Sample Solutions for Assignment 4.

Reading: Chapters 2 and 3 in the text plus supplemental material on finite elements.

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

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

\[ u(0) = 0, \ u(1) = 0. \]

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

Let \( \varphi_1(x), \ldots, \varphi_m(x) \) be the piecewise linear basis functions associated with each interior node \( x_1, \ldots, x_m \). Note that the true solution \( u(x) \) satisfies

\[ \int_0^1 \frac{d}{dx} \left( (1 + x^2) \frac{du}{dx} \right) \varphi_i(x) \, dx = \int_0^1 f(x) \varphi_i(x) \, dx, \ i = 1, \ldots, m, \]

and, after integrating by parts, this becomes

\[ -\int_0^1 (1 + x^2) u'(x) \varphi_i'(x) \, dx = \int_0^1 f(x) \varphi_i(x) \, dx, \ i = 1, \ldots, m. \] (1)

Write the approximate solution \( \hat{u}(x) \) as \( \sum_{j=1}^m c_j \varphi_j(x) \), and choose \( c_1, \ldots, c_m \) so that equation (1) holds for \( \hat{u} \):

\[ -\sum_{j=1}^m c_j \int_0^1 (1 + x^2) \varphi_j'(x) \varphi_i'(x) \, dx = \int_0^1 f(x) \varphi_i(x) \, dx, \ i = 1, \ldots, m. \] (2)

Note that

\[ \varphi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & x_{i-1} \leq x \leq x_i \\ \frac{x_{i+1}-x}{x_{i+1}-x_{i+1}} & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \]

\[ \varphi_i'(x) = \begin{cases} \frac{1}{x_i-x_{i-1}} & x_{i-1} \leq x \leq x_i \\ \frac{-1}{x_{i+1}-x_i} & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \]

Hence equation (2) can be written in the form \( A \mathbf{u} = \mathbf{f} \), where \( A \) is tridiagonal and

\[ A_{ii} = -\frac{1}{(x_i-x_{i-1})^2} \int_{x_{i-1}}^{x_i} (1 + x^2) \, dx - \frac{1}{(x_{i+1}-x_i)^2} \int_{x_i}^{x_{i+1}} (1 + x^2) \, dx \]

\[ = -\frac{1}{x_i-x_{i-1}} \left[ 1 + \frac{1}{3} (x_i^2 + x_i x_{i-1} + x_{i-1}^2) - \frac{1}{x_{i+1}-x_i} \left[ 1 + \frac{1}{3} (x_{i+1}^2 + x_i x_{i+1} + x_{i+1}^2) \right] \right], \]
\[
A_{i+1} = A_{i+1,i} = \frac{1}{(x_{i+1} - x_i)^2} \int_{x_i}^{x_{i+1}} (1+x^2) \, dx = \frac{1}{x_{i+1} - x_i} \left[ 1 + \frac{1}{3} (x_i^2 + x_i x_{i+1} + x_{i+1}^2) \right],
\]

\[
f_i = \int_{x_{i-1}}^{x_i} f(x) \frac{x - x_{i-1}}{x_i - x_{i-1}} \, dx + \int_{x_i}^{x_{i+1}} f(x) \frac{x_{i+1} - x}{x_{i+1} - x_i} \, dx.
\]

(b) Write a MATLAB 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 \( f(x) = -2(3x^2-x+1) \).

See Matlab code steady1dfem.m.

(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?

Following is a table of values of the \( L_2 \)-norm of the error, for different mesh sizes \( h \). The method is clearly second order accurate.

<table>
<thead>
<tr>
<th>( h )</th>
<th>( L_2 )-norm of error</th>
<th>ratio ( \text{err}(2h)/\text{err}(h) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10</td>
<td>( 5.665e-4 )</td>
<td>( 4.0011 )</td>
</tr>
<tr>
<td>1/20</td>
<td>( 1.416e-4 )</td>
<td>( 4.0003 )</td>
</tr>
<tr>
<td>1/40</td>
<td>( 3.54e-5 )</td>
<td>( 4.0001 )</td>
</tr>
<tr>
<td>1/80</td>
<td>( 8.8e-6 )</td>
<td>( 4.0001 )</td>
</tr>
</tbody>
</table>

(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, \( \max(x_{i+1} - x_i) \)?

See Matlab code fem_nonuniform.m (which also can do a uniform mesh).

I tried 10, 20, 40, and 80 subintervals and got the following results:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( h_{\max} )</th>
<th>( L_2 )-norm of error</th>
<th>( \text{err}/h_{\max}^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>.1900</td>
<td>( 2.0755e-3 )</td>
<td>.0575</td>
</tr>
<tr>
<td>20</td>
<td>.0975</td>
<td>( 5.7502e-4 )</td>
<td>.0605</td>
</tr>
<tr>
<td>40</td>
<td>.0494</td>
<td>( 1.4702e-4 )</td>
<td>.0603</td>
</tr>
<tr>
<td>80</td>
<td>.0248</td>
<td>( 3.6895e-5 )</td>
<td>.0598</td>
</tr>
</tbody>
</table>

Since the last column is nearly constant, the method still seems to be second order accurate.

2. (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 \( \infty \)-norm of the error.

The following commands:
d = domain(0,1) % Domain of problem
L = chebop(@(x,u) diff((1+x.^2).*diff(u,1),1),d,0,0) % Differential op and bcs
f = chebfun(@(x) -2*(3*x.^2-x+1),d) % Right-hand side
u = L\f % Solve the BVP
u_true = chebfun(@(x) x.*(1-x),d) % True soln to BVP
errL2 = norm(u-u_true,2) % 2-norm of difference
errInf = norm(u-u_true,Inf) % Max-norm of difference

produced the following results:
errL2 =
3.8084e-15
errInf =
6.3133e-15

3. Write a code to solve Poisson’s equation on the unit square with Dirichlet boundary conditions:
\[ u_{xx} + u_{yy} = f(x,y), \quad 0 < x, y < 1 \]
\[ u(x,0) = u(x,1) = u(0,y) = u(1,y) = 1. \]

Take \( f(x,y) = x^2 + y^2 \), and demonstrate numerically that your code achieves second order accuracy. [Note: If you do not know an analytic solution to a problem, one way to check the code is to solve the problem on a fine grid and pretend that the result is the exact solution, then solve on coarser grids and compare your answers to the fine grid solution. However, you must be sure to compare solution values corresponding to the same points in the domain.]

See the code poisson.m on the course web page.

I used a fine grid with \( h = 1/64 \) and then solved on grids with \( h = 1/32 \), \( h = 1/16 \), and \( h = 1/8 \). I looked at the maximum difference between the coarse grid values and the fine grid solution at the nodes of the coarse grid, and obtained the following results.

<table>
<thead>
<tr>
<th>h</th>
<th>maxerr</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.125e-02</td>
<td>4.595e-05</td>
<td></td>
</tr>
<tr>
<td>6.250e-02</td>
<td>2.288e-04</td>
<td>4.979e+00</td>
</tr>
<tr>
<td>1.250e-01</td>
<td>9.454e-04</td>
<td>4.132e+00</td>
</tr>
</tbody>
</table>

This looks essentially like second order accuracy. The error appears to be reduced by more than a factor of 4 in going from \( h = 1/16 \) to \( h = 1/32 \), but this may be because the computed solution with \( h = 1/32 \) is more like that with \( h = 1/64 \) (which is being used for comparison) than it is like the true solution. Probably the actual error reduction is closer to a factor of 4.
4. We have discussed using finite element methods to solve elliptic PDE’s such as

\[ \triangle u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial \Omega, \]

with homogeneous Dirichlet boundary conditions. How could you modify the procedure to solve the inhomogeneous Dirichlet problem:

\[ \triangle u = f \quad \text{in } \Omega, \quad u = g \quad \text{on } \partial \Omega, \]

where \( g \) is some given function? Derive the equations that you would need to solve to compute, say, a continuous piecewise bilinear approximation for this problem when \( \Omega \) is the unit square \( (0, 1) \times (0, 1) \).

Let the total number of interior nodes be \( N \) and the total number of boundary nodes be \( M \). Write the approximate solution as \( \hat{u}(x, y) = G(x, y) + \sum_{k=1}^{N+M} c_k \varphi_k(x, y) \), where \( \varphi_k \) is a bilinear basis function associated with interior node \( k \), and \( G \) matches or approximately matches \( g \) on \( \partial \Omega \); for example, if we number the boundary nodes as \( N + 1, \ldots, N + M \), then we could take \( G(x, y) \) to be the piecewise bilinear interpolant of \( g \) on \( \partial \Omega \):

\[ G(x, y) = \sum_{k=N+1}^{N+M} g_k \varphi_k(x, y), \]

where \( g_k \) is the value of \( g \) at boundary point \( k \). Choose the coefficients \( c_1, \ldots, c_N \) so that

\[ \int \int_{\Omega} (\hat{u}_{xx} + \hat{u}_{yy}) \varphi_\ell \, dx \, dy = \int \int_{\Omega} f \varphi_\ell \, dx \, dy, \quad \ell = 1, \ldots, N. \]

After using Green’s theorem, this becomes

\[ -\int \int_{\Omega} (\hat{u}_x \varphi_\ell_x + \hat{u}_y \varphi_\ell_y) \, dx \, dy + \int_{\partial \Omega} \hat{u}_n \varphi_\ell \, d\gamma = \int \int_{\Omega} f \varphi_\ell \, dx \, dy, \quad \ell = 1, \ldots, N. \]

Since each \( \varphi_\ell \), \( \ell = 1, \ldots, N \), associated with an interior node is 0 on the boundary, the boundary term vanishes and we are left with

\[ -\int \int_{\Omega} (\hat{u}_x \varphi_\ell_x + \hat{u}_y \varphi_\ell_y) \, dx \, dy = \int \int_{\Omega} f \varphi_\ell \, dx \, dy, \quad \ell = 1, \ldots, N. \]

Now substituting the expression for \( \hat{u} \), we can write

\[ -\int \int_{\Omega} \left[ \left( \sum_{k=1}^{N} c_k \varphi_{k,x} + \sum_{k=N+1}^{N+M} g_k \varphi_{k,x} \right) \varphi_\ell_x + \left( \sum_{k=1}^{N} c_k \varphi_{k,y} + \sum_{k=N+1}^{N+M} g_k \varphi_{k,y} \right) \varphi_\ell_y \right] \, dx \, dy = \]

\[ \int \int_{\Omega} f \varphi_\ell \, dx \, dy, \quad \ell = 1, \ldots, N. \]

Pulling sums outside the integrals and bringing known terms to the right-hand side, this becomes

\[ -\sum_{k=1}^{N} c_k \int \int_{\Omega} (\varphi_{k,x} \varphi_\ell_x + \varphi_{k,y} \varphi_\ell_y) \, dx \, dy = \]
\[
\int\int_{\Omega} \left[ f\varphi_{\ell} + \sum_{k=N+1}^{N+M} g_k (\varphi_{kx}\varphi_{\ell x} + \varphi_{ky}\varphi_{\ell y}) \right] \, dx \, dy, \quad \ell = 1, \ldots, N.
\]

The matrix is the same as for a standard bilinear finite element approximation for Poisson’s equation with Dirichlet boundary conditions:

\[
A_{\ell,k} = \int\int_{\Omega} (\varphi_{kx}\varphi_{\ell x} + \varphi_{ky}\varphi_{\ell y}) \, dx \, dy.
\]

The right-hand side is different for equations corresponding to nodes that are next to the boundary.