MATH 517 Finite Differences Homework 3

Caleb Logemann

March 1, 2016

1. Consider Poisson's equation in 2D:

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

Discretize this equation using the 9-point Laplacian on a uniform mesh $\Delta x = \Delta y = h$. Use the standard row-wise ordering.

2. Write a MATLAB code that constructs the sparse coefficient matrix A and the appropriate right-hand side vector $\mathbf{F}$. NOTE: you will need to modify the right-hand side vector to include the appropriate Laplacian of the right-hand side function.

The following code is a function that solve the 2D Poisson problem using the 9 Point Laplacian.

```
function [U, Ux, Uy] = Poisson2D_9PointLaplacian(f, g, L, N, fLaplacian)
    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.addRequired('fLaplacian', @Utils.isFunctionHandle);
    p.parse(f, g, L, N, fLaplacian);

    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 = 6*h^2*(f(Ux, Uy)+h^2/12*fLaplacian(Ux, Uy));
    % add on boundary conditions to F
    % add bottom boundary
    kBottom = 1:N;
    F(kBottom) = F(kBottom) + 4*g(x(2:N+1),0) + g(x(1:N),0) + g(x(3:N+2),0);
    % add top boundary
    kTop = (N^2-N+1):N^2;
    F(kTop) = F(kTop) + 4*g(x(2:N+1), L) + g(x(1:N), L) + g(x(3:N+2),L);
    % add left boundary
    kLeft = 1:N:(N^2-N)+1;
    F(kLeft) = F(kLeft) + 4*g(0, y(2:N+1)) + g(0, y(1:N)) + g(0, y(3:N+2));
    % add right boundary
    kRight = N:N:N^2;
    F(kRight) = F(kRight) + 4*g(L, y(2:N+1)) + g(L, y(1:N)) + g(L, y(3:N+2));
    % each corner got a double boundary condition
    F(1) = F(1) - g(0, 0);
    F(N) = F(N) - g(L, 0);
    F(N^2 - N + 1) = F(N^2 - N + 1) - g(0, L);
    F(N^2) = F(N^2) - g(L, L);

    % build sparse matrix A
    % A is block tridiagonal with symmetric upper and lower diagonals
    e = ones(N, 1);
```

```
    % block on main diagonal
    T = spdiags([-4*e, 20*e, -4*e], [-1, 0, 1], N, N);
    % block for upper and lower diagonals
    S = spdiags([-1*e, -4*e, -1*e], [-1, 0, 1], N, N);
    % shape of main diagonals and off diagonals
    I = eye(N);
    O = spdiags([e, e], [-1, 1], N, N);
    A = kron(I, T) + kron(O, S);

    U = A\F';

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

3. Using your code do a numerical convergence study for the following right-hand side forcing and exact solution:

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

Just use the built-in backslash operator in MATLAB to solve the linear system.

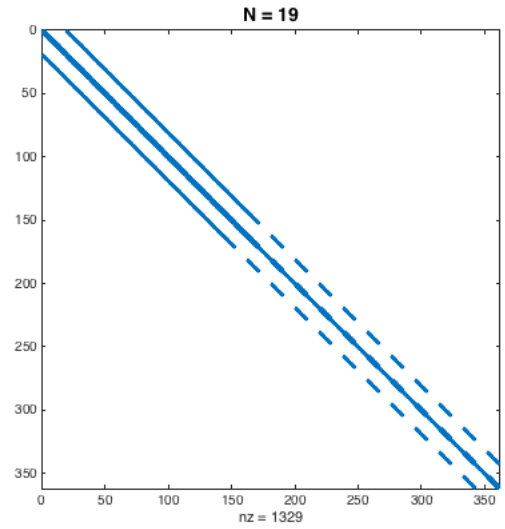
The following script uses the previous function to test the convergence.

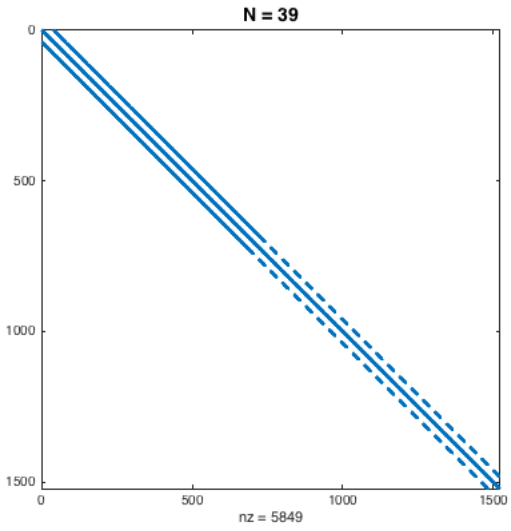
```
f = @(x,y) -1.25*exp(x + .5*y);
u = @(x,y) exp(x + .5*y);
fLaplacian = @(x, y) -1.5625*exp(x + .5*y);
Error = [];
for N = [3, 10*2.^(0:5)]
    [U, Ux, Uy] = Poisson2D_9PointLaplacian(f, u, 1, N, fLaplacian);
    uExact = u(Ux, Uy);
%     USquare = vec2mat(U,N);
%     uExactSquare = vec2mat(uExact,N);
%     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)
```

The scripts output is as follows.

4.

5. For $N = 19$ and $N = 39$ produce a spy plot of the matrix.



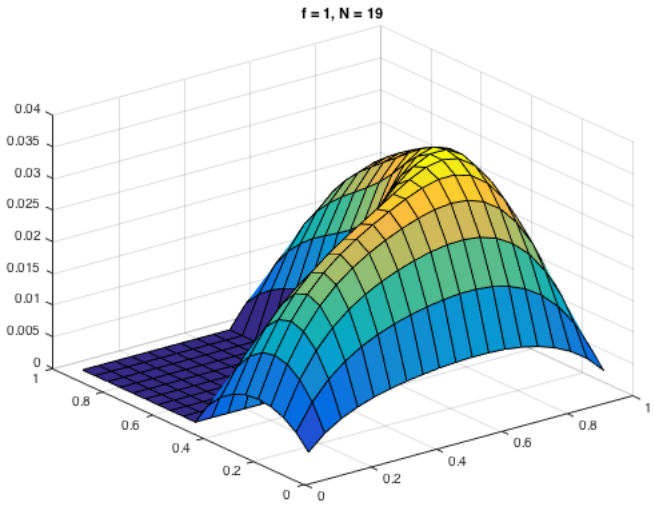


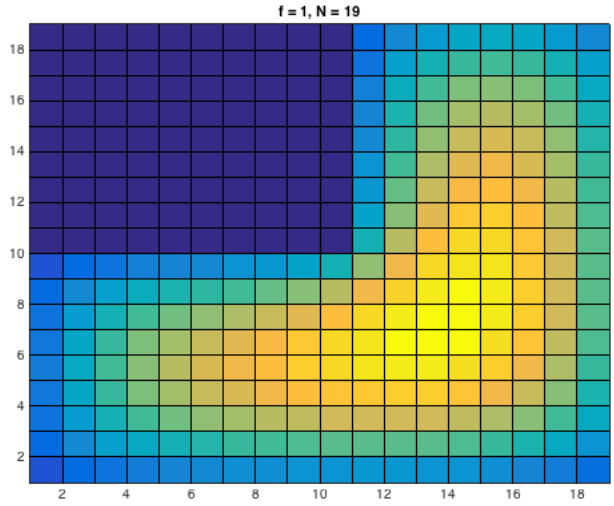
6. Solve the PDE using your code with the right hand side

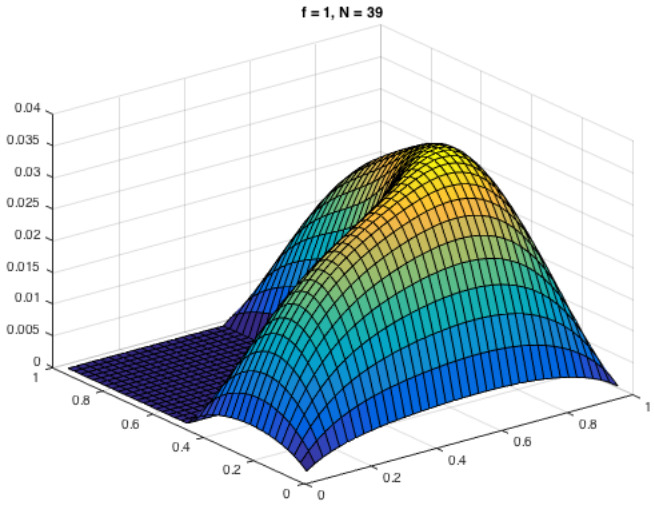
$$f(x, y) = 1.$$

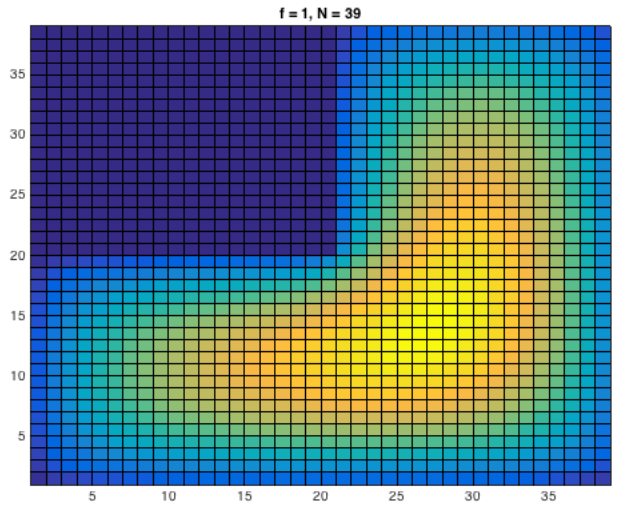
The following code evalates the function from problem 4 for the given right hand side. The images produced are shown below.

```
% Problem 6
f = @(x,y) ones(size(x));
for N = [19, 39]
    [U, Ux, Uy, A] = Poisson2D_5PointLaplacian_IrregularGeometry(f, 1, N);
    USquare = vec2mat(U,N);
    figure;
    surf(vec2mat(Ux, N), vec2mat(Uy, N), USquare);
    title(['f = 1, N = ',num2str(N)])
    figure
    pcolor(USquare);
    title(['f = 1, N = ',num2str(N)])
end
```


f = 1, N = 19
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


f = 1, N = 19
18
16
14
12
10
8
6
4
2
2
4
6
8
10
12
14
16
18


f = 1, N = 39
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

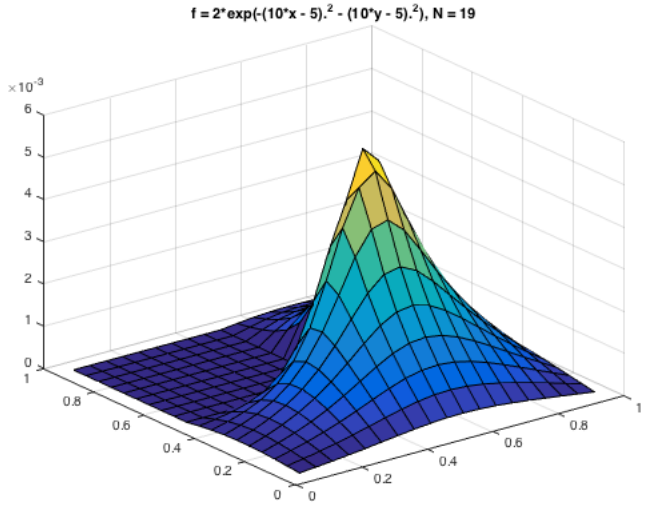


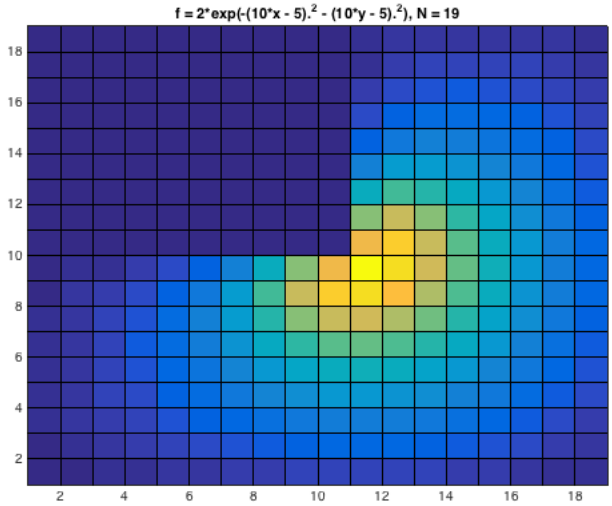
7. Solve the PDE using your code with the right hand side

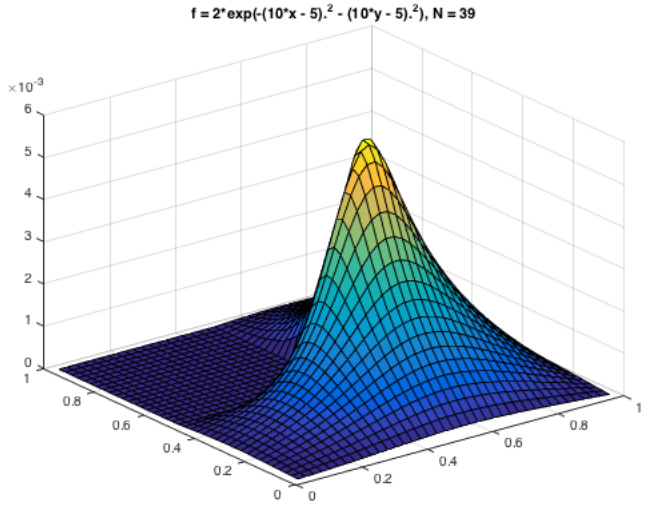
$$f(x,y) = 2exp\Big[-(10x-5)^2 - (10y-5)^2\Big].$$

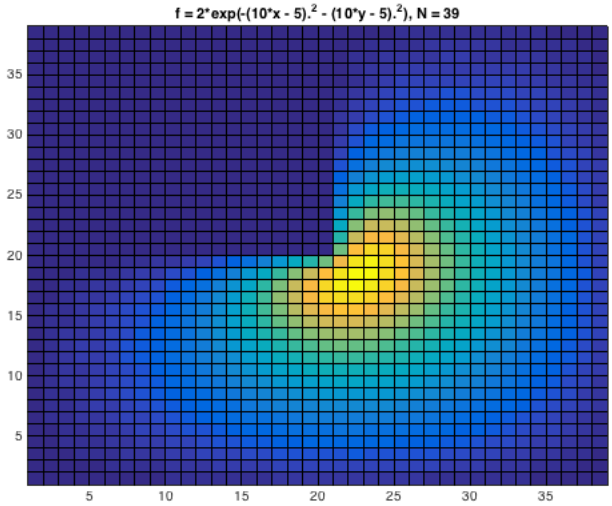
The following code evalates the function from problem 4 for the given right hand side. The images produced are shown below.

```
% Problem 7
f = @(x,y) 2*exp(-(10*x - 5).^2 - (10*y - 5).^2);
for N = [19, 39]
    [U, Ux, Uy, A] = Poisson2D_5PointLaplacian_IrregularGeometry(f, 1, N);
    USquare = vec2mat(U,N);
    figure;
    surf(vec2mat(Ux, N), vec2mat(Uy, N), USquare);
    title(['f = 2*exp(-(10*x - 5).^2 - (10*y - 5).^2), N = ',num2str(N)])
    figure
    pcolor(USquare);
    title(['f = 2*exp(-(10*x - 5).^2 - (10*y - 5).^2), N = ',num2str(N)])
end
```









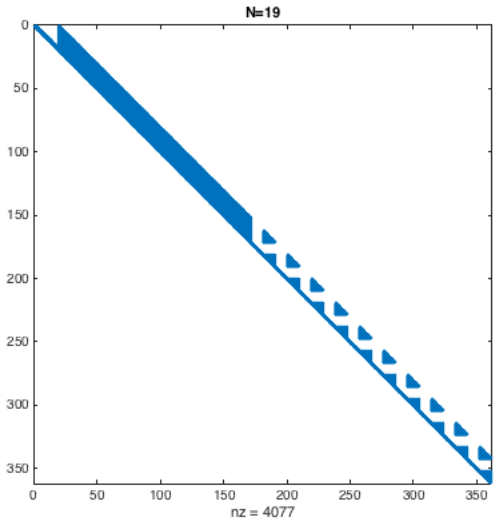
8. Compute the Cholesky factorization of A is MATLAB: R = chol(A); where R is an upper triangular matrix such that $A = R^T R$. For $N = 19$ and $N = 39$ produce a spy plot of R. Create a table

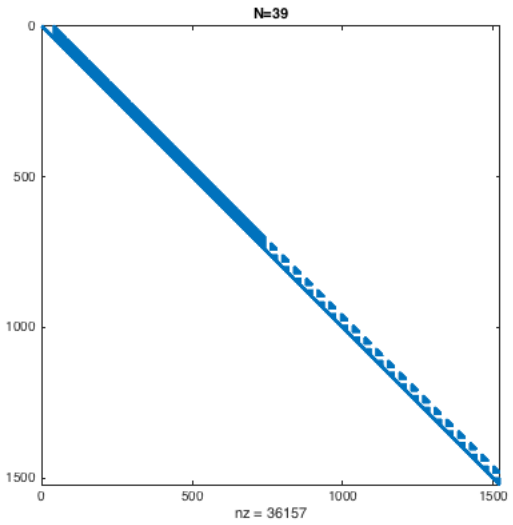
showing the non-zeros in R for $N = 9, 19, 39, 79, 159, 319$.

The following code is for problems 8 and 9.

```
% Problem 8 and 9
countsA = [];
countsB = [];
N = 10*2.^(0:5) - 1;
for iN = N
    [U, Ux, Uy, A] = Poisson2D_5PointLaplacian_IrregularGeometry(f, 1, iN);
    R = chol(A);
    % spy plots
    if(iN == 19 || iN == 39)
        figure
        spy(R);
        title(strcat('N= ',num2str(iN)));
    end
    countsA = [countsA; nnz(R)];
    P = symrcm(A);
    B = A(P, P);
    R = chol(B);
    if(iN == 19 || iN == 39)
        figure
        spy(R);
        title(strcat('Reverse Cuthill-Mckee, N= ',num2str(iN)));
    end
    countsB = [countsB; nnz(R)];
end
table(N', countsA)
table(N', countsB)
```

The following is the spy plots and table for problem 8.

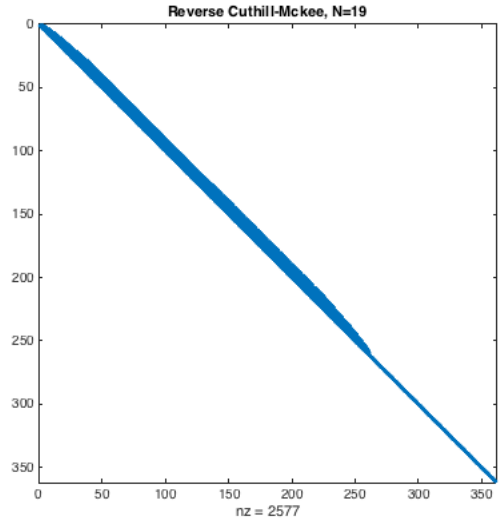


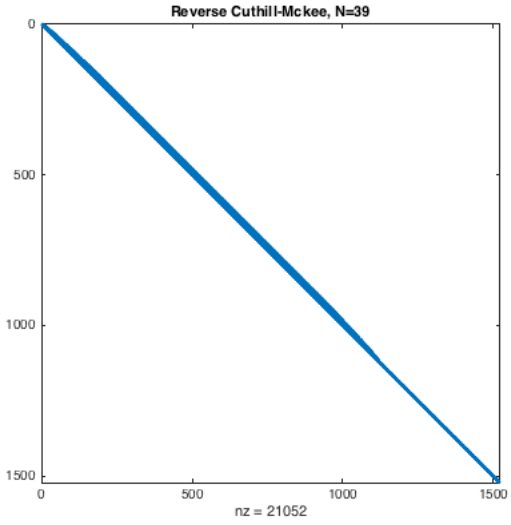



```
ans =

    Var1      countsA
    ----    ----------

    9             412
    19           4077
    39          36157
    79      3.0432e+05
    159      2.4966e+06
    319      2.0225e+07
```

9. Permute the matrix A using the reverse Cuthill-Mckee algorithm in MATLAB: P = symrcm(A); B = A(P, P); Compute the Cholesky factorization of B in MATLAB: R = chol(B); For $N = 19$ and $N = 39$ produce a spy plot of R. Create a table showing the non-zeros in R for $N = 9, 19, 39, 79, 159, 319$.

   The following is the spy plots and table for problem 9 generated by the code shown under problem 8.






```
ans =

    Var1     countsB
    ----    ----------

    9             302
    19           2577
    39          21052
    79       1.697e+05
    159      1.3618e+06
    319      1.0909e+07
```