Jonas Schäfer
Keyuan Wang

Homework 4

Task Description

Consider the following function with two random inputs:

$$f(Z_1, Z_2) = -(Z_1 - 2)(Z_2 - 1)^3 + \exp\left[-\frac{1}{2}(Z_1 - 2)^2 - \frac{1}{10}(Z_2 - 1)^2\right]$$

With $Z1 \sim U[1,3]$ and $Z2 \sim N[1,1]$. Construct a truncated gPC expansion to approximate f, for two different truncations: one with $i = i_1 + i_2 \leq 3$ and one with $max(i_1, i_2) \leq 3$. Show a plot of the two approximations as well as the exact function f. Compute the mean and variance of $f(Z_1, Z_2)$ from the gPC expansion coefficients. Compare them to estimates for the mean and variance obtained with Monte Carlo sampling of $f(Z_1, Z_2)$.

Task 1

Firstly, we change the interval for Z_1. By setting $Z_1^* = Z_1 - 2$, we have $Z_1^* \sim U[-1, 1]$, which is perfect for Gaussian Quadrature $[-1.1]$. The function turns to:

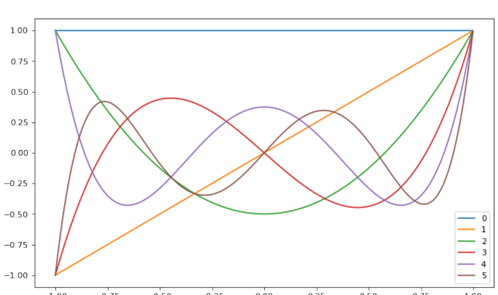
$$f(Z_1^*, Z_2) = -Z_1^* * (Z_2 - 1)^3 + \exp\left[-\frac{1}{2}Z_1^{*2} - \frac{1}{10}(Z_2 - 1)^2\right]$$

Then we find the orthogonal polynomials corresponding to Z_1^*, Z_2.
For Z_1^*, Legendre Polynomials:

$$L_0(Z) = 1, \quad L_1(Z) = Z, \quad L_2(Z) = \frac{3}{2}Z^2 - \frac{1}{2}, \quad L_3(Z) = \frac{5}{2}Z^3 - \frac{3}{2}Z, \ldots$$

For Z_2, Hermite Polynomials:

$$H_0(Z) = 1, \quad H_1(Z) = 2Z, \quad H_2(Z) = 4Z^2 - 2, \quad H_3(Z) = 8Z^3 - 12Z, \ldots$$



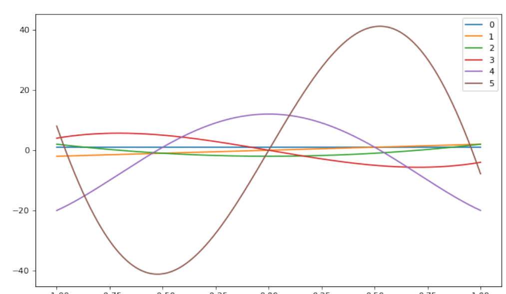


Figure 1: Legendre (left) and Hermite (right) polynomials

Task 2

We construct the orthogonal polynomials sets $\{\phi_i\}$ with the equation:

$$\Phi_i = \prod_{k=1}^{d} \phi_{i_k}^{(k)}\left(Z_k\right), \quad i = (i_1, i_2, \ldots, i_d)$$

Then we implement the orthogonal projection:

$$f \approx P_N f = \sum_{|i| \leq N} \hat{f}_i \phi_i(Z)$$

with

$$\hat{f}_i = \left(\gamma_i\right)^{-1} E\left[f(Z)\phi_i(Z)\right]$$

By using Gaussian Quadrature, we get:

$$\hat{f}_i = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} w_i^{(1)} w_j^{(2)} f\left(Z_1^*, Z_2\right) \phi_{i1}^{(1)}\left(Z_1^*\right) \phi_{i2}^{(2)}\left(Z_2\right)}{\sum_{i=0}^{n} \varnothing_{i1}^{(1)}\left(Z_1^*\right)^2 w_i^{(1)} * \sum_{i=0}^{n} \phi_{i2}^{(2)}\left(Z_2\right)^2 w_i^{(2)}}$$

So, we can numerically compute the orthogonal projection $P_N f$ with different truncations.

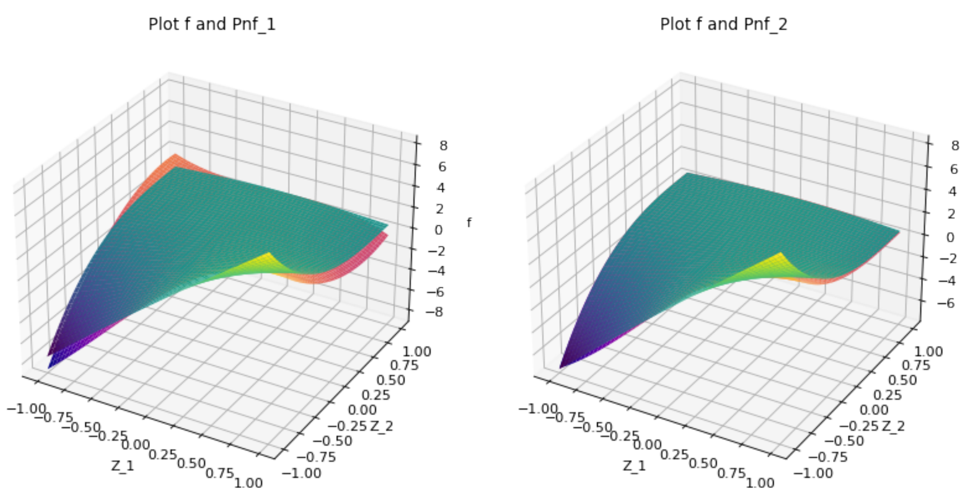


Figure 2: f plotted next to $P_N f_1$ $(i_1 + i_2 \leq 3)$ and $P_N f_2$ $(max(i_1, i_2) \leq 3)$

Unsurprisingly, we can see that the increased number of basis functions helps to approximate f better, hence $P_N f_2$ performed better.

Task 3

We estimate the mean and variance for original function $f(Z_1^*, Z_1)$ using Monte Carlo Sampling. We get a mean of 0.7794 and a variance of 5.017 rounded to 4 significant digits.

Then we compute the mean and variance from gPC expansion coefficients:

$$\begin{aligned} \text{Mean } &: E\left(f_N(Z)\right) = \hat{f}_0 \gamma_0 \approx 0.717 \\ \text{Variance } &: \operatorname{Var}\left(f_N(Z)\right) = \textstyle\sum_{n=1}^{N} \left(\hat{f}_n\right)^2 \gamma_n \end{aligned}$$

Appendix

Code 1: Libraries

```
# Libraries used

import numpy as np
import numpy.polynomial.legendre as lg
import numpy.polynomial.hermite as her
import scipy as sp
from matplotlib import pyplot as plt
```

Code 2: Legendre Polynomials

```
# Visualize Legendre Polynomials
t = np.arange(-1, 1, 0.001)

plt.figure(figsize=(10, 6), dpi=80)
for n in range(6):
    l = lg.Legendre.basis(n)
    plt.plot(t, l(t))
plt.legend([n for n in range(10)])
plt.show()
```

Code 3: Hermite Polynomials

```
# Visualize Hermite Polynomials
plt.figure(figsize=(10, 6), dpi=80)
for n in range(6):
    l = her.Hermite.basis(n)
    plt.plot(t, l(t))
plt.legend([n for n in range(10)])
plt.show()
```

Code 4: General functions

```
# Random Variables, f and basis function creation utility
z_u = np.random.uniform(1, 3)
z_u_scaled = np.random.uniform(-1, 1) + 2
z_n = np.random.normal(-1, 1)
# We modify f by substituting z_1 with (z_1 + 2) and hence drawing from distribution [-1, 1]
f = lambda z_1, z_2: -(z_1 + 2 - 2)*(z_2 - 1)**3 + \
    np.exp(-0.5*(z_1 + 2 - 2)**2 - 0.1*(z_2 - 1)**2)
# phi_def helps us generate polynomial a basis combining legendre and hermite polynomials
phi_def = lambda i_1, i_2: \
    (lambda z_1, z_2: lg.Legendre.basis(i_1)(z_1) * her.Hermite.basis(i_2)(z_2))
```

Code 5: Generation of multivariate gPC functions

```
# i_1 + i_2 <= 3
I_1 = [(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (2,0), (2, 1), (3,0)]
# generates basis for this I using our util function phi_def
phi_1 = [phi_def(i_1, i_2) for (i_1, i_2) in I_1]

# max(i_1, i_2) <= 3
I_2 = [(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1),\
    (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)]
# generates basis for this I using our util function phi_def
phi_2 = [phi_def(i_1, i_2) for (i_1, i_2) in I_2]

# A generator of Pnf functions given I and phi, it returns a gPC function
def Pnf_generator(I, phi):
    # number of basis
    N = len(I)
    M = N + 1
    # to collect f_n_hat coefficients
    f_n_hat = []
    x_1, w_1 = lg.leggauss(M)
    x_2, w_2 = her.hermgauss(M)
    for n in range(0, N):
        phi_n = phi[n]
        # Quadrature calculation
        c_i = 0
        for i in range(N):
            for j in range(N):
                c_i += w_1[i] * w_2[j] * f(x_1[i], x_2[j]) * phi_n(x_1[i], x_2[j])
        g_i = sum([lg.Legendre.basis(I[n][0])(x_1[j])**2 * w_1[j] for j in range(N)]) * \
            sum([her.Hermite.basis(I[n][1])(x_2[j])**2 * w_2[j] for j in range(N)])
        # Coefficient added to list
        f_n_hat += [c_i / g_i]
    # Returns our function ready for use by calling it just like f
    return lambda z_1, z_2: \
        sum([f_n * phi_def(i_1, i_2)(z_1, z_2) for f_n, (i_1, i_2) in zip(f_n_hat, I)])

Pnf_1 = Pnf_generator(I_1, phi_1)
Pnf_2 = Pnf_generator(I_2, phi_2)
```

Code 6: Plotting data setup

```
# Plotting space
t_1 = np.arange(-1, 1, 0.01)
t_2 = np.arange(-1, 1, 0.01)
X, Y = np.meshgrid(t_1, t_2)
Z_pnf_1 = Pnf_1(X, Y)
Z_pnf_2 = Pnf_2(X, Y)
Z_f = f(X, Y)
```

Code 7: Plot $P_N f_1$

```
fig = plt.figure(figsize=(10, 6), dpi=80)
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z_f, cmap='viridis')
ax.plot_surface(X, Y, Z_pnf_1, cmap='plasma')
ax.set_xlabel('Z_1')
ax.set_ylabel('Z_2')
ax.set_zlabel('f')
plt.title('Plot f and Pnf_1')
fig.show()
```

Code 8: Plot $P_N f_2$

```
fig = plt.figure(figsize=(10, 6), dpi=80)
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z_f, cmap='viridis')
ax.plot_surface(X, Y, Z_pnf_2, cmap='plasma')
ax.set_xlabel('Z_1')
ax.set_ylabel('Z_2')
ax.set_zlabel('f')
plt.title('Plot f and Pnf_2')
fig.show()
```

Code 9: Monte-Carlo sampling for mean and variance estimation

```
N_samples = 1_000_000
# We use our modified Z_1 re-adjusted in f
Z_1 = np.random.uniform(-1, 1, N_samples)
Z_2 = np.random.normal(1, 1, N_samples)
Z = f(Z_1, Z_2)
mean = np.mean(Z)
var = np.var(Z)
print(f'Mean: {mean}\nVar: {var}')
```