MATH 517 Finite Differences Homework 2

Caleb Logemann

February 18, 2016

1. Consider the 2-pt boundary value problem:

$$\begin{aligned} -u'' &= f(x) \text{ on } 0 < x < L \\ u(0) = \alpha, &\quad u'(L) = \sigma. \end{aligned}$$

Discretize this problem using the $O(h^2)$ central finite differences and a ghost point near $x = L$ to handle the Neumann boundary condition. Write out the resulting linear system.

To discretize this problem let $x_i = ih$ where $h = \frac{L}{N+1}$ and N is the number of points in the discretization. Thus the solution to this BVP will be approximated on these grid points. Let U_i be the approximate value of $u(x_i)$. Thus an approximate solution to this BVP will be the values of U_i for $0 \leq i \leq N+1$. From the boundary conditions, it can be noted that $U_0 = \alpha$. The other boundary condition can be handled by introducing a ghost point U_{N+2} with the following two conditions

$$\frac{1}{h^2}(-U_N + 2U_{N+1} - U_{N+2}) = f(x_{N+1}) = f(L)\frac{1}{2h}(U_{N+2} - U_N) \qquad = \sigma.$$

In other words the central finite difference for the second derivative must hold on U_N, 2_{N+1}, and U_{N+2}, and the central difference for the first derivative must equal σ when centered on U_{N+1}. These two equations can be combined so that the ghost point U_{N+2} no longer appears in our finite difference method.

$$\frac{1}{h^2}(-2U_N + 2U_{N+1}) = f(x_{N+1}) + \frac{2\sigma}{h}$$

Now instead of a system of N equations for $i = 1, 2, \ldots, N$, we have a system of $N+1$ equations for $i = 1, 2, \ldots, N+1$. For $i = 2, \ldots, N$, the central finite difference can be applied directly to the PDE to result in the following $N-1$ equations.

$$\frac{1}{h^2}(-U_{i-1} + 2U_i - U_{i+1}) = f(x_i)$$

For $i = 1$, we simply replace U_0 with α and simplify to

$$\frac{1}{h^2}(2U_1 - U_2) = f(x_1) + \frac{\alpha}{h^2}$$

For $i = N+1$, we use the equation we found before after removing the ghost point.

$$\frac{1}{h^2}(-2U_N + 2U_{N+1}) = f(x_{N+1}) + \frac{2\sigma}{h}$$

Then this system of equations cen be written as a matrix equation as follows

$$A\mathbf{U} = \mathbf{f}$$

where

$$\begin{aligned} \mathbf{U} &= [U_1, U_2, \cdots, U_{N+1}]^T \\ \mathbf{f} &= \left[f(x_1) + \frac{\alpha}{h^2}, f(x_2), \cdots, f(x_N), f(x_{N+1}) + \frac{2\sigma}{h}\right]^T \\ A &= \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{bmatrix} \end{aligned}$$

2. Find the Green's function that satisfies:

$$-G'' = \delta(x-\xi), \quad G(0;\xi) = 0, \quad G'(L;\xi) = 0.$$

First we can see that integrating both sides over some interval $[a,b]$ results in

$$\begin{aligned}\int_a^b -G'' \,\mathrm{d}x &= \int_a^b \delta(x-\xi)\,\mathrm{d}x \\ G'(a;\xi) - G'(b'\xi) &= \int_a^b \delta(x-\xi)\,\mathrm{d}x\end{aligned}$$

If either $a, b < \xi$ or $a, b > \xi$, then $G'(a;\xi) - G'(b;\xi) = 0$. Since this applies for any interval $[a,b]$ we can conclude that $G'(a;\xi) = G'(b;\xi)$, and thus G is piecewise linear before and after ξ. If $a < \xi < b$, then $G'(a;\xi) - G'(b;\xi) = 1$. In particular $G'(L;\xi) = 0$, and since G is linear $G'(b;\xi) = 0$ for and $b > \xi$. Thus we can conclude that $G'(a;\xi) = 1$ for any $a < \xi$. Also since $G(0;\xi) = 0$, we can conclude that $G(x;\xi) = x$ for any $x < \xi$. Since G is continuous, $G(\xi;\xi) = \xi$. Now since $G'(x;\xi) = 0$ for any $x > \xi$, we can conclude that $G(x;\xi) = \xi$ for any $x > \xi$. Now G can be described in a piecewise fashion as follows

$$G(x;\xi) = \begin{cases} x & x \le \xi \\ \xi & x > \xi \end{cases}$$

3. Use the result from Problem 2 to write out the exact solution to the boundary value problem with general $f(x)$, α, and σ.

First we must find the homogenous solution, that is we must find u_1 that satisfies $-u_1'' = 0$ and $u_1(0) = \alpha$ and $u_1'(L) = \sigma$. The general solution to $-u_1'' = 0$ is $u_1(x) = mx + b$. Using the boundary conditions we find that $u_1(x) = \sigma x + \alpha$. Next we must find the nonhomogenous solution, that is we must find u_2 that satisfies $-u_2'' = f(x)$ and $u_2(0) = 0$ and $u_2'(L) = 0$.

$$\begin{aligned}-u_2'' &= f(x) \\ \int_0^L -u_2'' G(x;\xi)\,\mathrm{d}x &= \int_0^L f(x)G(x;\xi)\,\mathrm{d}x \\ u_2' &= f(\xi) \\ \int_0^L u_2' G(x;\xi)\,\mathrm{d}\xi &= \int_0^L f(\xi)G(x;\xi)\,\mathrm{d}\xi \\ u_2 &= \int_0^L f(\xi)G(x;\xi)\,\mathrm{d}\xi\end{aligned}$$

We find that $u_2(x) = \int_0^L f(\xi)G(x;\xi)\,\mathrm{d}\xi$. Thus the exact solution to the entire problem is

$$\begin{aligned}u(x) &= u_1(x) + u_2(x) \\ &= \sigma x + \alpha + \int_0^L f(\xi)G(x;\xi)\,\mathrm{d}\xi.\end{aligned}$$

4. Use the results from Problems 2 and 3 to find the exact inverse to the finite difference matrix found in Problem 1.

The exact inverse of the matrix A can be found by sampling the Green's function on the mesh, that is $A_{ij}^{-1} = hG(x_i;x_j)$. In a piecewise notation

$$A_{ij}^{-1} = \begin{cases} h \times ih & i \le j \\ h \times jh & i > j \end{cases}$$

5. Use the result in Problem 4 to prove that the finite difference method in Problem 1 is L_∞-stable.

   In order to prove that the finite difference method is L_∞-stable, it must be shown that the infinity norm of A^{-1} is bounded, that is that there exists some constant C such that $\left\|A^{-1}\right\|_\infty \leq C$. First note that the infinity norm of a matrix is the maximum row sum of that matrix, therefore

$$\left\|A^{-1}\right\|_\infty = \max_{1\leq i\leq N} \sum_{j=1}^{N} \left(\left|A_{ij}^{-1}\right|\right)$$

   From problem 4 we can see that $\left|A_{ij}^{-1}\right| \leq hL$, since $ih < L$ and $jh < L$. Therfore

$$\left\|A^{-1}\right\|_\infty \leq \max_{1\leq i\leq N} \sum_{j=1}^{N} (hL)$$

   Because this no longer depends on i or j

$$\begin{aligned}\left\|A^{-1}\right\|_\infty &\leq NhL \\ &= N\frac{L}{N+1}L \\ &\leq L^2\end{aligned}$$

   Thus $\left\|A^{-1}\right\|_\infty \leq L^2$, and since L^2 is constant the infinity norm is bounded and finite difference method is L_∞-stable. The stability of this problem can then be coupled with the consistency to show that this method is globally convergent.

6. Consider the uniform mesh $x_j = jh$ and let

$$U_j = u(x_j) \quad \text{and} \quad W_j \approx u'(x_j).$$

   In standard finite differences, we typically find linear combinations of U_j to define the approximation W_i to $u'(x_j)$:

$$W_i = \sum_j \beta_j U_j$$

   In compact finite differences we are allowed to generalize this to

$$\sum_j \alpha_j W_j = \sum_j \beta_j U_j$$

   Find the compact finite difference with the optimal local truncation error that has the following form:

$$\alpha W_{j-1} + W_j + \alpha W_{j+1} = \beta\left(\frac{U_{j+1} - U_{j-1}}{2h}\right).$$

   We can find the local truncation error by inserting the exact solution into the finite difference equation and using Taylor series.

$$\tau_j = -\alpha u'(x_{j-1}) - u'(x_j) - \alpha u'(x_{j+1}) + \beta\left(\frac{u(x_{j+1}) - u(x_{j-1})}{2h}\right)$$

$$u(x_{j-1}) = u(x_j) - hu'(x_j) + \frac{h^2}{2}u''(x_j) - \frac{h^3}{6}u'''(x_j) + \frac{h^4}{24}u^{(4)}(x_j) - \frac{h^5}{120}u^{(5)}(x_j) + O(h^6)$$

$$u(x_{j+1}) = u(x_j) + hu'(x_j) + \frac{h^2}{2}u''(x_j) + \frac{h^3}{6}u'''(x_j) + \frac{h^4}{24}u^{(4)}(x_j) + \frac{h^5}{120}u^{(5)}(x_j) + O(h^6)$$

$$u'(x_{j-1}) = u'(x_j) - hu''(x_j) + \frac{h^2}{2}u'''(x_j) - \frac{h^3}{6}u^{(4)}(x_j) + \frac{h^4}{24}u^{(5)}(x_j) - \frac{h^5}{120}u^{(6)}(x_j) + O(h^6)$$

$$u'(x_{j+1}) = u'(x_j) + hu''(x_j) + \frac{h^2}{2}u'''(x_j) + \frac{h^3}{6}u^{(4)}(x_j) + \frac{h^4}{24}u^{(5)}(x_j) + \frac{h^5}{120}u^{(6)}(x_j) + O(h^6)$$

Substituting in these Taylor series and simplifying results in

$$\tau_j = (-1 - 2\alpha + \beta)u'(x_j) + \frac{h^2}{6}(-6\alpha + \beta)u^{(3)}(x_j) + \frac{h^4}{120}(-10\alpha + \beta)u^{(5)}(x_j) + O(h^6)$$

To make the local truncation error as small as possible, we must choose α and β such that the following two equations are satisfied.

$$\begin{aligned} 0 &= -1 - 2\alpha + \beta \\ 0 &= -6\alpha + \beta \end{aligned}$$

If both of these equations are satisfied then this finite difference will be a fourth order approximation. These two equations can be solved as follows.

$$\begin{aligned} \beta &= 6\alpha \\ 0 &= -1 - 2\alpha + 6\alpha \\ 1 &= 4\alpha \\ \alpha &= \frac{1}{4} \\ \beta &= \frac{3}{2} \end{aligned}$$

If α and β are set to these values, then the local truncation error becomes

$$\tau_j = \frac{h^4}{120}(-10\alpha + \beta)u^{(5)}(x_j) + O(h^6)$$

.

Thus the compact finite difference operator will be

$$\frac{1}{4}W_{j-1} + W_j + \frac{1}{4}W_{j+1} = 3\left(\frac{U_{j+1} - U_{j-1}}{4h}\right).$$

which is 4th order accurate, because the local truncation error is $\tau = O(h^4)$.

7. Consider Poisson's equation in 2D:

$$\begin{aligned} -u_{xx} - u_{yy} = f(x, y) \text{ in } \Omega = [0, 1] \times [0, 1], \\ u = g(x, y) \text{ on } \partial\Omega \end{aligned}$$

Discretize this equation using the 5-point Laplacian on a uniform mesh $\Delta x = \Delta y = h$. Use the standard natural row-wise ordering.

First we need to specify the discretization of the space Ω. Let $x_i = ih$ for $i = 0, 1, \ldots, N + 1$ and let $y_j = jh$ for $j = 0, 1, \ldots, N + 1$, where $h = \frac{1}{N+1}$. Then the solution to this PDE can be described by approximating u on this mesh, that is $U_{i,j} \approx u(x_i, y_j)$ is an approximation to the exact solution.

Now we can apply finite differences to this PDE. The 5-point Laplacian on a uniform mesh is given by

$$\Delta u \approx \frac{1}{h^2}(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}).$$

Using this finite difference in the PDE results in the following set of equations

$$\frac{1}{h^2}(4U_{i,j} - U_{i-1,j} - U_{i+1,j} - U_{i,j-1} - U_{i,j+1}) = f_{ij}$$

where $f_{ij} = f(x_i, y_j)$.

Now in order to turn this into a linear system, a new numbering scheme needs to be imposed. I will use the natural row-wise ordering where each point is numbered along the rows starting at $U_{1,1} = U_1$. In general $U_k = U_{i,j}$ when $k = i + (j-1)N$.

Using this numbering scheme, the finite difference method turn into the following linear system.

$$A\mathbf{U} = \mathbf{f}$$

$A \in \mathbb{R}^{N^2 \times N^2}$ is the block matrix

$$A = \begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{bmatrix}$$

$$T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}$$

and I is the $N \times N$ identity matrix

$$\mathbf{f} = \begin{bmatrix} h^2 f(x_1, y_1) + g(x_1, y_0) + g(x_0, y_1) \\ h^2 f(x_2, y_1) + g(x_2, y_0) \\ \vdots \\ h^2 f(x_{N-1}, y_N) + g(x_{N-1}, y_{N+1}) \\ h^2 f(x_N, y_N) + g(x_{N+1}, y_N) + g(x_N, y_{N+1}) \end{bmatrix}$$

For the vector $\mathbf{f}$ the boundary conditions are present for any index k that appears on the boundary of the mesh, otherwise the entry of $\mathbf{f}$ is simply $h^2 f(x_i, y_j)$.

8. Write a MATLAB code that constructs the sparse coefficient matrix A and the appropriate right hand side vector $\mathbf{F}$.

   The following function solves the 2D Poisson equations on $[0, L] \times [0, L]$, with source function f, Dirichlet boundary conditions g, and N points in the x and y discretization.

```
function [U, Ux, Uy] = Poisson2D(f, g, L, N)
    p = inputParser;
    p.addRequired('f', @Utils.isFunctionHandle);
    p.addRequired('g', @Utils.isFunctionHandle);
    p.addRequired('L', @(x) isnumeric(x) && x > 0);
    p.addRequired('N', @Utils.isInteger);
    p.parse(f, g, L, N);

    h = L/(N+1);
    x = 0:h:L;
    y = 0:h:L;

    % create functions to swap between row-wise ordering and i,j ordering
    %kFun = @(i, j) i + (j-1)*N;
    iFun = @(k) mod(k-1,N)+1;
```

```
    jFun = @(k) floor((k-1)/N) + 1;

    k = 1:N^2;
    % create vectors for x and y positions for each entry in U
    Ux = x(iFun(k) + 1);
    Uy = y(jFun(k) + 1);

    % vector of forcing function values at each index k
    F = h^2*f(Ux, Uy);
    %arrayfun(@(k) h^2*f(x(iFun(k) + 1), y(jFun(k) + 1)), k);
    % add on boundary conditions to F
    % add bottom boundary
    kBottom = 1:N;
    F(kBottom) = F(kBottom) + g(x(2:N+1),0);
    % add top boundary
    kTop = (N^2-N+1):N^2;
    F(kTop) = F(kTop) + g(x(2:N+1), L);
    % add left boundary
    kLeft = 1:N:(N^2-N)+1;
    F(kLeft) = F(kLeft) + g(0, y(2:N+1));
    % add right boundary
    kRight = N:N:N^2;
    F(kRight) = F(kRight) + g(L, y(2:N+1));

    % build sparse matrix A
    % build main diagonal
    iMain = k';
    jMain = k';
    sMain = 4*ones(N^2,1);
    % build upper main diagonal
    iUpper = k';
    iUpper(N:N:N^2) = [];
    jUpper = k';
    jUpper(1:N:N^2) = [];
    sUpper = -ones(size(iUpper));
    % build lower main diagonal
    iLower = jUpper;
    jLower = iUpper;
    sLower = sUpper;
    % build off upper diagonal
    iU = (1:(N^2-N))';
    jU = ((N+1):N^2)';
    sU = -ones(size(iU));
    % build off lower diagonal
    iL = jU;
    jL = iU;
    sL = sU;

    iA = [iMain; iUpper; iLower; iU; iL];
    jA = [jMain; jUpper; jLower; jU; jL];
    sA = [sMain; sUpper; sLower; sU; sL];
    A = sparse(iA, jA, sA);

    U = A\F';

    % change U from 1D vector to 2D matrix
    %U = vec2mat(U, N);
end
```

9. Using your code, do a numerical convergence study for the following right-hand side forcing and

exact solution:

$$f(x,y) = -1.25e^{x+.5y} \quad \text{and} \quad u(x,y) = e^{x+.5y}$$

The following code evaluates the function from the previous problem for the given BVP.

```
f = @(x,y) -1.25*exp(x + .5*y);
u = @(x,y) exp(x + .5*y);
Error = [];
for N = [10*2.^(0:5)]
    [U, Ux, Uy] = Poisson2D(f, u, 1, N);
    uExact = u(Ux, Uy);
%     USquare = vec2mat(U,N);
%     uExactSquare = vec2mat(uExact,N);
%     figure;
%     surf(USquare);
%     hold on
%     surf(uExactSquare);
%     pause
    Error = [Error; Ux(2) - Ux(1), norm(U - uExact', 'inf')];

end
hRatios = Error(1:end-1,1)./Error(2:end,1);
errorRatios = Error(1:end-1,2)./Error(2:end,2);
order = log(errorRatios)./log(hRatios);
table(hRatios, errorRatios, order)
```

```
>> H02_9

ans =

    hRatios    errorRatios    order
    -------    -----------    ------

    1.9091     3.6179         1.9886
    1.9524     3.8055         1.9975
    1.9756     3.9002         1.9989
    1.9877     3.9498         1.9996
    1.9938      3.975         1.9999
```