Scientific Computing Exercise Set 1

Adrian Ruesink (12195669) & Ziya Alim Şungkar (14904950)

I. Introduction

Many scientific formulas are very difficult to solve analytically, therefor scientist have had to come up with simpler and easier methods to solve these problems. A common way of doing this is by creating computational models with certain parameters which accurately describe the situation they are modelling. Although there are many ways of doing this, finite difference equations are a simple way of describing formulas which change by iteration (or over time).

This paper will first look at creating models of the wave equation on a vibrating string and the time-dependent 2D diffusion equation using finite difference equations. Furthermore multiple different iteration methods will be studied which are all used to solve the time-independent diffusion equation and will then be compared to other.

II. Theory

A. Vibrating String

The first problem being looked at is the vibrating string where a one-dimensional wave equation is used to describe the movement of the vibrating sting. This equation is

$$\frac{\partial^2 \Psi}{\partial t^2} = c^2 \frac{\partial^2 \Psi}{\partial x^2}, \tag{1}$$

where $\Psi(x,t)$ is the displacement or amplitude of the wave at a certain position and time and c is the wave speed through a medium. The equation shows how the displacement of the wave changes over both space and time.

B. Diffusion

1) Time-Dependent: The time dependent diffusion equation describes how a quantity such as a concentration of gas diffuses over a given medium. It is described using the equation

$$\frac{\partial c}{\partial t} = D\nabla^2 c, \tag{2}$$

where $c(x,y,z,t)$ is the concentration in $molecules/m^3$ and D is the diffusion constant which changes for varying substances. It expresses how the concentration of a diffusing substance changes over time. The diffusion equation is derived from Fick's law which is used to describe the rate of diffusion of through a medium and is represented as

$$J = -D\nabla c, \tag{3}$$

where D is the diffusion constant, c is the concentration and J is the flux, which describes the amount of the substance which passes through a unit area per time. It describes that the rate of diffusion is proportional to the concentration gradient with the diffusion occurring from areas of high concentration to areas of low concentration.

2) Time-independent: The time-independent diffusion equation describes how a substance diffuses through a medium when the concentration gradient is in equilibrium, meaning that it does not change over time. This is described by

$$\nabla^2 c = 0, \tag{4}$$

as $\frac{\partial c}{\partial t}$ from equation 2 is equal to 0. This allows for the understanding of the spacial distribution of a substance in a steady state.

III. Method

A. Vibrating String

1) Discretization of the Wave Equation: Firstly, the wave equation from eq.(1) was discretized, to make it suitable for programming. The finite difference method was implemented. First, the string with length $L = 1$ was divided into N intervals this gave the interval length of $\Delta x = \frac{L}{N}$. Next, the equation was discretized in both space and time. Δt was taken as the time step, and Ψ_i^n as the value of Ψ at position i and time step n, where i ranges from 0 to N and n represents the time step count. The discretized version of the spatial and temporal derivatives are:

$$\frac{\partial^2 \Psi}{\partial x^2} \approx \frac{\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n}{(\Delta x)^2}, \tag{5}$$

$$\frac{\partial^2 \Psi}{\partial t^2} \approx \frac{\Psi_i^{n+1} - 2\Psi_i^n + \Psi_i^{n-1}}{(\Delta t)^2}, \tag{6}$$

This is achieved using the finite difference method,

$$\left.\frac{d^2f}{dx^2}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{(\Delta x)^2}, \quad (7)$$

Subbing eq.(5) and eq.(6) into eq.(1) and solving for Ψ_i^{n+1} gives:

$$\Psi_i^{n+1} = 2\Psi_i^n - \Psi_i^{n-1} + \frac{(\Delta x)^2}{c^2(\Delta t)^2}\left(\Psi_{i+1}^n - 2\Psi_i^n + \Psi_{i-1}^n\right), \quad (8)$$

This equation g the ability to compute the future state of Ψ at any point i based on its current and past states, and the states of its immediate neighbors in space. The $\frac{c^2(\Delta t)^2}{(\Delta x)^2}$ term acts as a scaling factor for the spatial differences.

2) Boundary Conditions Further, a boundary condition is put at both ends of the string, as if it were between an imaginary wall, which are $\Psi(x = 0, t) = 0$ and $\Psi(x = L, t) = 0$, this gives $\Psi_0^n = 0$ and $\Psi_N^n = 0$ for all time steps n

B. Diffusion

1) Time-Dependent: To create a 2 dimensional model of diffusion, firstly equation 2 must be rewritten into 2 dimensions, giving

$$\frac{\partial c}{\partial t} = D(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}), \quad (9)$$

where $c(x, y, t)$ is the concentration in $molecules/m^2$ and D is the diffusion constant. From here, a 2 dimensional lattice is created which is split up into sections along each axis, these sections are divided up into small steps of δx and δy, where δx = δy, meaning that y can be described as $y = m\delta y$ and x as $x = l\delta x$. The time t can be also be split up into steps giving $t = n\delta t$ and allowing the concentration to be described as $c(l\delta x, m\delta y, n\delta t) = c_{l,m}^n$. Using this information, equation 9 can be rewritten into a lattice notation which allows the implementation of the program easier. By taking the Taylor expansion of the time derivative it gives

$$c(x, y, t + \delta t) = c(x, y, t) + \delta t\frac{\partial c}{\partial t} + O(\delta t^2). \quad (10)$$

Here the $O(\delta t^2)$ term can be neglected and the expansion can be rearranged and rewritten into lattice form giving

$$\frac{\partial c}{\partial t} \approx \frac{c(x, y, t + \delta t) - c(x, y, t)}{\delta t} = \frac{c_{l,m}^{n+1} - c_{l,m}^n}{\delta t}. \quad (11)$$

This same method can be applied to the second order Taylor expansions of the spacial derivatives of $\frac{\partial^2 c}{\partial x^2}$ and $\frac{\partial^2 c}{\partial y^2}$ to give their lattice notations. Doing this gives

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{c(x + \delta x, y, t) + c(x - \delta x, y, t) - 2c(x, y, t)}{\delta x^2} = \frac{c_{l+1,m}^n + c_{l-1,m}^n - 2c_{l,m}^n}{\delta x^2}, \quad (12)$$

$$\frac{\partial^2 c}{\partial y^2} \approx \frac{c(x, y + \delta y, t) + c(x, y - \delta y, t) - 2c(x, y, t)}{\delta x^2} = \frac{c_{l,m+1}^n + c_{l,m-1}^n - 2c_{l,m}^n}{\delta x^2}, \quad (13)$$

where both there equations have a denominator of δx as $\delta x = \delta y$ as stated earlier. From here equations 11, 12 and 13 can be placed into equation 9 and rearranged to give an equation which can be used to calculate the value of a section in the lattice for each time step, this equation being

$$c_{l,m}^{n+1} = \frac{D\delta t}{\delta x^2}(c_{l+1,m}^n + c_{l-1,m}^n + c_{l,m+1}^n + c_{l,m-1}^n - 4c_{l,m}^n) + c_{l,m}^n. \quad (14)$$

In the model boundary contiditions were set so that $c(x, y; t = 0) = 0$ and $0 \leq x \leq 1, 0 \leq y \leq 1$. Furthermore $c(x, y = 1; t) = 1$ and $c(x, y = 0; t) = 0$, meaning y=1 is the source of the concentration and y=0 acts as a sink. The final boundary set was that x has a periodic boundary where $c(x = 0, y; t) = c(x = 1, y; t)$. Because of this it was possible to model as a one dimensional problem as for the same values of x the concentration would always be the same, meaning that both neighbours on the x axis are equal to the current lattice point we are looking at, meaning we can rewrite equation 14 as

$$c_{l,m}^{n+1} = \frac{D\delta t}{\delta x^2}(c_{l,m+1}^n + c_{l,m-1}^n - 2c_{l,m}^n) + c_{l,m}^n. \quad (15)$$

This equation makes sure that even at the boundaries of the lattice the simulation still runs correctly. Furthermore for the model, values for δx and δt had to be selected so that

$$\frac{4\delta t D}{\delta x^2} \leq 1, \quad (16)$$

this is necessary for the model to be stable, so values of $D = 1$, $\delta x = 0.05$ and $\delta t = 0.0001$ were used.

To test the correctness of the simulation, it was later compared to the analytical solution of the 2D time-dependent diffusion equation for these boundary conditions, this being

$$c(y,t) = \sum_{i=0}^{\infty} \text{erfc}\left(\frac{1-y+2i}{2\sqrt{Dt}}\right) - \text{erfc}\left(\frac{1+y+2i}{2\sqrt{Dt}}\right) \tag{17}$$

C. Time-Independent Diffusion

For the time-independent diffusion equation there are multiple different methods which were used to simulate. The basis of simulating the time-independent diffusion equation, by applying the same method as before for the time-dependent diffusion equation on equation 4 giving the finite difference equation of

$$\frac{1}{4}(c_{i+1,j} + c_{i-1,j} + c_{i,j+1} + c_{i,j-1}) = c_{i,j} \tag{18}$$

is obtained, however here there is no k as the steps do not depend on time.

1) Jacobi Iteration: The first simulation method used was the Jacobi Iteration, which uses the finite difference equation

$$c_{i,j}^{k+1} = \frac{1}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k} + c_{i,j+1}^{k} + c_{i,j-1}^{k}), \tag{19}$$

where k is each iteration. For the time-dependent method the simulation would run for a certain amount of time steps, however for the time independent methods a stopping condition is needed. In this case the simulation stops when every single lattice point has converged to a point where its previous value has a small enough difference compared to its current point, this difference is marked as δ and is represented by

$$\delta \equiv \max_{i,j} \left| ci, j^{k+1} - c_{i,j}^{k} \right| < \epsilon, \tag{20}$$

meaning that if the delta value for every point is smaller than the chosen epsilon, the model has converged. For this model, the value chosen for $\epsilon = 10^{-5}$.

2) Gauss-Seidel Iteration: Gauss-Seidel Iteration is another method which can be used to model time-independent diffusion and should be an improvement compared to the Jacobi Iteration method as it should converge in less iterations. The finite difference equation used in this method is

$$c_{i,j}^{k+1} = \frac{1}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}). \tag{21}$$

It uses the most updated values available at the time of calculation. If some neighbors of point (i,j) have already been updated in the current iteration $(k+1)$, those updates are used, resulting in faster convergence unlike the *Jacobi Method* only uses values from the previous iteration (k) for all neighbor point.

This method uses the same stopping condition as the Jacobi iteration method, meaning all lattice points need to agree with equation 20 with $\epsilon = 10^{-5}$.

3) Successive Over Relaxation: The Successive Over Relaxation (SOR) method is an improvement over the methods mentioned previously and is obtained from the Gauss-Seidel method by an over-correction of the new iterate. The SOR is represented by the finite difference equation

$$c_{i,j}^{k+1} = \frac{\omega}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}) + (1-\omega)c_{i,j}^{k}, \tag{22}$$

where ω is the relaxation parameter.

The *SOR Method*'s use of ω allows for adjustment of the update magnitude at each iteration. Faster convergence can be achieved by 'overshooting' (for $\omega > 1$) the updates based on the immediate previous iteration's value. Making a SOR method with optimally chosen ω much more efficient than the *Gauss-Seidel Method*.

Once again the same stopping condition is used which can be found in equation 20 with $\epsilon = 10^{-5}$.

IV. Results

A. Vibrating Sting

Eq.(8) is used to calculate the value of Ψ. The string is set to rest at $t = 0$. The string shows different patterns at different initial conditions, and its periodic movements are recorded between $t = 0$ and $t = 0.2$, as shown in Fig.(1), Fig.(2), and Fig.(3)

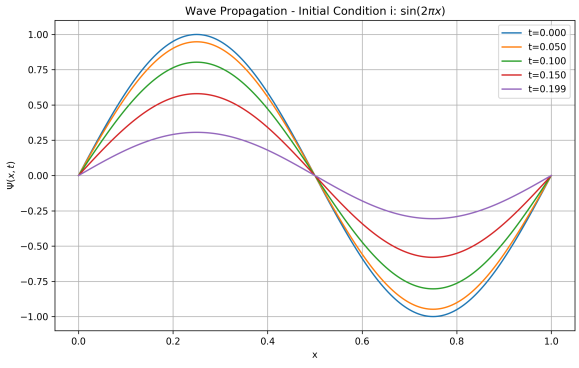


Fig. 1: Movement of the string captured at $t = 0, 0.05, 0.1, 0.15, 0.199$, with $\Delta t = 0.001$, at initial condition $\Psi(x, t = 0) = sin(2\pi x)$

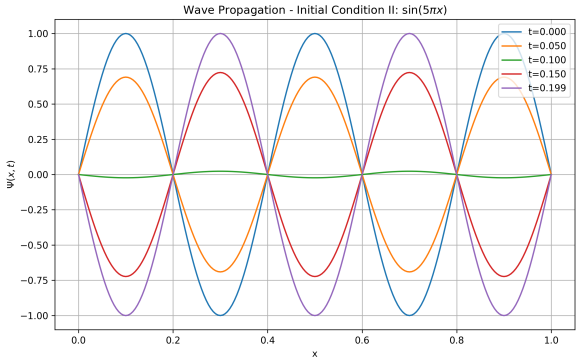


Fig. 2: Movement of the string captured at $t = 0, 0.05, 0.1, 0.15, 0.199$, with $\Delta t = 0.001$, at initial condition $\Psi(x, t = 0) = sin(5\pi x)$

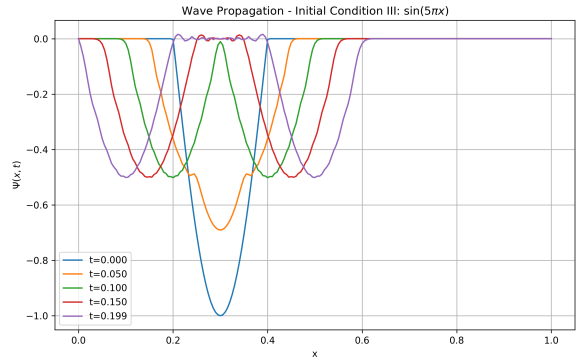


Fig. 3: Movement of the string captured at $t = 0, 0.05, 0.1, 0.15, 0.199$, with $\Delta t = 0.001$, at initial condition $\Psi(x, t = 0) = sin(5\pi x)$ if $\frac{1}{5} < x < \frac{2}{5}$, else $\Psi = 0$

B. Time-Dependent Diffusion

To simulate the time-dependent diffusion equation, equation 15 was used to calculate the value of every lattice point after a time step of δt. For the simulation values of $D = 1$, $\Delta x = 0.05$ and $\Delta t = 0.0001$ were used. First, the correctness of the simulation was tested, where for different points in time the concentration along the y axis $(c(y))$ was measured and compared to the analytical solution from equation 17 as can be seen in figure 4. Here it is clear to see that the values of the model are very close to the analytical solution meaning that the correctness of the model is very high.

Figure 5 shows the concentration of the model for various time steps, here the lighter color represents a high concentration and the darker colors a lower concentration. As can be seen over time the concentration of the simulation becomes more linear as the systems reaches a state of equilibrium.

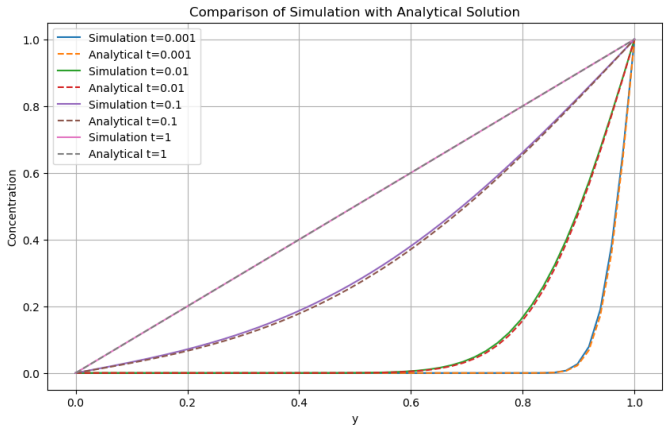
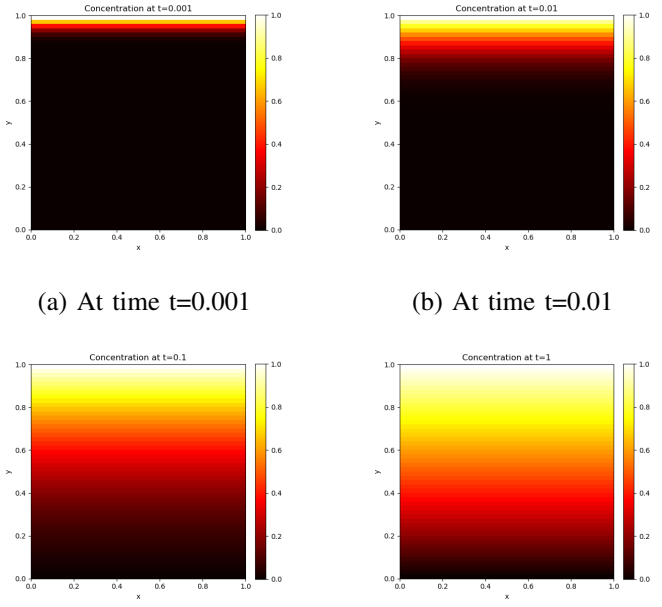


Fig. 4: Graph showing the concentration along the y axis for time-dependent diffusion model with values $D = 1$, $\delta x = 0.05$ and $\delta t = 0.0001$ compared to the analytical solution.



(a) At time t=0.001 (b) At time t=0.01

(c) At time t=0.1 (d) At time t=1

Fig. 5: A visualisation for the concentration c(y) of the time-independent diffusion model with parameters $D = 1$, $\delta x = 0.05$ and $\delta t = 0.0001$ for different values of time.

C. Time-Independent Diffusion

For the time-independent diffusion. The system is no longer dependent on time and has reached a steady state. Each iterative solver was implemented and it was counted how many iterations it took them to converge as stated in eq.(20). The numerical result was also compared to the analytical result which is

$$\lim_{t\to\infty} c(y,t) = y, \tag{23}$$

To do this the root-mean-square-error method was implemented, this being

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{24}$$

Here, the observed value (i.e., y_i) is the result our iterative solvers produced, and the predicted value (i.e., $\hat{y}_i$) is the result the analytical solution produced. The results are:

- For *Jacobi Method* it took the solver 4068 iterations to converge in a 50x50 grid. The RMSE was 0.0068
- For *Gauss-Seidel Method* it took the solver 2385 iterations to converge in a 50x50 grid. The RMSE was 0.0034.
- For *SOR Method* it took the solver 403 iterations to converge in a 50x50 grid, with $\omega = 1.8$. The RMSE was 0.0003.

The *SOR Method* has the lowest RMSE value and converges the fastest, as predicted. The *Gauss-Seidel Method* is an upgrade to the *Jacobi Method* with shorter iteration taken to converge and higher accuracy. But out of all the solvers the *SOR* solver is the most efficient and the most accurate one.

This is also evident in fig.(6), convergence measure δ vs number of iteration k

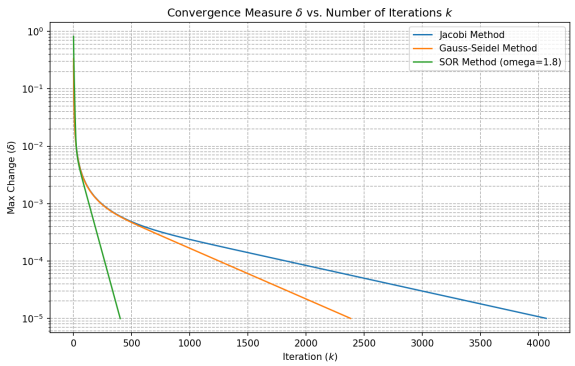


Fig. 6: Iterations that take different solvers to reach convergence. The SOR method reaches convergence with the lowest iteration count

The SOR method has something special compared to the other two methods, it has a unique factor called the relaxation factor, ω, which can be used to accelerate the convergence of the iterator or sometimes dampen it.

When ω is lower than 1, it makes the solver harder to converge, when it is equal to 1, the solver becomes the *Gauss-Seidel* solver. But when $\omega > 1$, the relaxation factor accelerates the solver, making it easier to converge.

For a 50x50 grid, its found out that the most optimal ω (the ω that makes the solver converge the fastest) is $\omega = 1.88$. To find the relation between ω and N, different Ns are tested with ωs ranging between $1.7 < \omega < 2.0$. The fig.(7) shows the relationship between the optimal $omega$ and grid size N.

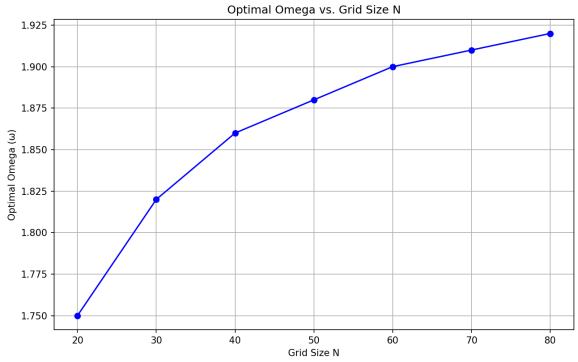


Fig. 7: Optimal ω grows as grid size N increases

V. Discussion/Conclusion

When looking at the time-dependent diffusion equation it can be said that an accurate model has been made with a high level of correctness as when looking at figure 4 the results are very close to each other.

Furthermore when comparing the different methods used for iteration for the time-independent diffusion model, it is clear that the Successive Over Relaxation method is the best and most efficient method as with a relaxation parameter of $\omega = 1.8$ it converges far faster than the others and has the smallest RMSE. And its optimal ω grows as the grid size N grows, at the same it might saturate after certain value.