

Assignment 4.

Jithin D. George, No. 1622555

Due Monday, Feb. 5.

1. (finite elements) Use the Galerkin finite element method with continuous piecewise linear basis functions to solve the problem

$$-\frac{d}{dx} \left((1+x^2) \frac{du}{dx} \right) = f(x), \quad 0 \leq x \leq 1,$$

$$u(0) = 0, \quad u(1) = 0.$$

- (a) Derive the matrix equation that you will need to solve for this problem.

Solution:

$$\hat{u} = \sum_{j=1}^{n-1} c_j \psi_j$$

$$\begin{pmatrix} a_{ii} & a_{ii+1} & & \\ a_{ii-1} & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & a_{ii-1} & a_{ii} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \langle f, \psi_1 \rangle \\ \vdots \\ \vdots \\ \langle f, \psi_{n-1} \rangle \end{pmatrix}$$

$$\begin{aligned} a_{ii} &= \int_0^1 (1+x^2) (\psi'_i)^2 dx \\ &= \int_{x_{i-1}}^{x_i} (1+x^2) \frac{1}{(x_i - x_{i-1})^2} dx + \int_{x_i}^{x_{i+1}} (1+x^2) \frac{1}{(x_{i+1} - x_i)^2} dx \\ &= \frac{1}{x_i - x_{i-1}} + \frac{x_i^3 - x_{i-1}^3}{3(x_i - x_{i-1})^2} + \frac{1}{x_{i+1} - x_i} + \frac{x_{i+1}^3 - x_i^3}{3(x_{i+1} - x_i)^2} \end{aligned}$$

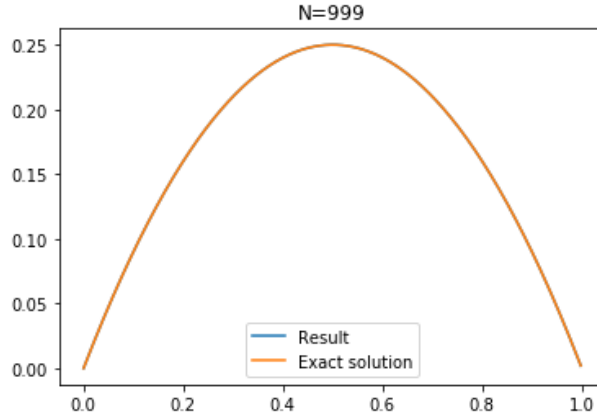
$$\begin{aligned} a_{ii-1} &= \int_0^1 (1+x^2) \psi_i \psi_{i-1} dx \\ &= - \int_{x_{i-1}}^{x_i} (1+x^2) \frac{1}{(x_i - x_{i-1})^2} dx \\ &= - \frac{1}{x_i - x_{i-1}} - \frac{x_i^3 - x_{i-1}^3}{3(x_i - x_{i-1})^2} \end{aligned}$$

$$\begin{aligned} a_{ii+1} &= \int_0^1 (1+x^2) \psi_i \psi_{i+1} dx \\ &= - \frac{1}{x_{i+1} - x_i} - \frac{x_{i+1}^3 - x_i^3}{3(x_{i+1} - x_i)^2} \end{aligned}$$

$$\langle f, \psi_i \rangle = \int_0^1 f \psi_i dx = \int_{x_{i-1}}^{x_i} f \psi_i dx + \int_{x_i}^{x_{i+1}} f \psi_i dx$$

- (b) Write a code to solve this set of equations. You can test your code on a problem where you know the solution by choosing a function $u(x)$ that satisfies the boundary conditions and determining what $f(x)$ must be in order for $u(x)$ to satisfy the differential equation. Try $u(x) = x(1 - x)$. Then $f(x) = 2(3x^2 - x + 1)$.

Solution:



```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 6*x**2-2*x+2
def xf(x):
    return 6*x**3-2*x**2+2*x
def intf(x):
    return 2*x**3-x**2+2*x
def intxf(x):
    return (6/4)*x**4-(2/3)*x**3+x**2
def fmaker(x,i):
    return (x[i+1]/(x[i+1]-x[i]))*(intf(x[i+1])-intf(x[i]))
        - (x[i-1]/(x[i]-x[i-1]))*(intf(x[i])-intf(x[i-1]))
        +(1/(x[i]-x[i-1]))*(intxf(x[i])-intxf(x[i-1])) -(1/(x
        [i+1]-x[i]))*(intxf(x[i+1])-intxf(x[i]))
def aij(x,i):
    return 1/(x[i]-x[i-1]) +(x[i]**3-x[i-1]**3)/(3*(x[i]-x[i
    -1])**2)
def aii(x,i):
    return aij(x,i)+aij(x,i+1)
def tridiag(a, b, c, k1=-1, k2=0, k3=1):
    return np.diag(a, k1) + np.diag(b, k2) + np.diag(c, k3)
N=1000
x=np.array([])
```

```

for i in range(N):
    x= np.append(x,(i/(N-1))**2)
#x=np.linspace(0,1,N)
fe = np.array([])
aiivec = np.array([])
aijvec = np.array([])
for i in range(1,len(x)-1):
    fe=np.append(fe,fmaker(x,i))
    aiivec =np.append(aiivec,aii(x,i))
    aijvec =np.append(aijvec,-aij(x,i))
mat2=tridiag(aijvec[1:],aiivec,aijvec[1:])
z=np.zeros(N-2)
k = np.linalg.solve(mat2,fe)
y=x[1:-1]
plt.plot(y,k, label = "Result")
exact=y*(1-y)
plt.plot(y,exact, label = "Exact solution")
plt.legend()
print(np.linalg.norm(k-exact, np.inf))
plt.title("N="+str(N))

```

- (c) Try several different values for the mesh size h . Based on your results, what would you say is the order of accuracy of the Galerkin method with continuous piecewise linear basis functions?

Solution:

The infinity norm is used here.

h	Error
0.01	3.2112324338e-06
0.001	3.1537675276e-08

The order seems to be $O(h^2)$

- (d) Now try a nonuniform mesh spacing, say, $x_i = (i/(m+1))^2$, $i = 0, 1, \dots, m+1$. Do you see the same order of accuracy, if h is defined as the maximum mesh spacing, $\max_i(x_{i+1} - x_i)$?

Solution:

The infinity norm is used here.

h	Error
0.02	1.23156638885e-05
0.002	1.20969674222e-07

Again, the order seems to be $O(h^2)$.

- (e) Suppose the boundary conditions were $u(0) = a$, $u(1) = b$. Show how you would represent the approximate solution $\hat{u}(x)$ as a linear combination of hat functions and how the matrix equation in part (a) would change.

Solution:

$$\hat{u} = \sum_{j=0}^n c_j \psi_j$$

$$\psi_0(x) = \begin{cases} \frac{x_1-x}{x_1}, & \text{for } 0 \leq x \leq x_1 \\ 0, & \text{elsewhere} \end{cases}$$

$$\psi_n(x) = \begin{cases} \frac{x-x_{n-1}}{x_n-x_{n-1}}, & \text{for } x_{n-1} \leq x \leq x_n \\ 0, & \text{elsewhere} \end{cases}$$

$$\begin{pmatrix} 1 & & & & & & \\ a_{ii-1} & a_{ii} & a_{ii+1} & & & & \\ & a_{ii-1} & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{ii-1} & a_{ii} & a_{ii+1} & \\ & & & & & & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} a \\ \langle f, \psi_1 \rangle \\ \vdots \\ \vdots \\ \langle f, \psi_{n-1} \rangle \\ b \end{pmatrix}$$

2. (spectral methods, **chebfun**) Download the package **chebfun** www.chebfun.org eggs from www.chebfun.org. This package works with functions that are represented (to machine precision) as sums of Chebyshev polynomials. It can solve 2-point boundary value problems using spectral methods. Use **chebfun** to solve the same problem as in the previous exercise and check the L_2 -norm and the ∞ -norm of the error.

Solution:

```

1 L = chebop(0, 1);
2 L.op = @(x,u)- diff((1+x.^2).* diff(u,1),1) ;
3 L.lbc = 0; L.rbc = 0;
4 x = chebfun('x', [0, 1]);
5 f = 2*(3*x.^2-x+1);
6 u = L\f;
7 LW = 'linewidth'; lw = 0.6;
8 plot(u, 'm', LW, lw)
9 w=x.*(1-x);
10
11 norm(w-u, Inf)
12 norm(w-u, 2)

```

L_2 norm : 2.3919e-15

∞ norm : 3.2752e-15

Spectral methods seem to give near machine precision results.