

# 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  $0 \leq x, y \leq 1$ , 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  $15 \times 15$  and  $151 \times 151$  grids for both  $n$  values.
- For the  $151 \times 151$  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  $\log_{10}(\text{residual}/\text{residual}_0)$  versus iteration number. Use the  $L_2$  norm

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

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

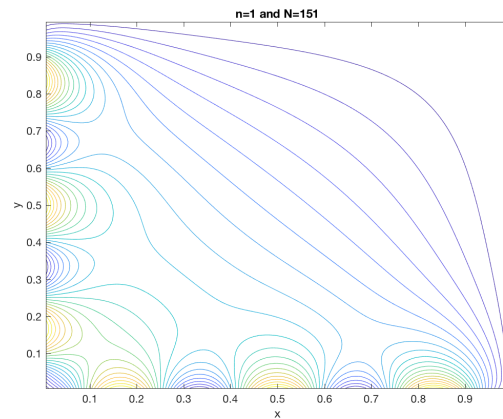
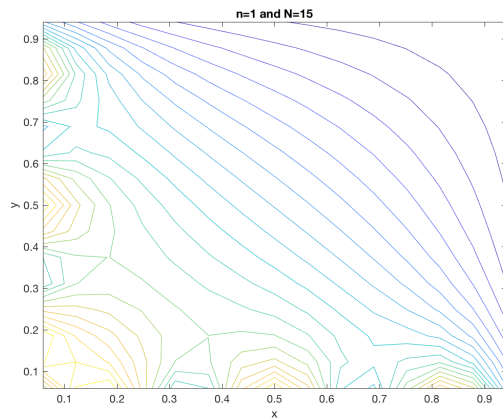
    u(i, j) = (1 - lambda)*u(i, j) + lambda*0.25*(uiplj + uimlj + uijp1 +
        ↪ uijm1);

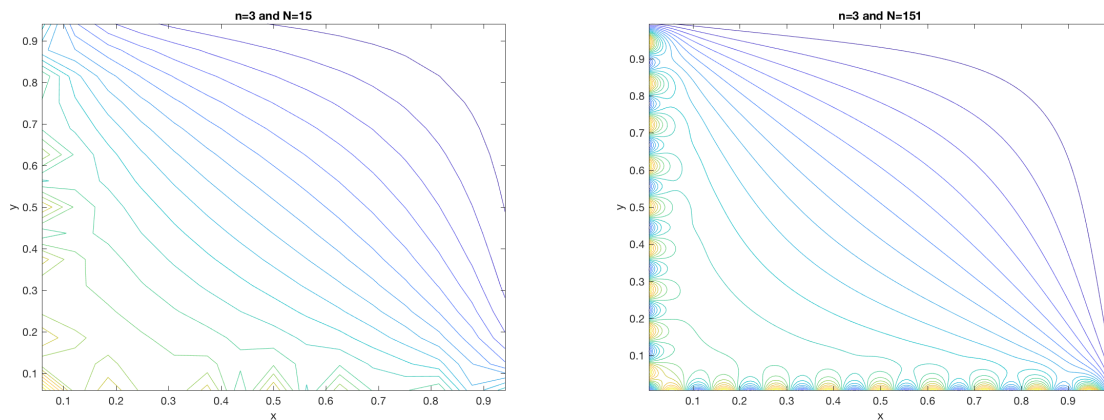
    end
end

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

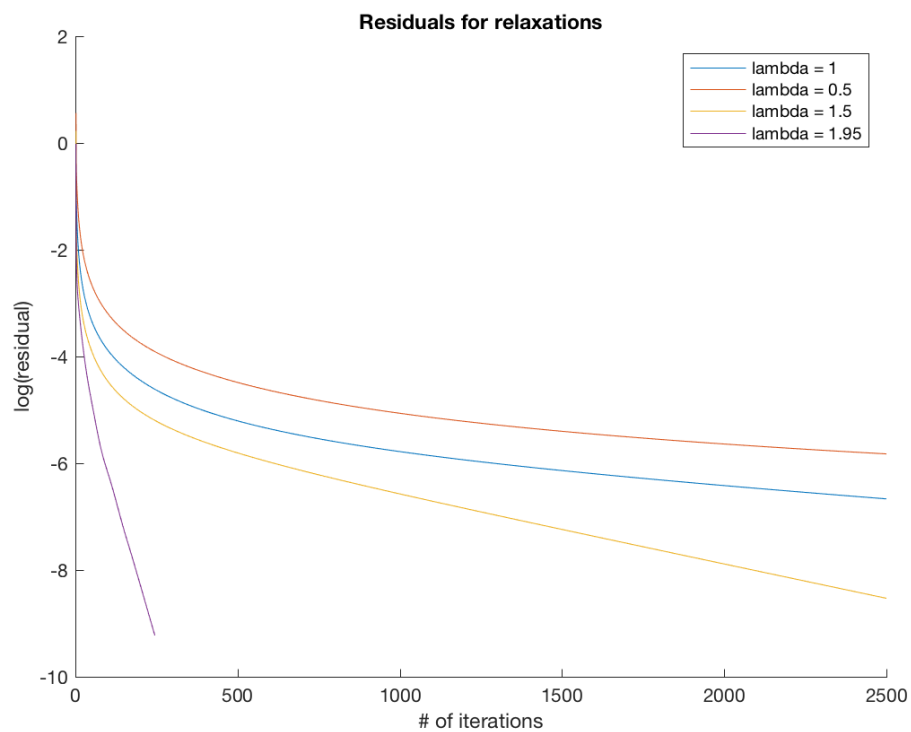
```

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  $\lambda$ . Note that only for  $\lambda = 1.95$ , did Gauss-Seidel with SOR converge within 2500 iterations.



## 2. Solve the Poisson Equation

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

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

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

- Provide the algorithm part of your code.
- Provide a plot of the grid and a line-contour plot of the converged 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  $0 \leq x \leq 1, 0 \leq y \leq 1$ , with boundary conditions

$$\psi(x, 0) = \sin^2(\pi x) + 5 \sin^2(4\pi x) + 10 \sin^2(8\pi x)$$

$$\psi(0, y) = \sin^2(4\pi y)$$

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

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

Use a  $151 \times 151$  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
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.

