AMATH 585 Assignment 4

Tyler Chen

Problem 1 (finite elements)

Use the Galerkin finite element method with continuous piecewise linear basis functions to solve the problem

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

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

(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 $f(x) = 2(3x^2 - x + 1)$.

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

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

(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

(a) Define,

$$\mathcal{L} = -\frac{d}{dx}\left((1+x^2)\frac{d}{dx}\right)$$

We seek a solution to $\mathcal{L}u = f$ subject to $u(0) = u(1) = 0$.

Define $\langle \cdot, \cdot \rangle$ as $\langle f, g \rangle = \int_0^1 f(x)g(x)dx$.

Suppose u satisfies $\mathcal{L}u = f$, $u(0) = u(1) = 0$. Then,

$$\langle \mathcal{L}u, \varphi \rangle = \int_0^1 \mathcal{L}u\varphi(x)dx = \int_0^1 f(x)\varphi(x) = \langle f, \varphi \rangle$$

for all functions φ.

First note that, integrating by parts we have,

$$\langle \mathcal{L}u, \varphi \rangle = \int_0^1 -\frac{d}{dx}\left((1+x^2)\frac{du}{dx}\right)\varphi(x)dx = \left[-\left((1+x^2)\frac{du}{dx}\right)\varphi(x)\right]_0^1 - \int_0^1 (1+x^2)\frac{du}{dx}\varphi'(x)dx$$

Assuming du/dx is "well behaved" and $\varphi(0) = \varphi(1) = 0$, we can simplify the above expression to,

$$\langle \mathcal{L}u, \varphi \rangle = -\int_0^1 (1+x^2)u'(x)\varphi'(x)dx$$

For $i = 1, 2, \dots, m$ define

$$\varphi_i(x) = \begin{cases} (x - x_{i-1})/(x_i - x_{i-1}) & x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/(x_{i+1} - x_i) & x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

Then $\{\varphi_i(x)\}_{i=1}^m$ form a basis for $S = \langle \varphi : \varphi \text{ picewise linear continuous with } \varphi(0) = \varphi(1) = 0\}$.
Suppose $\hat{u} \in S$. Then, for some $c_1, c_2, \dots, c_n$ we can write,

$$\hat{u} = \sum_{i=1}^m \varphi_i(x)$$

Suppose $\hat{u}$ satisfies,

$$\langle \mathcal{L}\hat{u}, \varphi_i \rangle = \langle f, \varphi_i \rangle, \qquad \forall i = 1, 2, \dots, m$$

Then, for any $\varphi = \sum_{i=1}^n d_i \varphi_i \in S$, we have,

$$\langle \mathcal{L}\hat{u} - f, \varphi \rangle = \langle \mathcal{L}\hat{u} - f, \sum_{i=1}^n d_i \varphi_i \rangle = \sum_{i=1}^n d_i \langle \mathcal{L}\hat{u} - f, \varphi_i \rangle = 0$$

This means for any $\varphi \in S$ we will satisfy the weak form of our differential equation.
For each $i = 1, 2, \dots, n$ we have,

$$\langle \mathcal{L}\hat{u}, \varphi_i \rangle = -\int_0^1 (1 + x^2) \hat{u}'(x) \varphi_i'(x) dx$$

Since $\hat{u}$ is a linear combination of n basis functions so is $\hat{u}'$. This gives a linear system,

$$A\vec{c} = \vec{f}$$

where c_i is the i-th coefficient of our decomposition of $\hat{u}$, and $f_i = \langle f, \varphi_i \rangle$.
Writing out the matrix A we have,

$$a_{ij} = \langle \mathcal{L}\varphi_j, \varphi_i \rangle = \int_0^1 (1 + x^2) \varphi_j'(x) \varphi_i'(x) dx$$

Clearly $\varphi_j'(x)\varphi_i'(x) = 0$ if $j \notin \{i-1, i, i+1\}$ so the matrix is tridiagonal. It is also clear that the matrix is symmetric.
Write,

$$P(x) := \int (1 + x^2) dx = x + x^3/3$$

Thus,

$$a_{i,i} = \int_{x_{i-1}}^{x_i} \frac{1 + x^2}{(x_i - x_{i-1})^2} dx + \int_{x_i}^{x_{i+1}} \frac{1 + x^2}{(x_{i+1} - x_i)^2} dx = \frac{P(x_i) - P(x_{i-1})}{(x_i - x_{i-1})^2} + \frac{P(x_{i+1}) - P(x_i)}{(x_{i+1} - x_i)^2}$$

$$a_{i,i-1} = -\int_{x_{i-1}}^{x_i} \frac{1 + x^2}{(x_i - x_{i-1})^2} dx = -\frac{P(x_i) - P(x_{i-1})}{(x_i - x_{i-1})^2}$$

Similarly, given $f(x)=2(3x^2-x+1)$, write,

$$F(x):=\int f(x)dx = 2x^3-x^2+2x \qquad G(x):=\int xf(x)dx = 3x^4/2-2x^3/3+x^2$$

Now, for $i=1,...,m$, we compute,

$$\begin{aligned}\langle f,\varphi_i\rangle &= \int_{x_{i-1}}^{x_i} f(x)\frac{x-x_{i-1}}{x_i-x_{i-1}}dx + \int_{x_i}^{x_{i+1}} f(x)\frac{x_{i+1}-x}{x_{i+1}-x_i}dx \\ &= \frac{G(x_i)-G(x_{i-1})-x_{i-1}(F(x_i)-F(x_{i-1}))}{x_i-x_{i-1}} + \frac{x_{i+1}(F(x_{i+1})-F(x_i))-(G(x_{i+1})-G(x_i))}{x_{i+1}-x_i}\end{aligned}$$

This is useful given a specific f on the right hand side (maybe we are solving for some physical system). However, if we would like to solve $\mathcal{L}u=f$ for arbitrary f then we should integrate numerically. Using the midpoint rule we have,

$$\begin{aligned}\langle f,\varphi_i\rangle &\approx (x_i-x_{i-1})\frac{f(x_{i-1/2})(x_{i-1/2}-x_{i-1})}{x_i-x_{i-1}} + (x_{i+1}-x_i)\frac{f(x_{i+1/2})(x_{i+1}-x_{i+1/2})}{x_{i+1}-x_{i+1/2}} \\ &= f(x_{i-1/2})(x_{i-1/2}-x_{i-1}) + f(x_{i+1/2})(x_{i+1}-x_{i+1/2})\end{aligned}$$

On a given interval the midpoint rule is $\mathcal{O}(h^3)$ so the order of our error is unaffected.

In the case of a uniform mesh these equations can be simplified some as $x_{i+1}-x_i=h$ for all i.

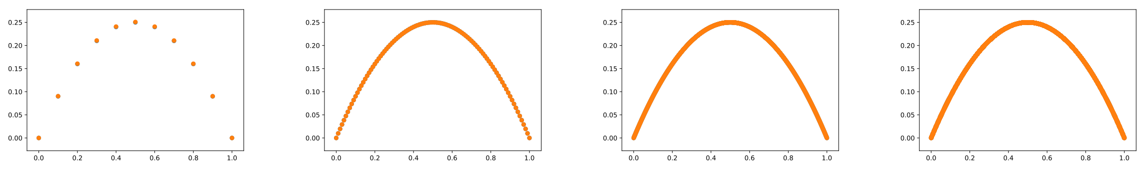
(b) We implement the finite element method in Python. Note that for the uniform grid, the difference between consecutive nodes is always h so it would be appropriate to replace any instance of the difference with h. However, for convenience, so that the code is usable for non-uniform meshes we have not implemented it in this way. We also implement numerical integration using the midpoint rule, however we have tested it with the analytic solution described above and it works to the same level of accuracy.

```
# define grid mesh types
grids = {"uniform": lambda m: np.linspace(0,1,m+2),
         "non-uniform": lambda m: (np.arange(0,m+2)/(m+1))**2
         }

# problem parameters
def rhs(x): return 2*(3*x**2-x+1)
a = 0
b = 0

def P(x): return x+x**3/3
for grid_type in ["uniform", "non-uniform"]:
    error = np.zeros((0,3))
    for e in [1,2,3,4]:

        # generate mesh
        m = 10**e-1
        x = grids[grid_type](m)
        d = x[1:m+2]-x[0:m+1]
        d2 = d**2
        h = np.max(d)
        xmid = (x[1:]+x[:m+1])/2
        dxmid = xmid-x[0:m+1]

        # generate A
        A = np.zeros((m,m))
        off_diag = - (P(x[1:m+1])-P(x[0:m])) / d2[0:m]
        main_diag = (P(x[1:m+1])-P(x[0:m])) / d2[0:m]
        main_diag += (P(x[2:m+2])-P(x[1:m+1])) / d2[1:m+1]

        A += np.diag(main_diag)
        A += np.diag(off_diag[1:],1) + np.diag(off_diag[1:],-1)

        # generate f with midpoint rule
        fmid = rhs(xmid)
        f = fmid[0:m]*(dxmid[0:m]) + fmid[1:m+1]*(dxmid[1:m+1])

        f[0] += a * (P(x[1])-P(x[0])) / (x[1]-x[0])**2
        f[m-1] += b * (P(x[m+1])-P(x[m])) / (x[m+1]-x[m])**2

        #solve
        U = np.zeros(m+2)
        U[1:m+1] = np.linalg.solve(A,f)
        U[0] = a
        U[m+1] = b

        error = np.append(error,[[h, np.sqrt(np.sum((U - x*(1-x))[1:m+2]*d[0:m
            +1])**2), np.max(np.abs( U - x*(1-x) ))]],axis=0)

        plt.figure()
        plt.scatter(x,x*(1-x))
        plt.scatter(x,U)
        plt.savefig('img/1/'+grid_type+'_'+str(m+1)+'.pdf')

    np.savetxt(grid_type+'.txt',error,delimiter=' & ', newline=' \\\\ \hline \
        n')
```

(c) We iterate over a few values of mesh size. The results are shown in Table 1. Based on these results it appears that the accuracy is order h^2, where h where is the (maximum) mesh spacing.

$\max_i\{x_{i+1}-x_i\}$	L_2 norm of error	infinity norm of error
1.000000000000000888e-01	5.091122389934837900e-04	7.812018813435628317e-04
1.000000000000000888e-02	5.138595328954780000e-06	7.868616310829912308e-06
1.000000000000000888e-03	5.139064257669400952e-08	7.868630846896706998e-08
1.000000000001000089e-04	5.070166105001218434e-10	7.758830244952719113e-10

Table 1: error vs. mesh size on uniform mesh (actual solution in blue)



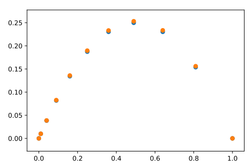
(a) $m+1=10$ (b) $m+1=100$ (c) $m+1=1000$ (d) $m+1=10000$

Figure 1: Solution on uniform grid

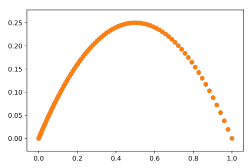
(d) We iterate over a few values of mesh size using a non-uniform mesh. The results are shown in Table 1. Based on these results it appears that the accuracy is order h^2, where h is the maximum mesh spacing.

$\max_i\{x_{i+1} - x_i\}$	L_2 norm of error	infinity norm of error
1.8999999999999999467e-01	1.9169527773882908 72e-03	3.2693499644591839 10e-03
1.9900000000000002878e-02	2.1167189915644363 71e-05	3.3145534149653377 30e-05
1.9989999999999973021e-03	2.1321303663102053 02e-07	3.3148657829773497 95e-07
1.9998999999999277256e-04	2.1325605029109202 09e-09	3.3135685861829244 91e-09

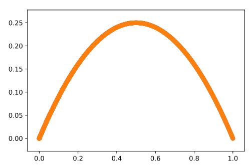
Table 2: error vs. mesh size on non-uniform mesh (actual solution in blue)



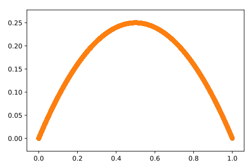
(a) $m+1 = 10$



(b) $m+1 = 100$



(c) $m+1 = 1000$



(d) $m+1 = 10000$

Figure 2: Solution on non-uniform grid

(e) The basis for S defined above is no longer a basis if we allow the functions in S to be nonzero at the endpoints. We must extend our original basis to a new basis. A simple way to do this would be to define,

$$\varphi_0 = \begin{cases} (x_1 - x)/x_1 & x \in [0, x_1] \\ 0 & \text{otherwise} \end{cases} \qquad \varphi_{m+1} = \begin{cases} (x - x_m)/(x_{m+1} - x_m) & x \in [x_n, x_{n+1}] \\ 0 & \text{otherwise} \end{cases}$$

If $u(0) = a$ and $u(1) = b$ we can now write,

$$\hat{u} = \sum_{i=0}^{m+1} c_i \varphi_i(x)$$

where $c_0 = a$ and $c_{m+1} = b$ are given/known.

The first and last equations in the matrix then become,

$$\begin{aligned} a\langle \mathcal{L}\varphi_0, \varphi_1\rangle + c_1\langle \mathcal{L}\varphi_1, \varphi_1\rangle + c_2\langle \mathcal{L}\varphi_2, \varphi_1\rangle &= \langle \mathcal{L}\hat{u}, \varphi_1\rangle = \langle f, \varphi_1\rangle \\ c_{m-1}\langle \mathcal{L}\varphi_{m-1}\varphi_m\rangle + c_m\langle \mathcal{L}\varphi_{m1}\varphi_m\rangle + b\langle \mathcal{L}\varphi_{m+1}\varphi_m\rangle &= \langle \mathcal{L}\hat{u}, \varphi_m\rangle = \langle f, \varphi_m\rangle \end{aligned}$$

Then, we can leave A unchanged and we modify f by subtracting, $a\langle \mathcal{L}\varphi_0, \varphi_1\rangle$ from the first row, and $b\langle \mathcal{L}\varphi_{m+1}\varphi_m\rangle$ from the last row.

Explicitly we have,

$$\begin{aligned} a\langle \varphi_0, \varphi_1\rangle &= a\int_{x_0}^{x_1} \frac{1+x^2}{(x_1 - 0)^2}dx = a\frac{P(x_1) - P(x_0)}{(x_1 - x_0)^2} = a\frac{P(x_1)}{x_1^2} \\ b\langle \varphi_{m+1}, \varphi_m\rangle &= b\int_{x_m}^{x_{m+1}} \frac{1+x^2}{(1 - x_m)^2}dx = b\frac{P(1) - P(x_m)}{(1 - x_m)^2} = b\frac{4/3 - P(x_m)}{(1 - x_m)^2} \end{aligned}$$

Problem 1 (spectral methods, `chebfun`)

Download the package `chebfun` 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

We implement this in MATLAB):

```
addpath(fullfile(cd,'chebfun'))

d = domain(0,1);
L = chebop(@(x,u) -diff((1+x.^2).*diff(u)),d,0,0);
f = chebfun(@(x) 2*(3*x.^2-x+1),d);
u = L\f;

u_true = chebfun(@(x) x.*(1-x),d);

norm(u-u_true)
norm(u-u_true,'inf')
```

This gives errors 3.3859×10^{-15} and 4.6952×10^{-15} for the L_2 and infinity norms respectively.

Note that according to the Chebfun guide the norm function returns "the square root of the integral of the square of the absolute value over the region of definition". This is the L_2 norm we are interested in.