# CE-580

**COMPUTATIONAL TECHNIQUES**

**FOR**

**FLUID DYNAMICS**

## HOMEWORK #6

## Determination of Best Overrelaxation Parameter for PSOR Method

**Taha Yaşar Demir**

**1881978**

CE 580 COMPUTATIONAL TECHNIQUES FOR FLUID DYNAMICS

Homework- 6

## Determination of Best Overrelaxation Parameter for PSOR Method

Write a computer program for solution of Poisson type equations using Point Successive Overrelaxation (PSOR) method. Improve your program to determine the **best overrelaxation parameter**, $\omega_{opt}$ , when the geometry of the computational domain, boundary conditions and the grid distribution are given.

### A) Determination of $\omega_{opt}$

1. Consider the analytical solution given as: $U(x,y) = x^2 - y^2$
   on the square domain $0 \le x \le 1$ and $0 \le y \le 1$
2. Generate NxN computational grid using $\Delta x = \Delta y = 1/(N-1)$.
3. Compute the solution $U_{i,j}$ from the expression given above.
4. Consider the finite difference solution $S_{i,j}$ and set the boundary values $S_{1,j}$, $S_{N,j}$, $S_{i,1}$ and $S_{i,N}$ from $U_{i,j}$.
5. Set the initial values $S_{i,j} = 0$ for all **internal nodes**.
6. Perform 20 iterations of PSOR for a given $\omega$.
7. Compute the overall space average error:

$$Error = \frac{\sum_{i=2}^{N-1}\sum_{j=2}^{N-1}\left|U_{i,j} - S_{i,j}\right|}{(N-2)(N-2)}$$

8. Print out Error and $\omega$
9. Repeat steps 5 to 8 for $1 \le \omega \le 2$ with $\Delta\omega=0.002$
10. Repeat the above procedure for N=21, 31, 41, 51, 61, 81 and 101
11. Make a plot of log(Error) vs $\omega$ for all N and determine the $\omega_{opt}$ values. Present the results in a table.

### B) Variation of error with number of iterations

Modify your program to obtain another set of outputs:

12. Set N=51
13. Set the initial data and boundary values as in part **A**
14. Start PSOR iterations and print out Error and iteration count at each iteration. Perform 1000 iterations.
15. Repeat steps 13 and 14 for $\omega$=1.6, 1.7, 1.8, 1.9, 1.99
16. Make a plot of log(Error) vs iteration count for each $\omega$ value.
17. Write a discussion on the results you obtained.

## Calculations

Model equations are

$$Laplace\ Equation: \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

$$Poisson\ Equation: \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = s$$

Numerical solutions to above equations are the same, both are solved using so called five point formula

$$Five - Point\ Formula: \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = s_{i,j} \quad O[(\Delta x)^2, (\Delta y)^2]$$

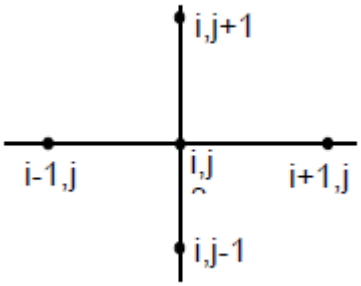


Figure 1: Five Point Stencil

Five-Point formula results in a system of K=(N-2)x(M-2) simultaneous linear algebraic equations and can be solved with direct and iterative methods. In this case we will use Point-Successive Over-Relaxation (PSOR) method which is an iterative method.

PSOR method is derived below

The derivatives can be expressed as

$$u_{xx} = au_{i+1,j} + bu_{i,j} + cu_{i-1,j}$$

$$u_{yy} = du_{i,j+1} + eu_{i,j} + fu_{ij-1}$$

Substituting into the Poisson equation and solving for $u_{i,j}$

$$u_{i,j} = -\frac{1}{b+e}\left(au_{i+1,j} + cu_{i-1,j} + du_{i,j+1} + fu_{ij-1} - s_{i,j}\right)$$

Which can be used to estimate $u_{i,j}^{n+1}$ as an explicit method running in positive I and j diretions

$$u_{i,j}^{n+1} = -\frac{1}{b+e}\left(au_{i+1,j}^{n} + cu_{i-1,j}^{n+1} + du_{i,j+1}^{n} + fu_{ij-1}^{n+1} - s_{i,j}^{n}\right)$$

Adding $\left(u_{i,j}^{n} - u_{i,j}^{n}\right)$ to the right-hand side and regrouping gives

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{1}{b+e}\left(au_{i+1,j}^{n} + cu_{i-1,j}^{n+1} + du_{i,j+1}^{n} + fu_{ij-1}^{n+1} + (b+e)u_{i,j}^{n} - s_{i,j}^{n}\right)$$

Or

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \omega R_{i,j}^{n+1}$$

Where $\omega$ is overrelaxation factor and $1 < \omega < 2$

If we assume $\Delta x = \Delta y = \Delta$ above relation becomes

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \omega R_{i,j}^{n+1}$$

$$R_{i,j}^{n+1} = \frac{1}{4}\left[u_{i+1,j}^{n} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n} + u_{ij-1}^{n+1} + 4u_{i,j}^{n} - \Delta^2 s_{i,j}^{n}\right]$$

***Problem and Domain***

Domain is an 1x1 square with constant division

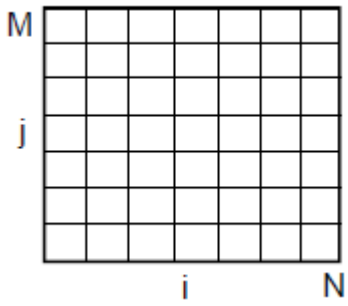


Figure 2: Square Domain (M=N)

Equation to be solved and used as boundary conditions is

$$U(x,y) = x^2 - y^2$$

Which satisfies Poisson equation

$$\frac{\partial^2 U(x,y)}{\partial x^2} + \frac{\partial^2 U(x,y)}{\partial y^2} = s = \frac{\partial^2 (x^2 - y^2)}{\partial x^2} + \frac{\partial^2 (x^2 - y^2)}{\partial y^2}$$

Taking derivative

$$\frac{\partial (2x)}{\partial x} + \frac{\partial (-2y)}{\partial y} = s$$

Taking derivative once again

$$2 - 2 = s = 0$$

As seen above source term is zero and Laplace's equation is to be solved with different grid numbers

***Boundary Conditions***

Boundary conditions is set from the analytical solution on boundary domain as

$$S(1,j) = U(1,j)$$

$$S(N,j) = U(N,j)$$

$$S(i,1) = U(i,1)$$

$$S(i,N) = U(i,N)$$

Where

$$x(1) = y(1) = 0.$$

$$x(N) = y(N) = 1.$$

***Error Calculation***

Error is calculated as

$$Error = \frac{\sum_{i=2}^{N-1}\sum_{j=2}^{N-1}\left|U_{i,j} - S_{i,j}\right|}{(N-2)(N-2)}$$

## Results and Discussion

### Part A

In this part above problem is iterated 20 times with for all values of $\omega$ between $1 < \omega < 2\ with\ \Delta\omega = 0.002$

Gird numbers and best $\omega$ values are tabulated below

| Grid Number (N) | $\omega$ | Error |
|---|---|---|
| 21 | 1.630 | 2.493E-04 |
| 31 | 1.786 | 4.791E-03 |
| 41 | 1.864 | 1.396E-02 |
| 51 | 1.900 | 2.495E-02 |
| 61 | 1.926 | 3.619E-02 |
| 81 | 1.956 | 5.559E-02 |
| 101 | 2.000 | 6.141E-02 |

Table 1: Best Possible $\omega$ values

Considering all cases has same boundary conditions and same domain shape, it is evident that best omega values are depend on grid number.

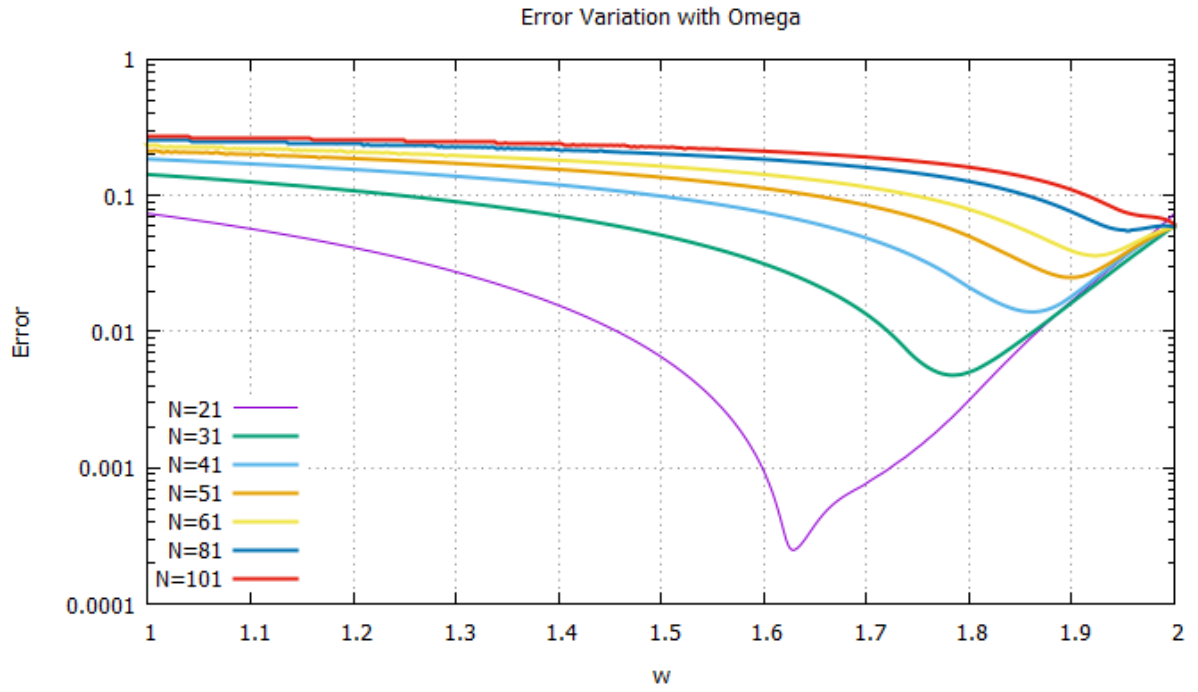


Figure 3: Error w.r.t Omega

Figure 3 shows the error variation with respect to omega, higher relaxation factor does not mean faster convergence. Gird number N=21 gives the best performance since the solution is limited to 20 iteration. Since an explicit solution is used, an information on the boundary marches slowly in domain. With 21 grid nodes, information on the boundary reaches everywhere in domain in 20 iterations, that is why the error is smallest in this case. When grid number is higher, probably internal nodes are not updated according to boundary conditions and introduce errors.

## Part B

In this part, domain has 51x51 grid and it is fixed. Iteration number is also fixed to 1000. Relaxation factors used are

$$\omega = 1.6\ ;\ 1.7\ ;\ 1.8\ ;\ 1.9;\ 1.99$$

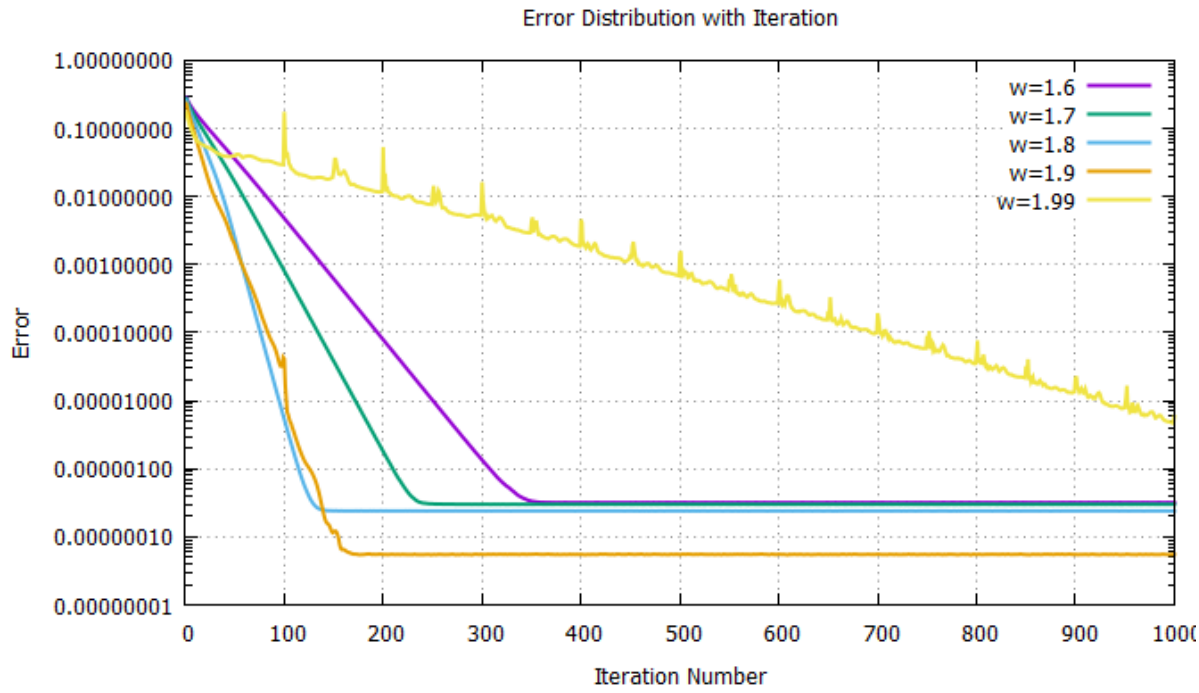


Figure 4: Error Variation with Iteration Count

The effect of omega number to error magnitude is shown in figure 4. As previously mentioned, higher omega does not mean faster convergence. When $\omega = 1.99$ , solution oversteps and misses the optimum point and fluctuations occur. For $w = 1.9$, there are still small fluctuations and for $\omega = 1.8$ fluctuations are removed. Considering fluctuation presence and final error magnitude, it can be said that best $\omega$ is between 1.8 and 1.9 for grid number 51.

## Source Code

```
      program OverRelaxation
c..Taha Yaşar Demir / 1881978
c..CE580 - HomeWork #6 PART-A
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      open(11,file="omega.dat")

      print*, "Enter division number"
      read*, N

      w = 1.
      call grid
      do while (w.le.2.)
            call init
            do k=1,20 ! solution iteration
                  call solution
            enddo
            call error_cal
            call output
            old_e = e
            w = w + dw
      enddo
      print*, w,e
      close(11)

      stop
      end
c-----------------------------------------------------------------------
      subroutine grid
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      dw   = 0.002
      dx   = 1./(N-1)
      dy   = 1./(N-1)
      x(1) = 0.
      y(1) = 0.

      do i=2,N
            x(i) = x(i-1) + dx
            y(i) = y(i-1) + dy
      enddo
```

```
      return
      end
c-----------------------------------------------------------------------
      subroutine init
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      do i=1,N
            do j=1,N
                  s(i,j) = 0. ! Initialize internal points
                  u(i,j) = x(i)**2 - y(j)**2 ! Analytical solution
            enddo
      enddo

      ! boundary conditions
      do i=1,N
            s(i,1) = u(i,1)
            s(i,N) = u(i,N)
      enddo
      do j=1,N
            s(1,j) = u(1,j)
            s(N,j) = u(N,j)
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine solution
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      do i=2,N-1
            do j=2,N-1
                  r(i,j) = 0.25*(s(i+1,j) + s(i-1,j)+
     +                        s(i,j+1) + s(i,j-1) - 4*s(i,j))
                  s(i,j) = s(i,j) + w*r(i,j)
            enddo
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine error_cal
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
```

```
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      e = 0. ! e : Error

      do i=2,N-1
            do j=2,N-1
                  e = e + abs(u(i,j)-s(i,j))/((N-2)*(N-2))
            enddo
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine output
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      write(11,*) w,e

      if (old_e.lt.e) print*, w,e   ! print out best omega value

      return
      end
```

Source Code for Part A

```
      program OverRelaxation
c..Taha Yaşar Demir / 1881978
c..CE580 - HomeWork #6 PART-B
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      open(11,file="error.dat")

      print*, "Enter Omega Value"
      read*, w

      call grid
      call init
      do k=1,1000 ! solution iteration
            call solution
            call error_cal
            call output(k)
      enddo

      close(11)

      stop
      end
c-----------------------------------------------------------------------
      subroutine grid
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      N    = 51
      dx   = 1./(N-1)
      dy   = 1./(N-1)
      x(1) = 0.
      y(1) = 0.

      do i=2,N
            x(i) = x(i-1) + dx
            y(i) = y(i-1) + dy
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine init
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
```

```
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      do i=1,N
            do j=1,N
                  s(i,j) = 0. ! Initialize internal points
                  u(i,j) = x(i)**2 - y(j)**2 ! Analytical solution
            enddo
      enddo

      ! boundary conditions
      do i=1,N
            s(i,1) = u(i,1)
            s(i,N) = u(i,N)
      enddo
      do j=1,N
            s(1,j) = u(1,j)
            s(N,j) = u(N,j)
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine solution
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      do i=2,N-1
            do j=2,N-1
                  r(i,j) = 0.25*(s(i+1,j) + s(i-1,j)+
     +                        s(i,j+1) + s(i,j-1) - 4*s(i,j))
                  s(i,j) = s(i,j) + w*r(i,j)
            enddo
      enddo

      return
      end
c-----------------------------------------------------------------------
      subroutine error_cal
      parameter(mx=101)
      common/mk_grid/   dx,dy,x(mx),y(mx),N
      common/analytic/  u(mx,mx)
      common/numeric/   s(mx,mx),w,dw,r(mx,mx)
      common/error/     e,old_e

      e = 0. ! e : Error

      do i=2,N-1
            do j=2,N-1
```

```
                        e = e + abs(u(i,j)-s(i,j))/((N-2)*(N-2))
                    enddo
            enddo

            return
            end
c-------------------------------------------------------------------
            subroutine output(iter)
            parameter(mx=101)
            common/mk_grid/   dx,dy,x(mx),y(mx),N
            common/analytic/  u(mx,mx)
            common/numeric/   s(mx,mx),w,dw,r(mx,mx)
            common/error/     e,old_e
            integer iter

            write(11,*) iter,e


            return
            end
```

Source Code for Part B