Lab 6 Plots

February 26, 2018

```
In [11]: import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd

         % matplotlib inline
```

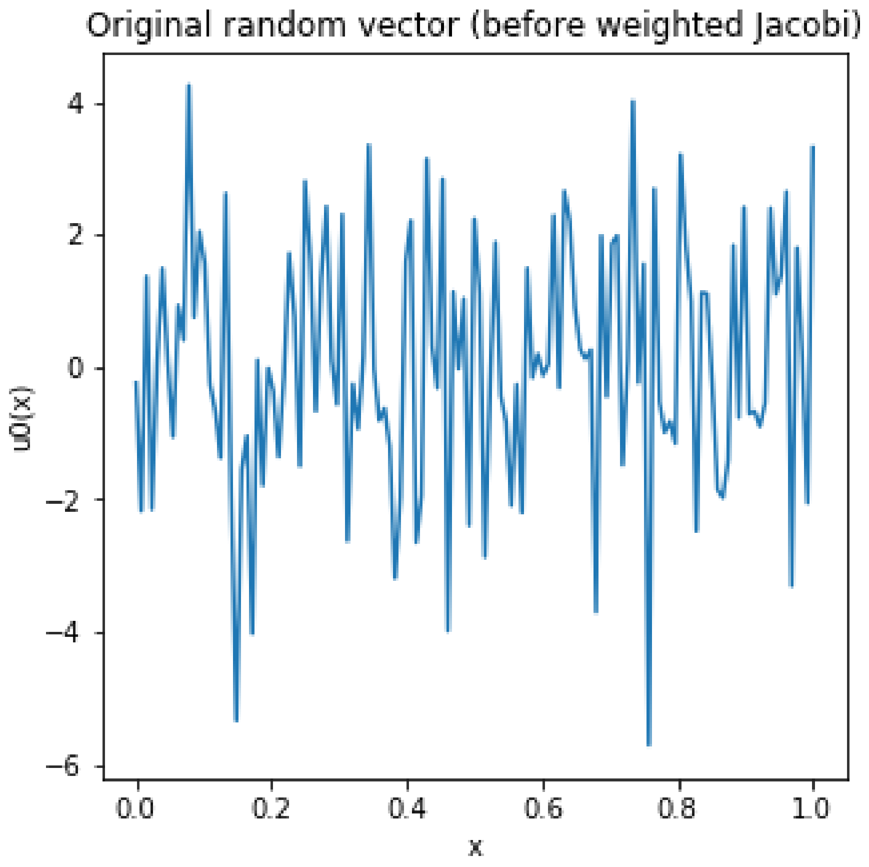
1 Lab 6: Finite Elements (Plots)

1.1 Deliverable 1: Weighted Jacobi

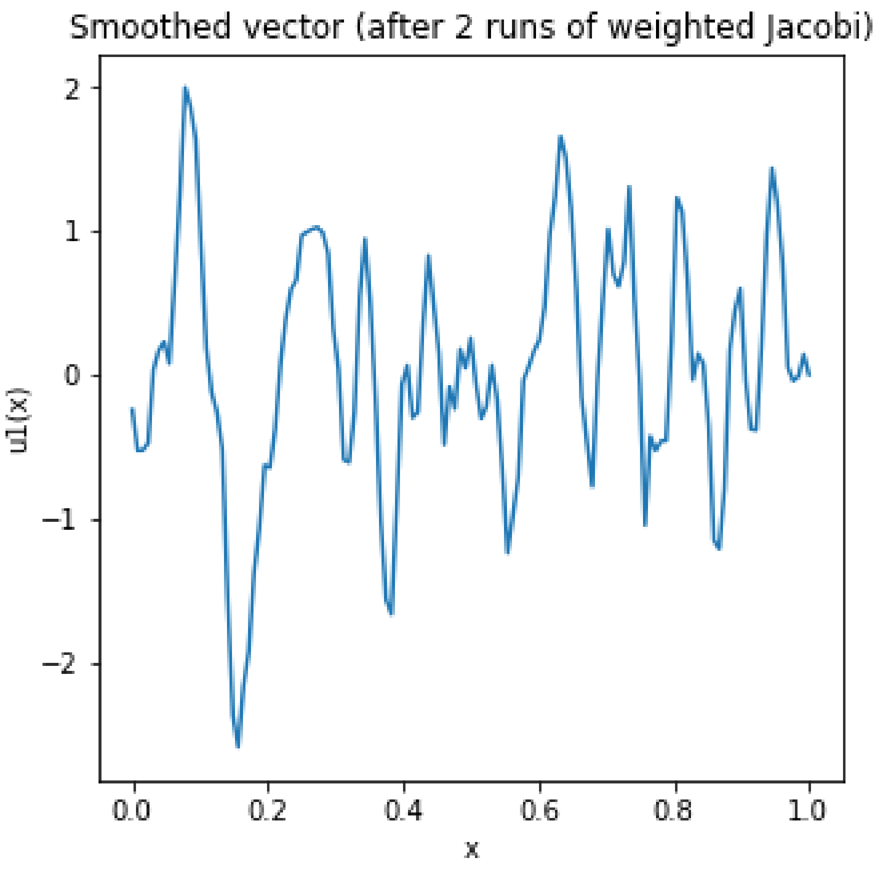
```
In [12]: data = np.loadtxt("u0_d1.txt")
         data1 = np.loadtxt("u1_d1.txt")

In [13]: t = np.linspace(0, 1, len(data))
         t1 = np.linspace(0, 1, len(data1))

In [30]: fig1 = plt.figure(figsize = (5, 5))
         plt.plot(t, data)
         plt.xlabel("x")
         plt.ylabel("u0(x)")
         plt.title("Original random vector (before weighted Jacobi)")
         plt.savefig("del1.png")
```



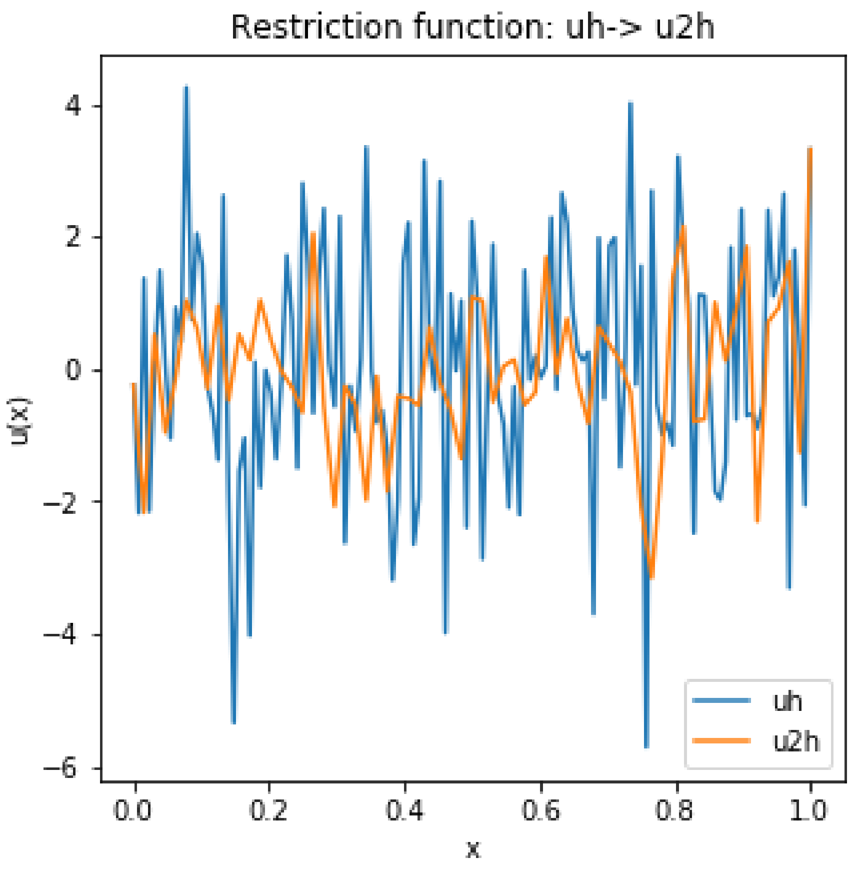
```
In [31]: fig2 = plt.figure(figsize = (5, 5))
         plt.plot(t1, data1)
         plt.xlabel("x")
         plt.ylabel("u1(x)")
         plt.title("Smoothed vector (after 2 runs of weighted Jacobi)")
         plt.savefig("del1_pt2.png")
```



1.2 Deliverable 2: Restriction

```
In [16]: data2 = np.loadtxt("u2h_d2.txt")
         t2 = np.linspace(0, 1, len(data2))
```

```
In [34]: fig3 = plt.figure(figsize= (5, 5))
         plt.plot(t, data, label= "uh")
         plt.plot(t2, data2, label = "u2h")
         plt.legend()
         plt.xlabel("x")
         plt.ylabel("u(x)")
         plt.title("Restriction function: uh-> u2h")
         plt.savefig("del2.png")
```

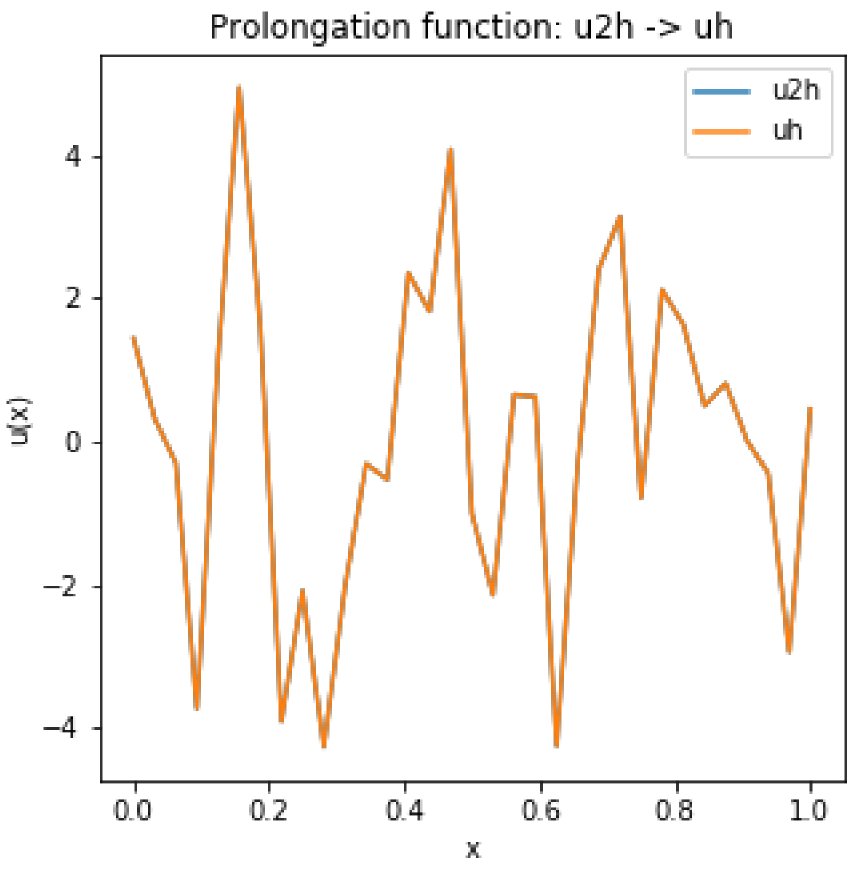


1.3 Deliverable 3: Prolongation

```
In [18]: data3 = np.loadtxt("uh_d3.txt")
         data4 = np.loadtxt("u2h_d3.txt")

In [19]: t3 = np.linspace(0, 1, len(data3))
         t4 = np.linspace(0, 1, len(data4))

In [35]: fig3 = plt.figure(figsize=(5, 5))
         plt.plot(t4, data4, label = "u2h")
         plt.plot(t3, data3, label= "uh")
         plt.legend()
         plt.xlabel("x")
         plt.ylabel("u(x)")
         plt.title("Prolongation function: u2h -> uh")
         plt.savefig("del3.png")
```



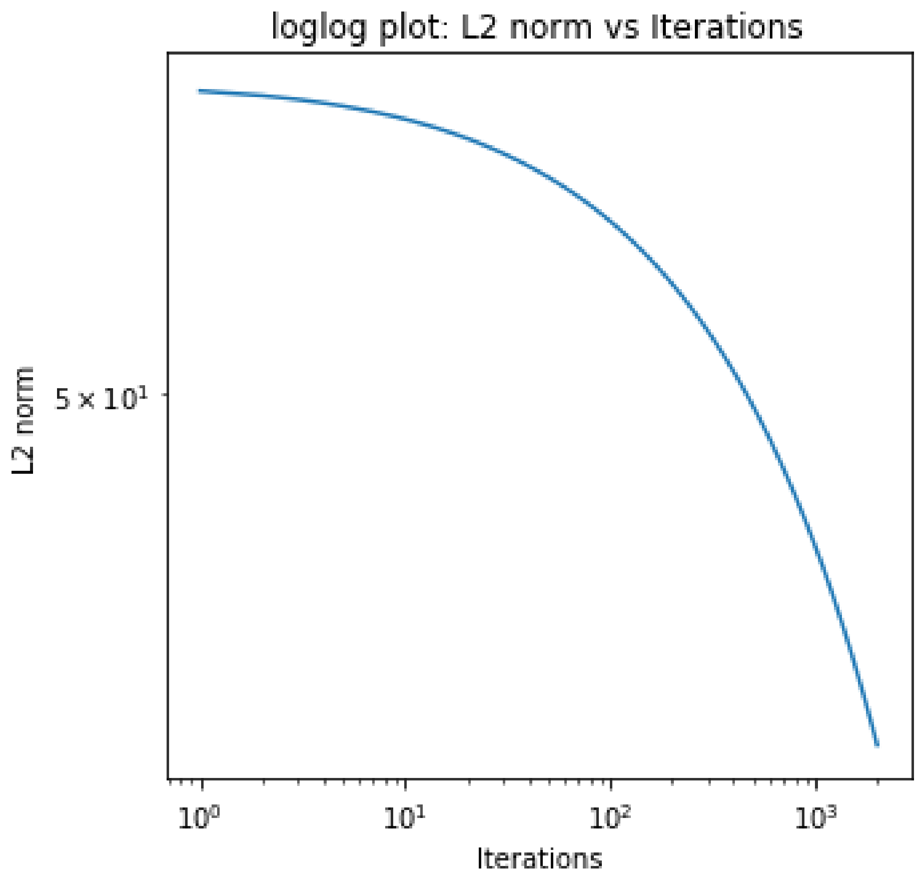
1.4 Deliverable 4: Thomas Solver

This is covered in the output of the program, but we reach the solution of $x = [1, 1, ..., 1]^T$

1.5 Task 1(or 5): weighted jacobi

```
In [21]: data5 = pd.read_csv("jacobi_1.csv", delimiter=";")
```

```
In [36]: fig5 = plt.figure(figsize=(5, 5))
         plt.loglog(data5['Iteration'], data5[' L2 norm '])
         plt.xlabel("Iterations")
         plt.ylabel("L2 norm")
         plt.title("loglog plot: L2 norm vs Iterations")
         plt.savefig("task5.png")
```



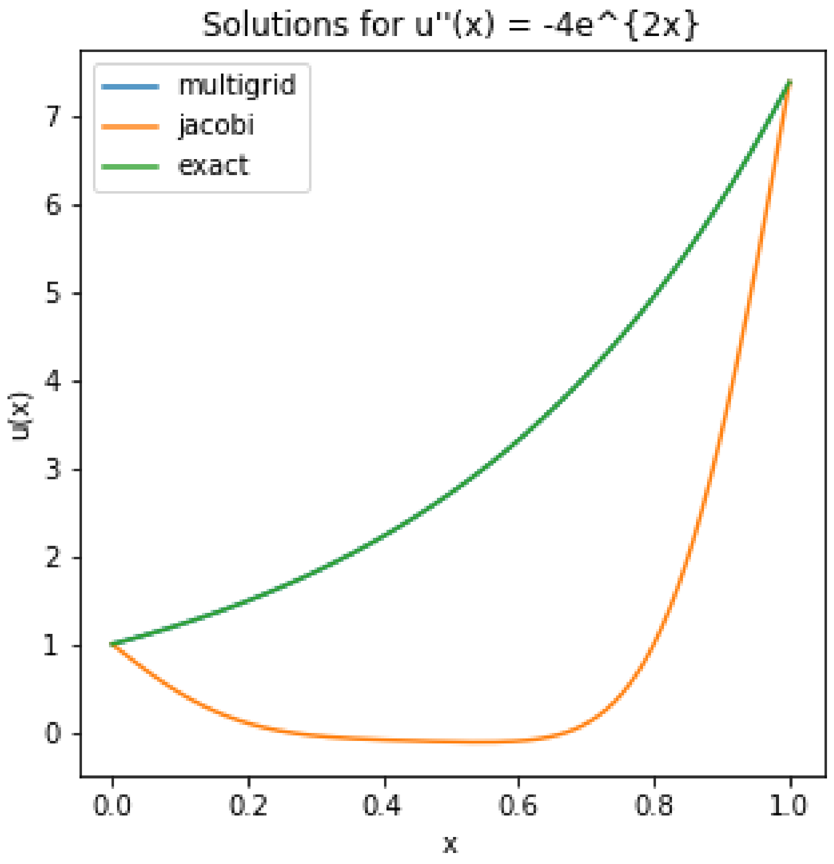
1.6 Task 2 (or 6): Multigrid

```
In [24]: data6 = np.loadtxt("u1h_multigrid1.txt")
         data7 = np.loadtxt("u1h_jacobi1.txt")

In [25]: def f(x):
             return np.exp(2*x)

In [37]: fig6 = plt.figure(figsize=(5, 5))
         x = np.linspace(0, 1, len(data6))
         plt.plot(x, data6, label = "multigrid")
         plt.plot(x, data7, label = "jacobi")
         plt.plot(x, f(x), label = "exact")
         plt.legend()

         plt.xlabel("x")
         plt.ylabel("u(x)")
         plt.title("Solutions for u''(x) = -4e^{2x}")
         plt.savefig("task6.png")
```



1.7 Task 3 (or 7): Computational Costs

Unfortunately I was unable to complete this task because my code would output either 1 or 10000 for each tolerance of iteration