CAAM 336 · DIFFERENTIAL EQUATIONS

Problem Set 7 · Solutions

Posted Wednesday 3 October 2012. Due Wednesday 10 October 2012, 5pm.

This problem set counts for 75 points, i.e., three-quarters the value of the earlier problem sets.
Because of the Centennial, late papers are due at 5pm on Monday, 15 October.

General advice: You may compute any integrals you encounter using symbolic mathematics tools such as WolframAlpha, Mathematica, or the Symbolic Math Toolbox in MATLAB.

1. [50 points: 20 points for (a); 10 points each for (b), (c), (d)]

   Use the finite element method to solve the differential equation

   $$-(u'(x)\kappa(x))' = 2x, \qquad 0 < x < 1$$

   for $\kappa(x) = 1 + x^2$, subject to homogeneous Dirichlet boundary conditions,

   $$u(0) = u(1) = 0,$$

   with the approximation space V_N given by the piecewise linear *hat functions* that featured on earlier problem sets: For $n \geq 1$, $h = 1/(N+1)$, and $x_k = kh$ for $k = 0, \ldots, N+1$, we have

   $$\phi_k(x) = \begin{cases} (x - x_{k-1})/h, & x \in [x_{k-1}, x_k); \\ (x_{k+1} - x)/h, & x \in [x_k, x_{k+1}); \\ 0, & \text{otherwise.} \end{cases}$$

   (a) Write MATLAB code that constructs the stiffness matrix $\mathbf{K}$ for a given value of N, with $\kappa(x) = 1 + x^2$.

   [You may edit the `fem_demo1.m` code from the class website. You should compute all necessary integrals (by hand or using a symbolic package) so as to obtain clean formulas that depend on h and the index of the hat functions involved (e.g.,$a(\phi_j, \phi_j)$ can depend on j).]

   (b) Write MATLAB code that constructs the load vector $\mathbf{f}$ for a given value of N, with $f(x) = 2x$.

   (c) For $N = 7$ and $N = 15$, produce plots comparing your solution u_N to the true solution

   $$u(x) = (4/\pi)\tan^{-1}(x) - x.$$

   (Note that you can compute $\tan^{-1}(x)$ as `atan(x)` in MATLAB.)

   (d) Produce a `loglog` plot showing how the error

   $$\max_{x \in [0,1]} |u_N(x) - u(x)|$$

   decreases as N increases. (For example, take $N = 8, 16, 32, 64, 128, 256, 512$.) On the same plot, show N^{-2} for the same values of N. If your code from parts (a) and (b) is working, your error curve should have the same slope as the N^{-2} curve. (Consult the `fem_demo1.m` code on the website for a demonstration of the style of plot we intend for part (d); edit this code as you like.)

Solution.

(a) First we compute the energy inner product of the basis functions. Note that

$$\frac{d\phi_k}{dx}(x) = \begin{cases} 1/h, & x \in [x_{k-1}, x_k); \\ -1/h, & x \in [x_k, x_{k+1}); \\ 0, & \text{otherwise.} \end{cases}$$

Thus we have

$$\begin{aligned} a(\phi_j, \phi_j) &= \int_0^1 (1+x^2)\Big(\frac{d\phi_j}{dx}(x)\Big)^2 \, dx \\ &= \int_{x_{j-1}}^{x_j} (1+x^2)\Big(\frac{1}{h}\Big)^2 \, dx + \int_{x_j}^{x_{j+1}} (1+x^2)\Big(-\frac{1}{h}\Big)^2 \, dx \\ &= \frac{1}{h^2}\int_{x_{j-1}}^{x_{j+1}} (1+x^2)\, dx \;=\; \frac{1}{h^2}\Big[x + \frac{x^3}{3}\Big]_{x_{j-1}}^{x_{j+1}} \;=\; \frac{2}{h} + \frac{2h}{3} + 2hj^2, \end{aligned}$$

$$\begin{aligned} a(\phi_j, \phi_{j+1}) &= \int_0^1 (1+x^2)\Big(\frac{d\phi_j}{dx}(x)\Big)\Big(\frac{d\phi_{j+1}}{dx}(x)\Big)\, dx \\ &= \int_{x_j}^{x_{j+1}} (1+x^2)\Big(-\frac{1}{h}\Big)\Big(\frac{1}{h}\Big)\, dx \\ &= -\frac{1}{h^2}\int_{x_j}^{x_{j+1}} (1+x^2)\, dx \;=\; -\frac{1}{h^2}\Big[x + \frac{x^3}{3}\Big]_{x_j}^{x_{j+1}} \;=\; -\frac{1}{h} - h\Big(j^2 + j + \frac{1}{3}\Big), \end{aligned}$$

and for $|j-k| > 1$,

$$a(\phi_j, \phi_k) = 0$$

since $\big(d\phi_j(x)/dx\big)\big(d\phi_k(x)/dx\big) = 0$ for all $x \in [0,1]$ (except at the nodes x_ℓ, where strictly speaking these derivatives are not defined—but these single isolated points do not add anything to the integral). The stiffness matrix is given by

$$\mathbf{K} = \begin{bmatrix} a(\phi_1, \phi_1) & \cdots & a(\phi_1, \phi_n) \\ \vdots & \ddots & \vdots \\ a(\phi_n, \phi_1) & \cdots & a(\phi_n, \phi_n) \end{bmatrix}.$$

(b) Next we compute the entries of the load vector:

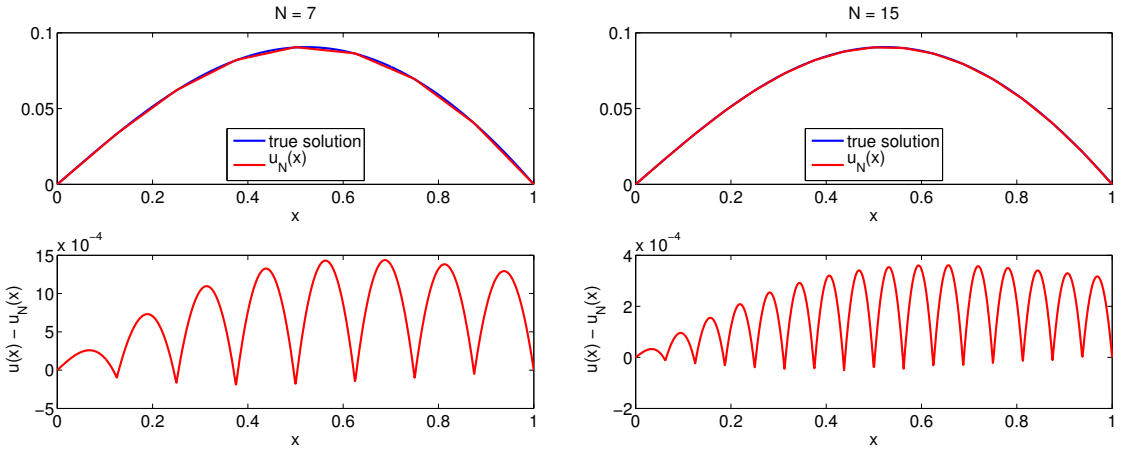
$$\begin{aligned} (f, \phi_j) &= \int_0^1 f(x)\phi_j(x)\, dx \\ &= \int_{x_{j-1}}^{x_j} (2x)\Big(\frac{x - x_{j-1}}{h}\Big)\, dx + \int_{x_j}^{x_{j+1}} (2x)\Big(\frac{x_{j+1} - x}{h}\Big)\, dx \\ &= \frac{1}{h}\Big[\frac{2x^3}{3} - x^2 x_{j-1}\Big]_{x_{j-1}}^{x_j} + \frac{1}{h}\Big[x^2 x_{j+1} - \frac{2x^3}{3}\Big]_{x_j}^{x_{j+1}} \\ &= 2h^2 j. \end{aligned}$$

The load vector is given by

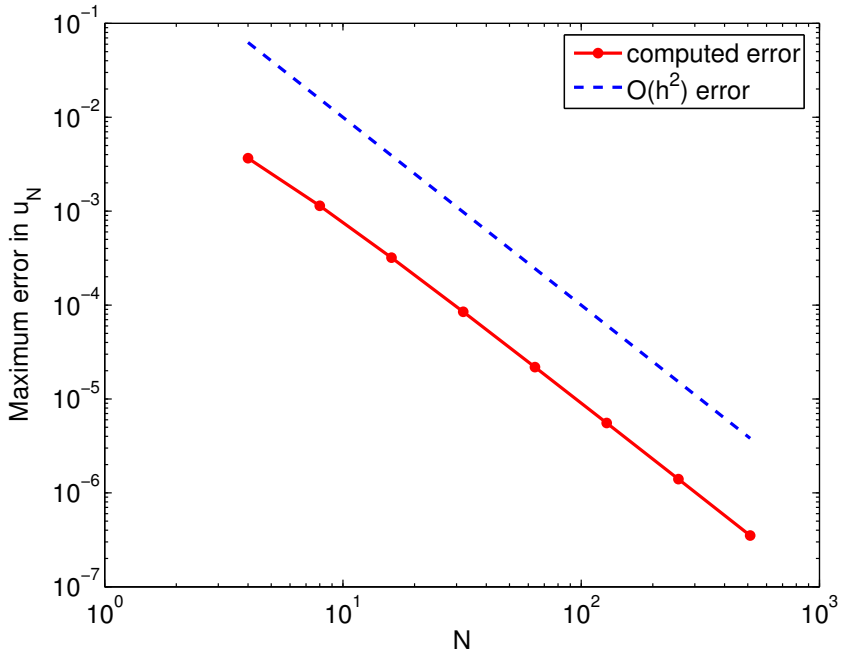
$$\mathbf{f} = \begin{bmatrix} (f, \phi_1) \\ \vdots \\ (f, \phi_n) \end{bmatrix}.$$

The MATLAB code at the end of this problem shows generates these matrices and produces plots similar to those shown in (b) and (c).

(c) The following plots show the solution (and error) at $N = 7$ (left) and $N = 15$ (right).



(d) The following plot shows the decay of the error as a function of N. Notice that the error decays like $1/N^2$.




```
% demo of the finite element method for the problem
%   -d/dx((1+x^2) du/dx)  = 2x, 0 < x < 1, u(0) = u(1) = 0.
% which has exact solution u(x) = (4/pi)*atan(x) - x.

Nvec = [4 8 16 32 64 128 256 512];   % vector of N values we shall use
maxerr = zeros(size(Nvec));          % vector to hold the max errors for each N

% each pass of the following loop handles a new N value...
for j=1:length(Nvec)
   N = Nvec(j);
   h = 1/(N+1);
   x = [1:N]*h;
```

```
% construct the stiffness matrix (integrals done by hand)
   maindiag = 2/h + 2*h/3 + 2*h*([1:N].^2);
   offdiag  = -1/h - h*(([1:N-1].^2) + [1:N-1] + 1/3);
   K = diag(maindiag) + diag(offdiag,1) + diag(offdiag,-1);

% construct the load vector (integrals done by hand)
   f = 2*h^2*[1:N]';

% solve for expansion coefficients of Galerkin approximation
   c = K\f;

% plot the true solution
   xx = linspace(0,1,1000)';     % finely spaced points between 0 and 1.
   u = (4/pi)*atan(xx)-xx;      % true solution

   figure(1), clf
   subplot(2,1,1)
   plot(xx, u, 'b-','linewidth',2)
   hold on

% plot the approximation solution
   uN = zeros(size(xx));
   for k=1:N
      uN = uN + c(k)*hat(xx,k,N);
   end
   plot(xx, uN, 'r-','linewidth',2)
   set(gca,'fontsize',16)
   xlabel('x')
   legend('true solution', 'u_N(x)','location','south')
   title(sprintf('N = %d', N))

% plot the error in the solution for this N
   subplot(2,1,2)
   plot(xx, u-uN, 'r-','linewidth',2)
   set(gca,'fontsize',16)
   xlabel('x')
   ylabel(' u(x) - u_N(x)')

% approximate the maximum error for this value of N
   maxerr(j) = max(abs(u - uN));

   input('hit return to continue')
end

% plot the maximum error
  figure(2), clf
  loglog(Nvec, maxerr, 'r.-', 'linewidth',2, 'markersize',20)
  hold on
  loglog(Nvec, Nvec.^(-2), 'b--', 'linewidth',2)
  legend('computed error', 'O(h^2) error')
  set(gca,'fontsize',16);
  xlabel('N')
  ylabel('Maximum error in u_N')
  print -depsc2 femb.eps
```

---

2. [25 points: 7 points for (a); 5 points for (b); 3 points for (c); 10 points for (d)]
A classical problem in quantum mechanics models a particle moving in an infinite square well, subject to an infinite potential at a point. The result is a Schrödinger operator posed on $C_D^2[0,1]$ of the form

$$Lu = -u'' + \delta_{1/2} u,$$

where $\delta_{1/2}$ is a "delta function" centered at the location of the infinite potential, $x = 1/2$. A beautiful theory supports these exotic functions (more properly called *distributions*). For this problem, you need

only know the following fact: for any function $g \in C[0,1]$,

$$\int_0^1 \delta_{1/2}(x) g(x)\, dx = g(1/2).$$

The equation $Lu = f$ has the equivalent weak form

$$a(u,w) = (f,w) \qquad \text{for all } w \in V = C_D^2[0,1],$$

where

$$a(u,w) = \int_0^1 \Big(u'(x)w'(x) + \delta_{1/2}(x)u(x)w(x) \Big)\, dx.$$

We wish to use the Galerkin method to approximate solutions to $Lu = f$ from the finite dimensional subspace $V_N = \text{span}\{\phi_1, \ldots, \phi_N\}$. Use as basis vectors the eigenfunctions from the problem without the potential at $x = 1/2$:

$$\phi_k(x) = \sqrt{2}\sin(k\pi x).$$

(a) Compute a general formula for $a(\phi_j, \phi_k)$.

(b) Write out (by hand) the stiffness matrix for $N = 5$.

(c) Write down a general formula for the entries in the load vector, (f, ϕ_k), when $f(x) = 1$. (You may use formulas from prior homework.)

(d) Plot your approximate solutions to $-u''(x) + \delta_{1/2}(x)u(x) = 1$ for $N = 5$ and $N = 35$.

Solution.

(a) Compute

$$\begin{aligned} a(\phi_j, \phi_k) &= \int_0^1 \big(\phi_j'(x)\phi_k'(x) + \delta_{1/2}(x)\phi_j(x)\phi_k(x) \big)\, dx \\ &= 2kj\pi^2 \int_0^1 \cos(j\pi x)\cos(k\phi x)\, dx + 2\int_0^1 \delta_{1/2}(x)\sin(j\pi x)\sin(k\pi x) \\ &= 2kj\pi^2 \int_0^1 \cos(j\pi x)\cos(k\phi x)\, dx + 2\sin(j\pi/2)\sin(k\pi/2). \end{aligned}$$

The integral in this last expression is $1/2$ when $j = k$, and zero otherwise. The second term will be zero if either j or k is even (since in that case one of the sine terms must be zero). If both j and k are odd, this term will be nonzero, ± 2. In general, we can write

$$a(\phi_j, \phi_k) = \begin{cases} j^2\pi^2 + 2\sin^2(j\pi/2), & \text{if } j = k; \\ 2\sin(j\pi/2)\sin(k\pi/2), & \text{otherwise.} \end{cases}$$

[**GRADERS**: the amount that students simplify $a(\phi_j, \phi_k)$ will vary. The ultimate solution need not take the precise form that we have given above, but it should be simplified beyond just writing down the definition of $a(\phi_j, \phi_k)$.]

(b) For $N = 5$ we have

$$\begin{bmatrix} \pi^2+2 & 0 & -2 & 0 & 2 \\ 0 & 4\pi^2 & 0 & 0 & 0 \\ -2 & 0 & 9\pi^2+2 & 0 & -2 \\ 0 & 0 & 0 & 16\pi^2 & 0 \\ 2 & 0 & -2 & 0 & 25\pi^2+2 \end{bmatrix}$$

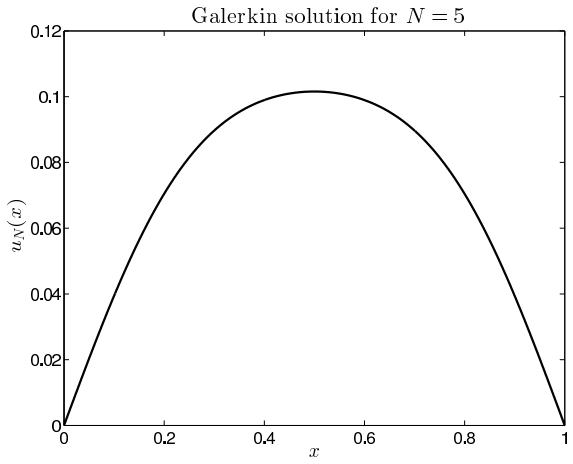
(c) The entries of the load vector are

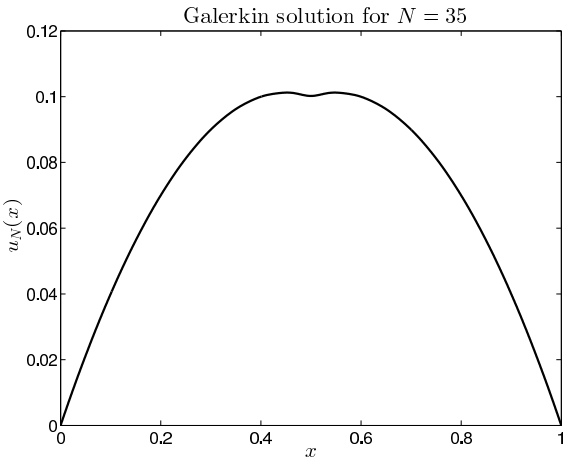
$$(f, \phi_k) = \int_0^1 1 \cdot \sqrt{2}\sin(k\pi x)\, dx = \begin{cases} 2\sqrt{2}/(n\pi), & \text{if } k \text{ is odd;} \\ 0, & \text{if } k \text{ is even,} \end{cases}$$

as computed in previous examples earlier in the semester.

[**GRADERS**: students do not need to show work for this formula.]

(d) Approximate solutions for $N = 5$ and $N = 35$ are shown below, followed by the code that produced them.






```
for N = [5 35]

 K = zeros(N); f = zeros(N,1);

 for j=1:N, for k=1:N
    K(j,k) = 2*sin(j*pi/2)*sin(k*pi/2);
 end, end

 K = K + diag([1:N].^2*pi^2);

 for k=1:N
    f(k) = (sqrt(2)/pi)*(1+(-1).^(k+1))./k;
 end

 c = K\f;

 xx = linspace(0,1,1000);
 uN = zeros(size(xx));

 for k=1:N
    uN = uN + c(k)*sqrt(2)*sin(k*pi*xx);
 end

 figure(N), clf
```

```
 plot(xx,uN,'k-','linewidth',2)
 title(sprintf('Galerkin solution for $N=%d$',N),'interpreter','latex','fontsize',18)
 xlabel('$x$','interpreter','latex','fontsize',16)
 ylabel('$u_N(x)$','interpreter','latex','fontsize',16)
 set(gca,'fontsize',14)
 eval(sprintf('print -depsc2 delta_%d', N))

 if N==5, disp(K), end
end
```