MECH 539 Computational Aerodynamics

Amogha V. Subramanya, 260732978

February 24, 2017

1 Assignment 2

1.1 Question 1

The Laplace equation can be written as,

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

Using Taylor series expansion,

$$u^n_{i+1,j} = u^n_{i,j} + (\Delta x)u_x + \frac{1}{2}(\Delta x)^2 u_{xx} + \frac{1}{6}(\Delta x)^3 u_{xxx} + \frac{1}{24}(\Delta x)^4 u_{xxxx} + \dots$$

and,

$$u^n_{i-1,j} = u^n_{i,j} - (\Delta x)u_x + \frac{1}{2}(\Delta x)^2 u_{xx} - \frac{1}{6}(\Delta x)^3 u_{xxx} + \frac{1}{24}(\Delta x)^4 u_{xxxx} + \dots$$

Adding the above two equations,

$$u^n_{i+1,j} + u^n_{i-1,j} = 2u^n_{i,j} + (\Delta x)^2 u_{xx} + \frac{1}{12}(\Delta x)^4 u_{xxxx} + \dots$$

Rearranging,

$$\frac{\partial^2 u}{\partial x^2} = u_{xx} = \frac{u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}}{(\Delta x)^2} - \frac{1}{12}(\Delta x)^2 u_{xxxx} + \dots$$

Using the same procedure that we followed above we can find u_{yy};

$$\frac{\partial^2 u}{\partial y^2} = u_{xx} = \frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{(\Delta y)^2} - \frac{1}{12}(\Delta y)^2 u_{yyyy} + \dots$$

Adding the above two equations,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}}{(\Delta x)^2} + \frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{(\Delta y)^2} \underbrace{- \frac{1}{12}\Big[(\Delta x)^2 u_{xxxx} + (\Delta y)^2 u_{yyyy} + \dots\Big]}_{\text{Truncation Error}}$$

Therefore,

$$\Rightarrow \text{Truncation Error} = -\frac{1}{12}\Big[(\Delta x)^2 u_{xxxx} + (\Delta y)^2 u_{yyyy} + \dots\Big]$$

1.2 Question 2

The discretization for the Laplace equation is given by,

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

$$\frac{u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}}{(\Delta x)^2} + \frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{(\Delta y)^2} = 0$$

Using Taylor series expansion,

$$u^n_{i+1,j} = u^n_{i,j} + (\Delta x)u_x + \frac{1}{2}(\Delta x)^2 u_{xx} + \frac{1}{6}(\Delta x)^3 u_{xxx} + \frac{1}{24}(\Delta x)^4 u_{xxxx} + \dots$$

and,

$$u^n_{i-1,j} = u^n_{i,j} - (\Delta x)u_x + \frac{1}{2}(\Delta x)^2 u_{xx} - \frac{1}{6}(\Delta x)^3 u_{xxx} + \frac{1}{24}(\Delta x)^4 u_{xxxx} + \dots$$

similarly find $u^n_{i,j+1}$ and $u^n_{i,j-1}$ and then substituting it in the discretization equation, we get the modified equation as;

$$u_{xx} + u_{yy} = -\frac{1}{12}\left[(\Delta x)^2 u_{xxxx} + (\Delta y)^2 u_{yyyy}\right]$$

The scheme is dispersive if the leading term of truncation error is odd, and the scheme would be dissipative if the leading order term of the truncation error is even.

1.3 Question 3

The solution for the Laplace equation using the given boundary conditions for the 400×400 is given by, This solution was calculated by using a second-order

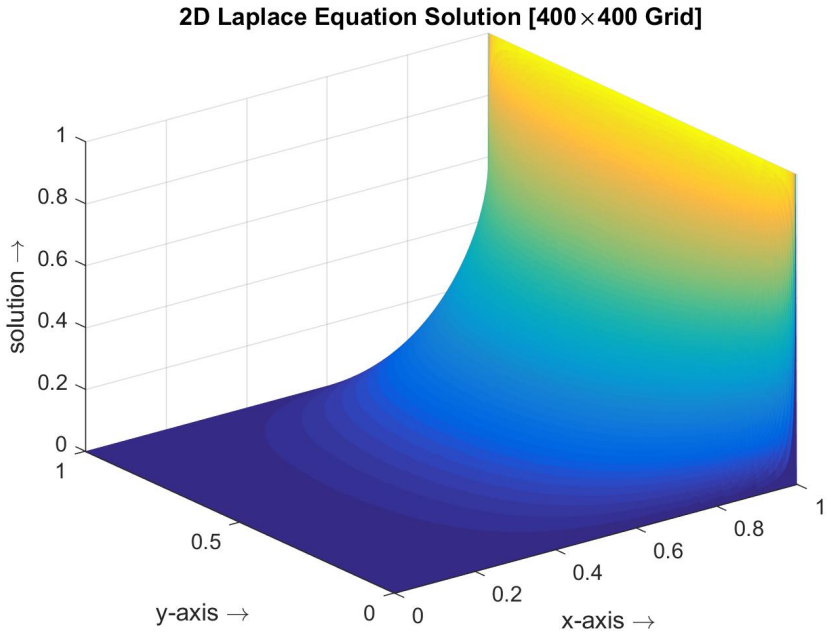

Figure 1: Solution for a 400×400 grid

finite-difference scheme and then solving it using the Gauss-Siedel method using an error tolerance of 1×10^{-6}.

1.4 Question 4

Here, I plot the log of the residual against the number of iterations taken to converge to the solution for all three methods on the same graph. The relaxation parameter for the Successive over relaxation method is taken as 1.1 ($\omega = 1.1$). The following graphs were obtained for the various different grid sizes;

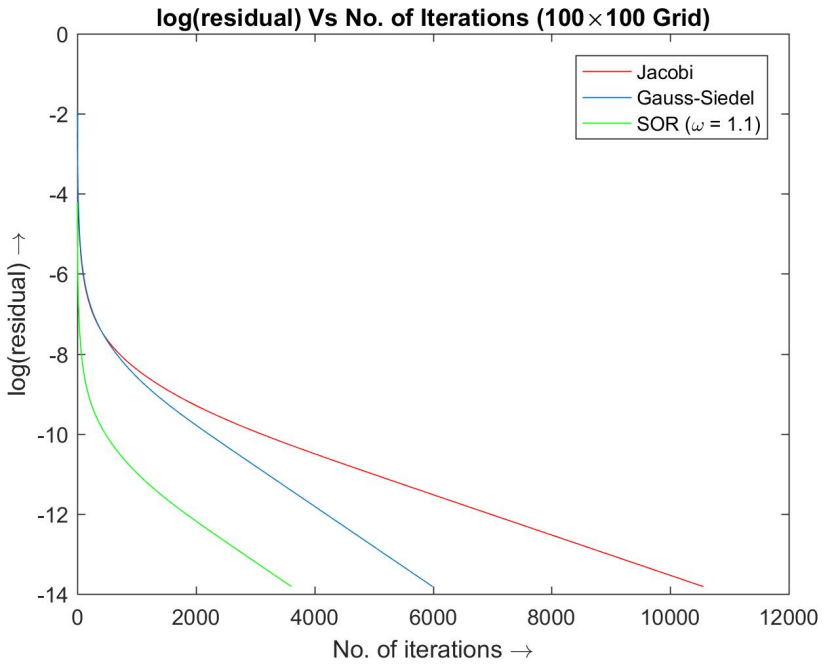

Figure 2: log(residual) Vs Number of Iterations for the 100×100 Grid

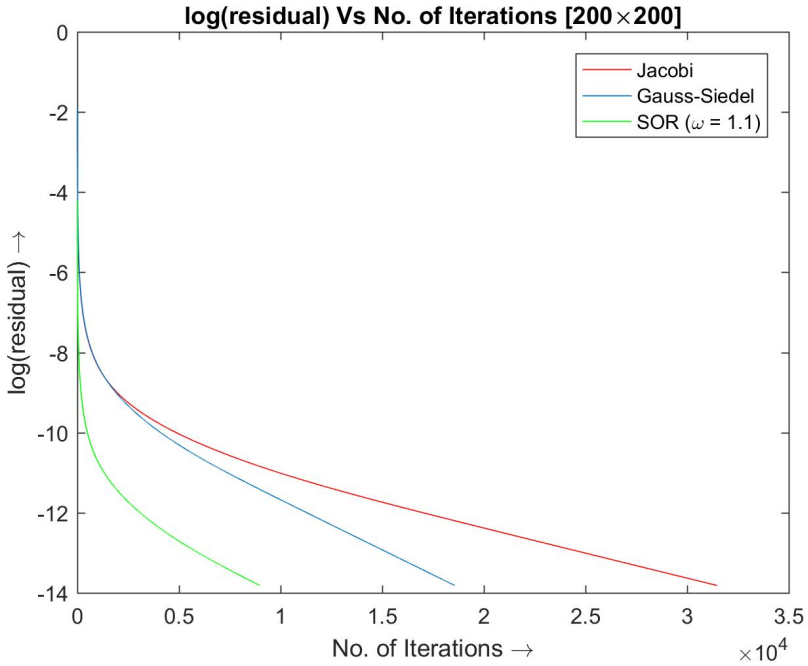

Figure 3: log(residual) Vs Number of Iterations for the 200×200 Grid

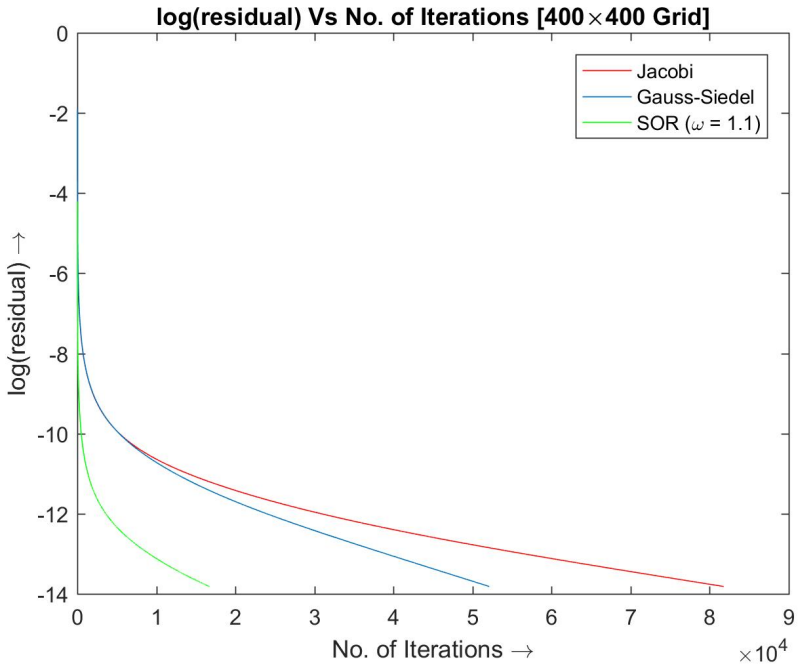

Figure 4: log(residual) Vs Number of Iterations for the 400×400 Grid

From the graphs we can clearly see that the successive over relaxation method converges the fastest to the solution, almost three times faster than the other methods. This is given that we select an appropriate relaxation parameter (In this case I have chosen the relaxation parameter, $\omega = 1.1$. This is discussed in more detail in Question 7.
The Jacobi method is the slowest to converge, taking more than 81000 iterations to converge on a 400×400 grid.
The Gauss-Siedel method is faster than than the Jacobi method to converge to the answer, although it is not faster than the successive over relaxation method. This is because in the Gauss-Siedel method, the values of 'u' are being updated continuously and the most up-to-date value of 'u' is used during every iteration whereas in the Jacobi method, the older value of 'u' is used as it is not updated continuously and instead updated only after one complete iteration of the complete matrix.

1.5 Question 5

In this question, I plot the log of the residual against the computational time required to converge to the solution for the three different schemes used respectively on the same graph. The following plots were obtained for the various different grid sizes;

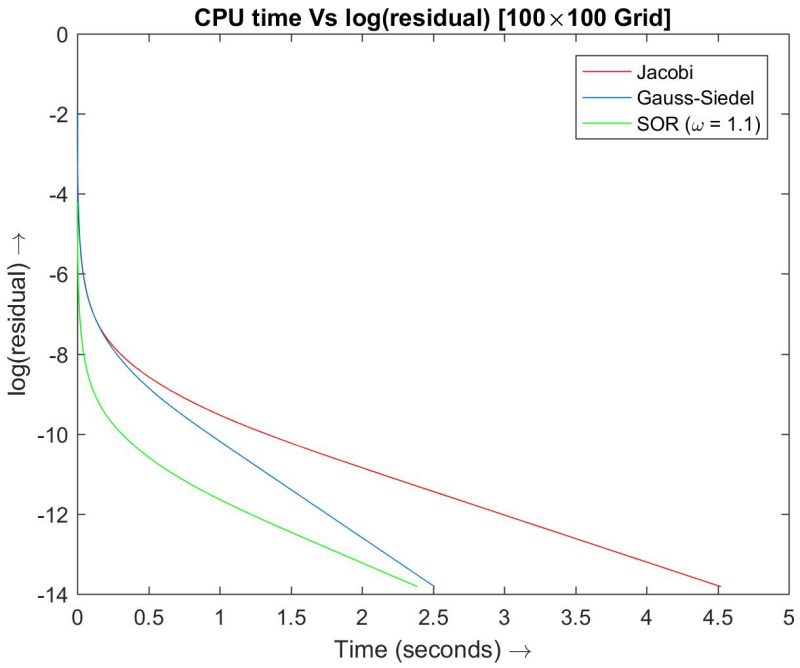

Figure 5: log(residual) Vs Computational Time for the 100×100 Grid

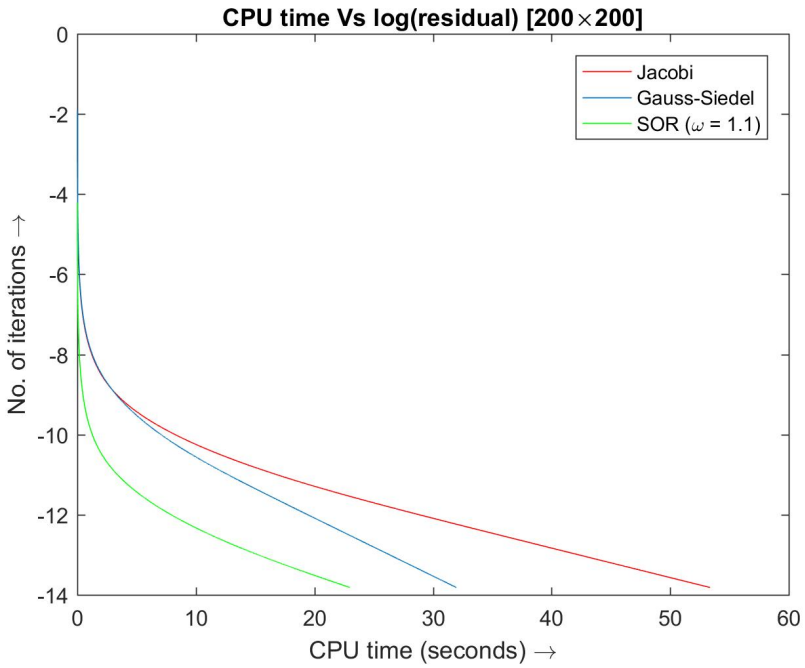

Figure 6: log(residual) Vs Computational Time for the 200×200 Grid

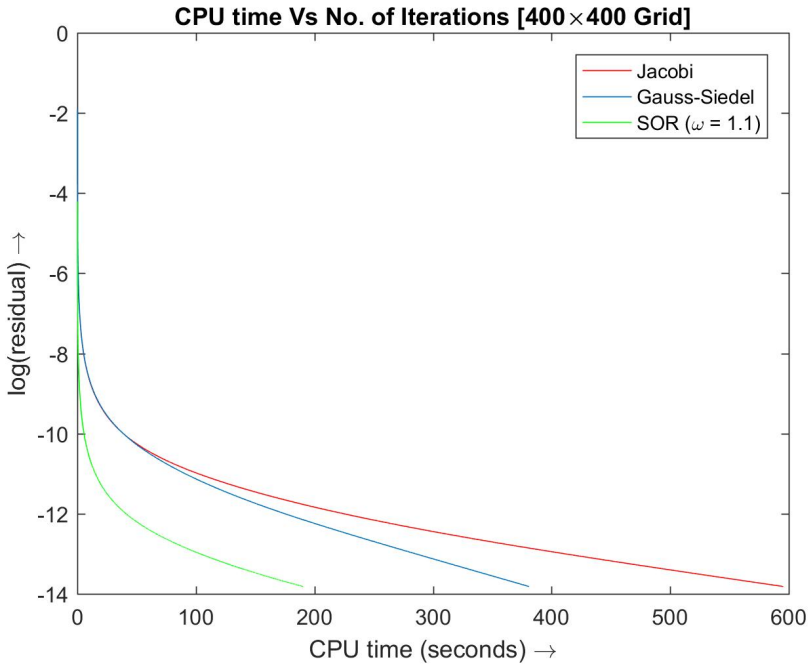

Figure 7: log(residual) Vs Computational Time for the 400×400 Grid

The Jacobi scheme requires two arrays to store the values of 'u'(the solution) and one vector to store the residual of the scheme. The Jacobi scheme also takes the longest amount of computational time to converge to an answer.
The Gauss-Siedel scheme on the other hand requires only one array as we don't need to store the old values of 'u' as they are continuously updated and used. The Gauss-Siedel scheme is the second fastest to converge to an answer after the successive over relaxation scheme.
The succession over relaxation scheme has the lowest computational time required to converge to answer, although this depends on the relaxation parameter that we choose (explained in question 7), in this case we chose the relaxation parameter, $\omega = 1.1$,which is the optimum value hence it converges really fast.
In terms of memory usage, the successive over relaxation is scheme is the most expensive as it requires three different arrays to process; although this is the least expensive in terms of computation. The second most expensive scheme in term of memory usage is the Jacobi scheme which requires two different arrays to converge to the solution. The least expensive in terms of memory usage among the three schemes here is the Gauss-Siedel scheme, which requires only one array to carry out its operations.

1.6 Question 6

In this question I plot the log of the residual between the solution calculated by the SOR scheme and the actual solution. The following graph was obtained:

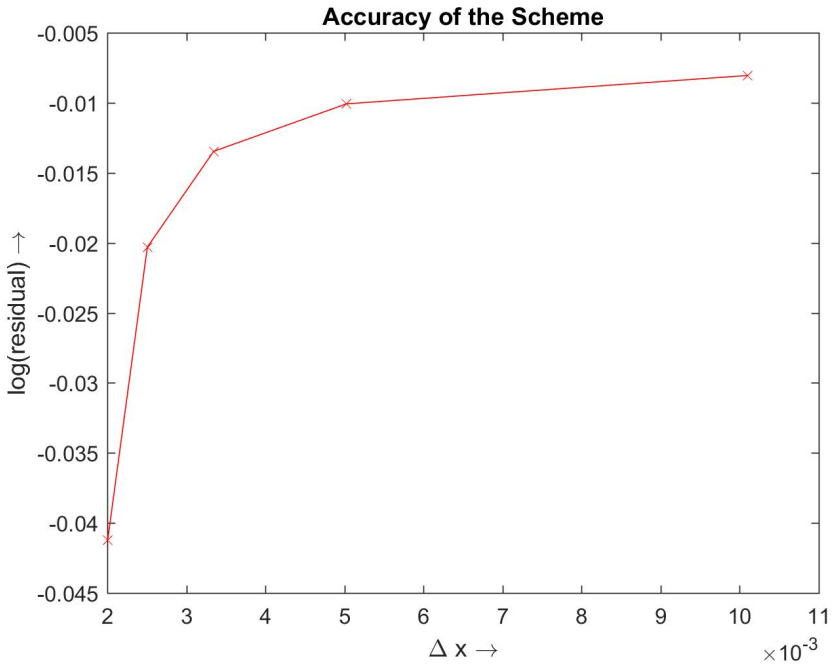

Figure 8: Accuracy of the scheme used compared to the analytical solution

1.7 Question 7

In this question, we check how different values of the relaxation parameter affects the Successive Over Relaxation scheme. To find out, I solved the Successive Over Relaxation schemes for different values of the relaxation parameter from 0.5 to 1.9 and since we are also interested in finding out if the optimum relaxation parameter varies for different grid sizes, I solved it for all the different grid sizes, my results are summarized in the bar graph below, in which i plotted the number of iterations it took the scheme to arrive at the solution for various different values of the relaxation parameter.

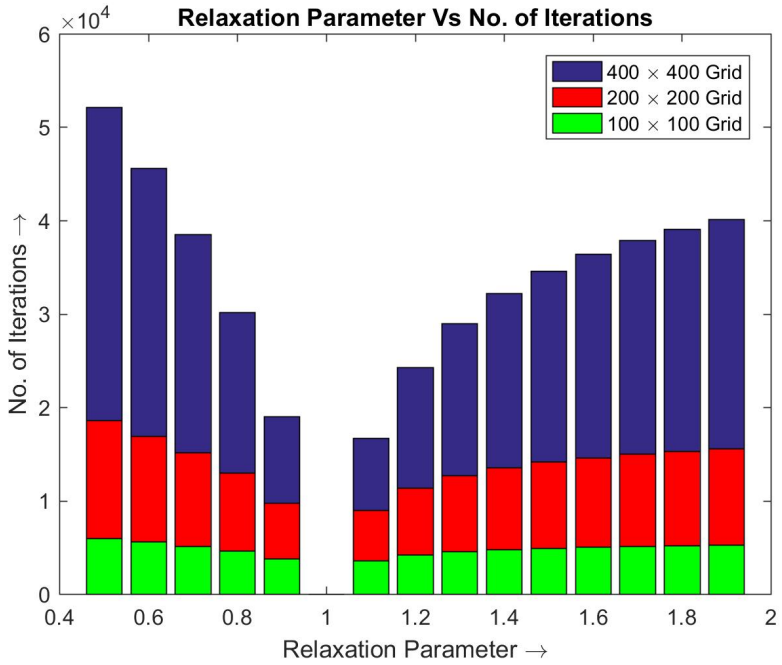

Figure 9: Effect of the Relaxation parameter on the number of iterations

As we can see from the bar graph above, there does exist an optimum value of the relaxation parameter, ω which in our case is 1.1. This optimum value of the relaxation parameter allows the scheme to converge to the solution in the least number of iterations.
Also, we can see that this optimum value is constant and does not depend on the size of the grid, and consistently converges the fastest no matter the size of the grid that we choose.
In the following graph I have plotted the log of the residual against the number of iterations for some of the various different values of the relaxation parameter while solving the 400×400 grid, and we can visually see how the solution converges faster for the optimum value of the relaxation parameter;

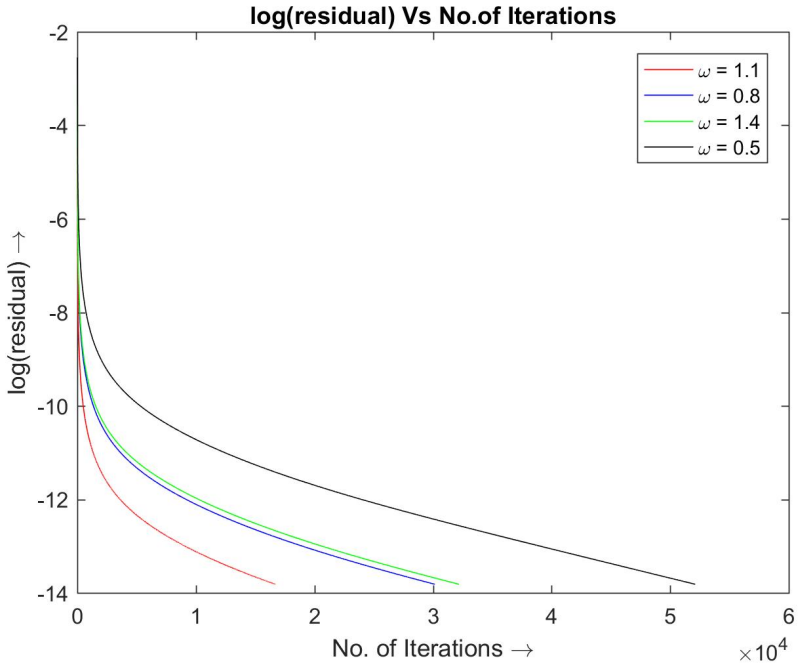

Figure 10: log of residual Vs number of iterations for various ω

2 Bonus Questions

2.1 Question 1

The linear advection equation is given by,

$$\frac{\delta u}{\delta x} + A\frac{\delta u}{\delta t} = 0$$

The leap frog scheme for solving this differential equation is given by,

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} + A\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0$$

If ϵ denotes the round off error, 'D' the exact solution to the algorithm and 'N' the solution we get by using single or double precision variables, we can say

$$N = D + \epsilon$$

If we substitute the above equation in the leap frog algorithm, 'D' cancels out on both sides as it is the exact solution and we are left with,

$$\frac{\epsilon_j^{n+1} - \epsilon_j^{n-1}}{2\Delta t} + A\frac{\epsilon_{j+1}^n - \epsilon_{j-1}^n}{2\Delta x} = 0$$

Let us assume that the error has a solution,

$$\epsilon = \sum_m b_m(t) e^{ik_m x}$$

where $b_m(t)$ are Fourier coefficients and k_m is the wave number. As $b_m(t)$ is only a function of time we can write the expression as,

$$\epsilon = e^{at} e^{ik_m x}$$

Substituting the above relation for ϵ in the algorithm, we get,

$$\frac{e^{a(t+\Delta t)} e^{ik_m x} - e^{a(t-\Delta x)} e^{ik_m x}}{2\Delta t} + A \frac{e^{at} e^{ik_m (x+\Delta x)} - e^{at} e^{ikm(x-\Delta x)}}{2\Delta x} = 0$$

Simplifying,

$$e^{a\Delta t} - e^{-a\Delta x} + \frac{A\Delta t}{\Delta x}(e^{ik_m \Delta x} - e^{ikm\Delta x}) = 0$$

Let $\upsilon = \frac{A\Delta t}{\Delta x}$ and $\beta = k_m \Delta x$, and substituting $e^{ik_m \Delta x} - e^{ikm\Delta x} = i2\sin(k_m \Delta x)$,

$$e^{a\Delta t} - e^{-a\Delta x} + i2\upsilon \sin\beta = 0$$

Multiplying by $e^{a\Delta t}$,

$$e^{2a\Delta t} + e^{a\Delta t} i2\upsilon \sin\beta - 1 = 0$$

Solving the above quadratic equation for $e^{a\Delta t}$,

$$e^{a\Delta t} = \frac{-i2\upsilon \sin\beta \pm \sqrt{-4\upsilon^2 \sin^2\beta + 4}}{2}$$

Therefore, we get the amplification factor,

$$\boxed{G = e^{a\Delta t} = \pm\sqrt{-\upsilon^2 \sin^2\beta + 1} - i\upsilon \sin\beta}$$

Hence, proved.

2.2 Question 2

The diffusion equation is given by,

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

Using Taylor series expansion,

$$u_j^{n+1} = u_j^n + \frac{\partial u}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 u}{\partial t^2}(\Delta t)^2 + \dots$$

Now,

$$\frac{\partial u}{\partial t} = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{1}{2}\frac{\partial^2 u}{\partial t^2}(\Delta t) + \dots$$

Again using Taylor series expansion,

$$u_{j+1}^n = u_j^n + \frac{\partial u}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(\Delta x)^2 + \frac{1}{6}\frac{\partial^3 u}{\partial x^3}(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 u}{\partial x^4}(\Delta x)^4 + \dots$$

and,

$$u_{j-1}^n = u_j^n - \frac{\partial u}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(\Delta x)^2 - \frac{1}{6}\frac{\partial^3 u}{\partial x^3}(\Delta x)^3 + \frac{1}{24}\frac{\partial^4 u}{\partial x^4}(\Delta x)^4 + \dots$$

Adding the above two equations,

$$u_{j+1}^n + u_{j-1}^n = 2u_j^n + \frac{\partial^2 u}{\partial x^2}(\Delta x)^2 + \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^4 + \dots$$

Now,

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{j+1}^n + u_{j-1}^n - 2u_j^n}{\Delta x^2} - \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2 + \dots$$

We can now write the following,

$$\frac{\partial u}{\partial t} - \alpha\frac{\partial^2 u}{\partial x^2} = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \alpha\frac{u_{j+1}^n + u_{j-1}^n - 2u_j^n}{\Delta x^2} = -\frac{1}{2}\frac{\partial^2 u}{\partial t^2}(\Delta t) + \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2 + \dots$$

where, Leading Order Term(LOR)= $-\frac{1}{2}\frac{\partial^2 u}{\partial t^2}(\Delta t) + \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2$
Now,

$$\frac{\partial}{\partial t}\Big(\frac{\partial u}{\partial t}\Big) = \frac{\partial}{\partial t}\Big(\alpha\frac{\partial^2 u}{\partial x^2}\Big) = \alpha\frac{\partial^2}{\partial x^2}\Big(\frac{\partial u}{\partial t}\Big) = \alpha^2\frac{\partial^4 u}{\partial x^4}$$

We can now write,

$$LOR = -\frac{\alpha^2}{2}\frac{\partial^4 u}{\partial x^4}(\Delta t) + \frac{1}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2$$

$$\Rightarrow \boxed{LOR = \Big[-\frac{1}{2}\alpha^2(\Delta t) + \frac{1}{12}(\Delta x)^2\Big]u_{xxxx}}$$

Hence, proved.
Finding the stability of the scheme,

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2}$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \alpha\Big[\frac{u_{j+1}^n + u_{j-1}^n - 2u_j^n}{\Delta x^2}\Big]$$

Following a similar procedure to that followed in question 1, by first substituting $N = D + \epsilon$, then D cancels out as it is the exact solution, now substituting $\epsilon = e^{at}e^{ik_m x}$, we get

$$e^{a(t+\Delta t)}e^{ik_m x} - e^{at}e^{ik_m x} = \frac{\alpha\Delta t}{(\Delta x)^2}(e^{at}e^{ik_m(x+\Delta x)} + e^{at}e^{ik_m(x-\Delta x)} - 2e^{at}e^{ik_m x})$$

$$e^{a\Delta t} - 1 = \frac{\alpha \Delta t}{(\Delta x)^2}(e^{ik_m \Delta x} + e^{-ik_m \Delta x} - 2)$$

Substituting $r = \frac{\alpha \Delta t}{(\Delta x)^2}$, $\beta = k_m \Delta x$, and $e^{i\beta} + e^{-i\beta} = 2\cos\beta$;

$$e^{a\Delta t} = 1 + 2r(\cos\beta - 1)$$

Substituting $\cos\beta = 1 - 2\sin^2\frac{\beta}{2}$,

$$e^{a\Delta t} = 1 - 4r\sin^2\frac{\beta}{2}$$

For the scheme to be stable,

$$| G | \leq 1$$

$$| e^{a\Delta t} | \leq 1$$

$$1 - 4r\sin^2\frac{\beta}{2} \leq 1$$

$$-4r\sin^2\frac{\beta}{2} \leq 0$$

$$4r\sin^2\frac{\beta}{2} \geq 0$$

$$\Rightarrow r \geq 0$$

Also,

$$1 - 4r\sin^2\frac{\beta}{2} \leq -1$$

$$-4r\sin^2\frac{\beta}{2} \leq -2$$

$$r\sin^2\frac{\beta}{2} \leq \frac{1}{2}$$

$$\Rightarrow r \leq \frac{1}{2}$$

Therefore, the stability condition is given by,

$$\boxed{0 \leq r \leq \frac{1}{2}}$$