Caleb Logemann
AER E 546 Fluid Mechanics and Heat Transfer I
Homework 5

1. Solve the Laplace equation

$$\partial_x^2 \psi + \partial_y^2 \psi = 0$$

in the domain $0 \le x, y \le 1$, with

$$\psi = \sin^2(3n\pi y) \text{ on } x = 0$$
$$\psi = \sin^2(3n\pi x) \text{ on } y = 0$$
$$\psi = 0 \text{ on } x = 1$$
$$\psi = 0 \text{ on } y = 1.$$

Obtain numerical solutions with $n = 1$ and $n = 3$.

Use Gauss-Seidel with SOR. Stop either when the modulus of the residual either has dropped by a factor of 10^{-4} or after 2500 iterations.

- Provide the algorithm part of your code.
- Provide contour plots of streamfunctions for each of 15×15 and 151×151 grids for both n values.
- For the 151×151 grid and $n = 1$, on a single graph, plot residual versus iteration for each of the relaxation parameters $\lambda = 1$, $\lambda = 0.5$, $\lambda = 1.5$, and $\lambda = 1.95$. Plot as $\log_{10}(residual/residual_0)$ versus iteration number. Use the L_2 norm

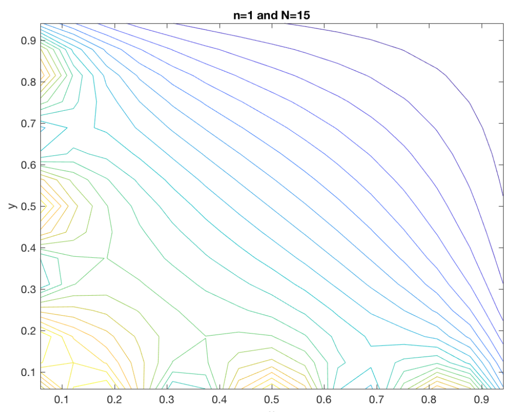
$$residual = \|\Delta\Psi\|_{L_2} = \sqrt{\sum_{i=1}^{I}\left(\sum_{j=1}^{J}\left((\Delta\Psi_{ij}^2)/(I \times J)\right)\right)}$$

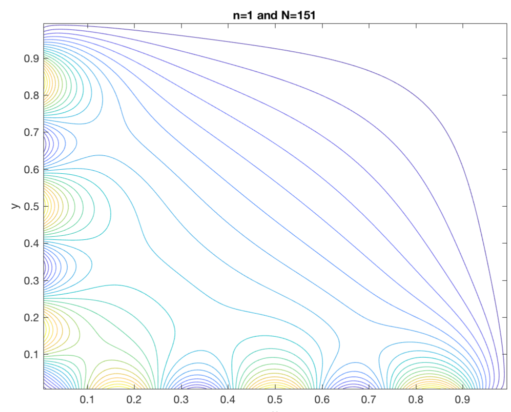
The following is my algorithm for Gauss-Seidel with SOR.

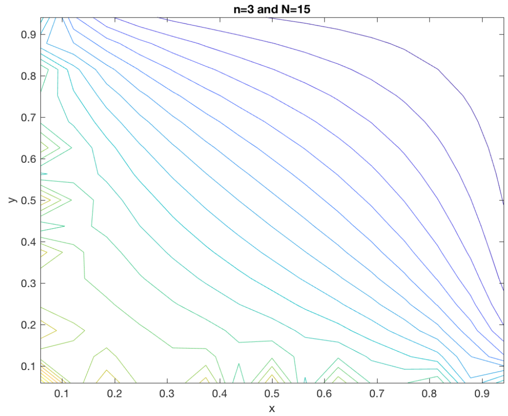
```
function [u, k, res] = sor(lambda, u, x, y, n, tol, maxIter)
    [I, J] = size(u);
    uold = u;
    k = 0;
    mstop = 1;
    res = zeros(1,maxIter);
    residual0 = 0;
    while(mstop && k < maxIter)
        k = k + 1;
        for i = 1:I
            for j = 1:J
                if (i == I)
                    uip1j = 0;
                    uim1j = u(i-1, j);
                elseif (i == 1)
                    uip1j = u(i+1,j);
                    uim1j = sin(3*n*pi*y(j))^2;
                else
                    uip1j = u(i+1,j);
                    uim1j = u(i-1, j);
                end

                if (j == J)
```

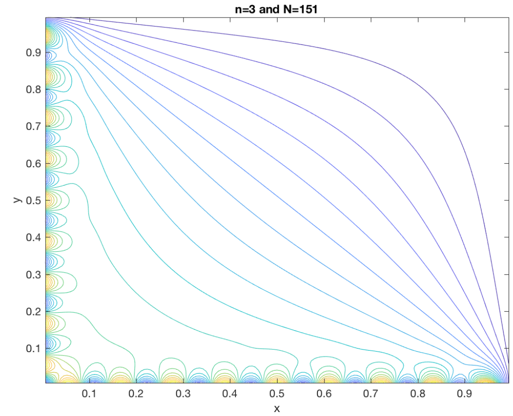
```
                    uijp1 = 0;
                    uijm1 = u(i,j-1);
                elseif (j == 1)
                    uijp1 = u(i,j+1);
                    uijm1 = sin(3*n*pi*x(i))^2;
                else
                    uijp1 = u(i,j+1);
                    uijm1 = u(i,j-1);
                end

                u(i, j) = (1 - lambda)*u(i, j) + lambda*0.25*(uip1j + uim1j + uijp1 +
                    ↪ uijm1);
            end
        end

        deltaU = u - uold;
        residual = sqrt(sum(sum(deltaU.^2))/(I*J));
        res(k) = residual;
        if (k == 1)
            residual0 = residual;
        else
            if (residual/residual0 <= tol)
                mstop = 0;
            else
                uold = u;
            end
        end
    end
    res = res(1:k);
end
```

The following are the contour plots that for the different values of n and N.

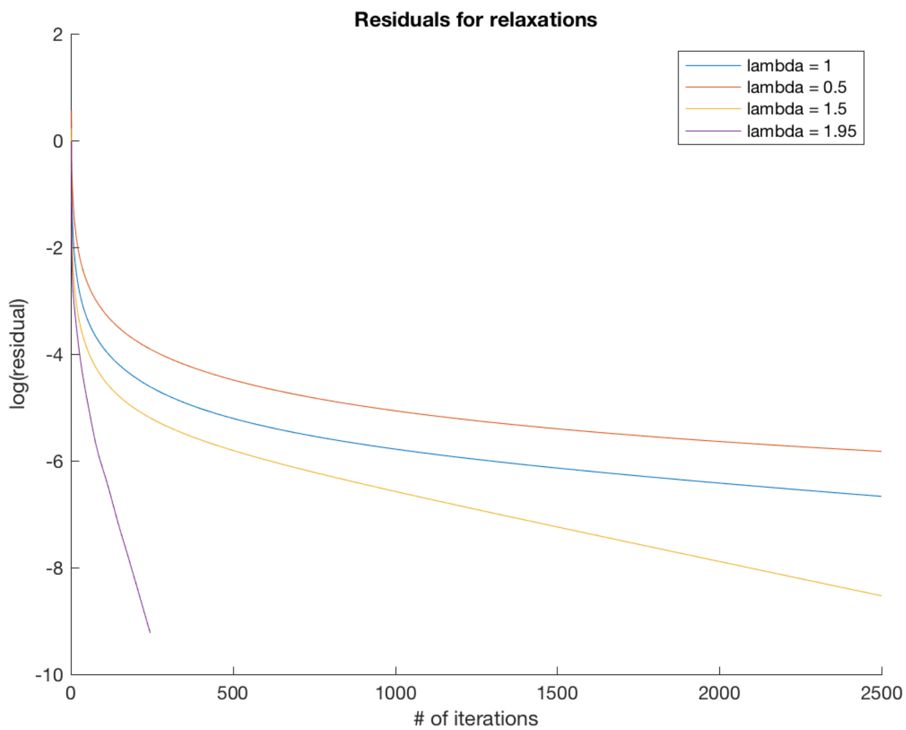








The following plot shows how the residual decreases with iteration for different values of λ. Note that only for $\lambda = 1.95$, did Gauss-Seidel with SOR converge within 2500 iterations.



2. Solve the Poisson Equation

$$\partial_x^2 \psi + \partial_y^2 \psi = 70 e^{-32\left((x-0.5)^2+(y-0.1)^2\right)}$$

in $0 \leq x, y \leq 1$, with boundary conditions

$$\psi(0, y) = -y$$
$$\psi(1, y) = -y$$
$$\psi(x, 0) = 0$$
$$\psi(x, 1) = -1$$

on the grid

$$x(j) = \frac{e^{2(j-1)/(J-1)} - 1}{e^2 - 1} \quad 1 \le j \le J$$
$$y(j) = \frac{e^{2(k-1)/(K-1)} - 1}{e^2 - 1} \quad 1 \le k \le K.$$

- Provide the algorithm part of your code.
- Provide a plot of the grid and a line-contour plot of the converged solution.

In order to solve this equation on this new grid we must do a change of variables. Since the new variables $\bar{x}$ and $\bar{y}$ depend only on x and y respectively, a new equation can be created on a uniform grid,

$$\frac{\partial^2 \bar{x}}{\partial x^2}\frac{\partial u}{\partial \bar{x}} + \left(\frac{\partial \bar{x}}{\partial x}\right)^2 \frac{\partial^2 u}{\partial \bar{x}^2} + \frac{\partial^2 \bar{y}}{\partial y^2}\frac{\partial u}{\partial \bar{y}} + \left(\frac{\partial \bar{y}}{\partial y}\right)^2 \frac{\partial^2 u}{\partial \bar{y}^2} = 70 e^{-32\left((\frac{e^{2\bar{x}}-1}{e^2-1} - 0.5)^2 + (\frac{e^{2\bar{y}}-1}{e^2-1} - 0.1)^2 \right)}$$

where

$$y = \frac{e^{2\bar{y}} - 1}{e^2 - 1}$$
$$x = \frac{e^{2\bar{x}} - 1}{e^2 - 1}.$$

Simplifying this equation gives

$$e^{-4\bar{x}}\left(\frac{\partial u}{\partial \bar{x}} - \frac{1}{2}\frac{\partial^2 u}{\partial \bar{x}^2}\right) + e^{-4\bar{y}}\left(\frac{\partial u}{\partial \bar{y}} - \frac{1}{2}\frac{\partial^2 u}{\partial \bar{y}^2}\right) = \frac{-140}{(e^2-1)^2} e^{-32\left(\left(\frac{e^{2\bar{x}}-1}{e^2-1} - 0.5\right)^2 + \left(\frac{e^{2\bar{y}}-1}{e^2-1} - 0.1\right)^2\right)}$$

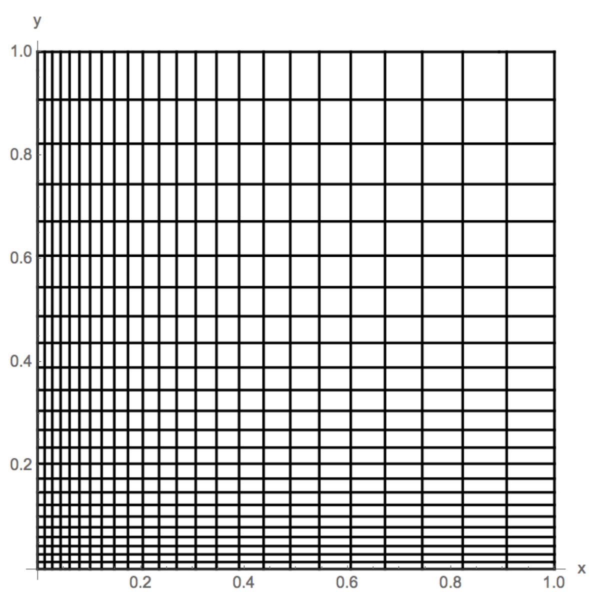
The following functions solve this modified equation.

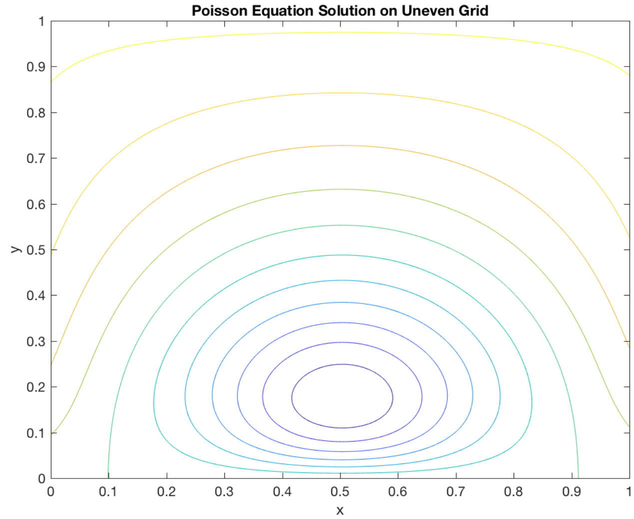
```
function [Au] = multByA2(u, x, y, deltaX, deltaY)
    [I, J] = size(u);
    Au = zeros(I, J);

    for i = 1:I
        for j = 1:J
                % x = 1
                if (i == I)
                    uip1j = -y(j);
                    uim1j = u(i-1, j);
                % x = 0
                elseif (i == 1)
                    uip1j = u(i+1, j);
                    uim1j = -y(j);
                else
                    uip1j = u(i+1, j);
                    uim1j = u(i-1, j);
                end

                % y == 1
                if (j == J)
                    uijp1 = -1;
                    uijm1 = u(i,j-1);
                % y == 0
                elseif (j == 1)
```

```
                uijp1 = u(i,j+1);
                uijm1 = 0;
            else
                uijp1 = u(i,j+1);
                uijm1 = u(i,j-1);
            end

            Au(i, j) = exp(-4*x(i))*((uip1j - uim1j)/(2*deltaX) - (uip1j - 2*u(i,j)
                ↪  + uim1j)/(2*deltaX^2)) ...
            + exp(-4*y(j))*((uijp1 - uijm1)/(2*deltaY) - (uijp1 - 2*u(i, j) + uijm1
                ↪ )/(2*deltaY^2));
        end
    end
end
```

```
function [u] = solvePoisson(multByA, rhs, I, J)
    A = zeros(I*J);
    iter = 0;
    for j = 1:J
        for i = 1:I
            iter = iter + 1;
            e = zeros(I, J);
            e(i, j) = 1;
            temp = multByA(e);
            for k = 1:J
                A(((k-1)*I+1):k*I,iter) = temp(:,k);
            end
        end
    end
    sol = A\rhs;
    u = zeros(I, J);
    for k = 1:J
        u(:,k) = sol(((k-1)*I+1):k*I);
    end
end
```

The following two images show the grid and the contour plot of the solution.





3. Solve the Poisson equation

$$\partial_x^2\psi + \partial_y^2\psi = 2000(\sin(4\pi x)\sin(4\pi y))^5$$

on the domain $0 \le x \le 1$, $0 \le y \le 1$, with boundary conditions

$$\begin{aligned}
\psi(x,0) &= \sin^2(\pi x) + 5\sin^2(4\pi x) + 10\sin^2(8\pi x)\\
\psi(0,y) &= \sin^2(4\pi y)\\
\psi(x,1) &= 0\\
\psi(1,y) &= 0
\end{aligned}$$

Use a 151×151 grid. Submit only the algorithm part of your code. Plot $\log_{10}()$ of the residual - defined at $\|\Delta\psi\|_{L_2}/\|\Delta\psi(0)\|_{L^2}$ - versus iteration. Stop when the residual goes below 10^{-5}. Provide a contour plot of the converged solution.

The following is my conjugate gradient algorithm.

```
function [x, iter, res] = conjugateGradient(multByA, x0, rhs, tol, maxIterations)
    x = x0;
    Ax = multByA(x);
    r = rhs - Ax;
    rsold = sum(dot(r,r));

    res = zeros(1,maxIterations);

    p = r;

    iter = 0;
    while(iter < maxIterations)
        iter = iter + 1;

        Ap = multByA(p);
        pAp = sum(dot(p, Ap));
        alpha = rsold/pAp;
        x = x + alpha*p;
        r = r - alpha*Ap;

        rsnew = sum(dot(r,r));
```

```
        res(iter) = rsnew;
        if(rsnew/res(1) < tol)
            rsold = rsnew;
            break;
        else
            p = r + rsnew/rsold*p;
            rsold = rsnew;
        end
    end
    if (iter == maxIterations && sqrt(rsold) > tol)
        disp('Conjugate Gradient did not converge');
    end
    res = res(1:iter);
end
```

The following images show a contour plot of the converged solution and a plot of the residual. The residual decreases as fast as Gauss-Seidel with SOR with weight $\lambda = 1.95$.