Jim Lambers
ENERGY 281
Spring Quarter 2007-08
Homework Assignment 4 Solution

1. Consider the boundary value problem given by the one dimensional advection-diffusion-reaction equation

$$\frac{d}{dx}\left(u - \frac{du}{dx}\right) = 0, \quad x \in [0,1],$$

with the boundary conditions

$$u(0) - \left.\frac{du}{dx}\right|_{x=0} = 1, \quad u(1) = 1.5.$$

(a) Obtain the weak form of the equation so that the space of trial and test functions are the same.

Solution Using integration by parts, we obtain

$$\begin{aligned} 0 &= \int_0^1 v(x)\frac{d}{dx}\left(u(x) - u'(x)\right)\, dx \\ &= \int_0^1 v(x)u'(x) - v(x)u''(x)\, dx \\ &= -v(x)u'(x)\big|_0^1 + \int_0^1 v(x)u'(x) + v'(x)u'(x)\, dx \\ &= v(0)u'(0) - v(1)u'(1) + \int_0^1 v(x)u'(x) + v'(x)u'(x)\, dx. \end{aligned}$$

(b) Consider a discretization of the domain in which the interval $[0,1]$ is divided up into N finite elements, each with length h. Using M generic basis functions $\phi_1, \phi_2, \ldots, \phi_M$, where the value of M depends on N and the boundary conditions, write down the Galerkin finite element equations, in terms of inner products of the basis functions and their derivatives.

Solution We define a grid of $N+1$ equally spaced points $x_1, \ldots, x_{N+1}$ where $x_i = (i-1)h$ with $h = 1/N$. Because we potentially have nonzero values at all $N+1$ nodes, we have $M = N+1$. However, because of the Dirichlet condition at the right boundary, only the first N basis functions $\phi_1, \ldots, \phi_N$ are used as test functions

to define the finite element equations. Following the notation of Lecture 12, we write

$$u(x) = \sum_{j=1}^{M} u_j \phi_j(x)$$

and obtain, for $i = 1, \ldots, N$,

$$\phi_i(0) \left(1 - \sum_{j=1}^{N+1} u_j \phi_j(0) \right) + \phi_i(1) \sum_{j=1}^{N+1} u_j \phi_j'(1) = \sum_{j=1}^{N+1} u_j [(\phi_i, \phi_j') + (\phi_i', \phi_j')],$$

where $\phi_i(x_j) = \delta_{ij}$. It follows that $\phi_i(1) = 0$ for $i = 1, \ldots, N$, so the second term on the left side vanishes. Furthermore, from the boundary conditions and the fact that $\phi_{N+1}(1) = 1$, we have $u_{N+1} = 1.5$, which yields the system of equations

$$\delta_{i,1} (1 - u_1) - 1.5[(\phi_i, \phi_{N+1}') + (\phi_i', \phi_{N+1}')] = \sum_{j=1}^{N} u_j [(\phi_i, \phi_j') + (\phi_i', \phi_j')].$$

(c) Assume the basis functions are the piecewise linear "hat" functions used in the notes for Lectures 11 and 12. Write down the elements of the stiffness matrix K and the load vector F, not accounting for the boundary conditions.

Solution Because there is no source term, we have

$$K_{ij} = (\phi_i, \phi_j') + (\phi_i', \phi_j'), \quad F_i = 0$$

so

$$K_{ii} = 2/h, \quad i > 1, \quad K_{11} = 1/h - 1/2,$$
$$K_{i+1,i} = -1/2 - 1/h, \quad K_{i,i+1} = 1/2 - 1/h.$$

(d) Incorporate the boundary conditions into the stiffness matrix and load vector.

Solution From the equations obtained in part 1b, we obtain

$$F_1 = 1, \quad F_N = -1.5 K_{N,N+1} = -1.5(1/2 - 1/h).$$

In addition, K_{11} must be increased by 1.

(e) For $N = 4$, 8 and 16, solve the system of equations from part 1d, using a tool such as MATLAB. Plot the approximate solution against the analytical solution (include all four solutions on the same plot). How good is the approximation? How does the accuracy improve as N increases?

Solution For all three values of N, the approximation is quite accurate, as can be seen in Figure 1. As N increases, the error

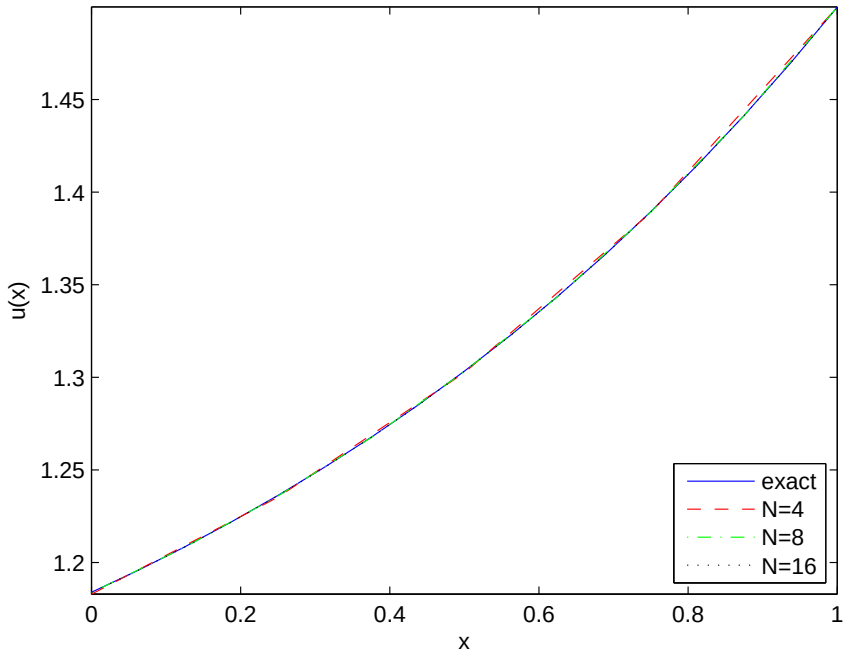

Figure 1: Plot for part 1e

decreases quadratically. The following MATLAB code computes the three approximate solutions.

```
[u1,err1]=femsolve(4);
[u2,err2]=femsolve(8);
[u3,err3]=femsolve(16);
L=1;
N=16;
h=L/N;
x=h*(0:N)';
```

```
exact=0.5*exp(x-1)+1;
figure(1)
clf
plot(x,exact)
hold on
plot(x(1:4:end),u1,'r--')
plot(x(1:2:end),u2,'g-.')
plot(x,u3,'k:')
xlabel('x')
ylabel('u(x)')
axis tight
legend('exact','N=4','N=8','N=16','Location','SouthEast')
```

The code for the function `femsolve` is

```
function [approx,err]=femsolve(N)
L=1;
h=L/N;
K=(2/h)*eye(N);
K=K+(1/2-1/h)*diag(ones(N-1,1),1);
K=K+(-1/2-1/h)*diag(ones(N-1,1),-1);
K(1,1)=1/h+1-1/2;
f=zeros(N,1);
f(1)=1;
f(N)=-(1/2-1/h)*1.5;
approx=[ K\f; 1.5 ];
x=h*(0:N)';
exact=0.5*exp(x-1)+1;
err=max(abs(exact-approx))./max(abs(exact));
```