the pseudo-spectral approach. Consider both non-sparse and sparse 5D pseudo-spectral computation of the coefficients, constructed on Gaussian nodes. To generate multi-variate quadrature nodes and weights consider $K = 5$ (for the 1D quadrature rule); to construct multi-variate orthogonal polynomials, consider $N = 2$ (for the 1D orthogonal polynomials). What do you observe?