Scientific Computing Exercise Set 1

Gileesa McCormack (13965751) & Alara Karadeniz (16384806) & Sooriyaa Karunaharan (16422805)

***Repository**: https://github.com/Gileesa/ScientificComputing*

I. Introduction

SCIENTIFIC computing is one of the most powerful tools in the construction and analysis of mathematical models. By implementing mathematical models on computers, we can not only understand them better, but also solve problems that are difficult to solve analytically.

In this report, our aim is to investigate the limits of numerical methods commonly used in solving two well-known partial differential equations (PDEs): the wave equation and the diffusion equation. More explicitly, we will use a finite-difference model to solve the one-dimensional wave equation. For the diffusion equation, we employ a finite-difference discretization method combined with iterative solvers, as well as the Jacobi method, the Gauss–Seidel method, and the Successive Over-Relaxation (SOR) method. Through the implementation of these simple yet powerful computational models, we aim to better understand their behavior, as well as their strengths and limitations.

II. Theory

A. One-Dimensional Wave Equation

A commonly used partial differential equation is the one-dimensional wave equation. It has many applications, such as describing a current in a lossless transmission line or the propagation of optical waves [5]. The one-dimensional wave equation is as follows:

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

In this equation, $\Psi(x,t)$ represents the vibration amplitude of the wave at a specific spatial and temporal position. The constant c denotes the wave's propagation speed.

1) Finite-Difference Discretization: The one-dimensional wave equation as described in Equation 1 can be solved numerically by implementing a uniform discretization of the spatial and temporal domains [5]. We divide the spatial domain into N_x intervals of size Δx, where each $x_i = i\Delta x\ ;\ i \in \mathbb{N}$ is one point on the grid. We do the same with the temporal domain, dividing it into N_t intervals of size Δt with $t_j = j\Delta t\ ;\ j \in \mathbb{N}$. We can then write the combination of a specific spatial and temporal point as $\Psi(x = i\Delta x, t = j\Delta t) \equiv \Psi_i^j$.

In addition, we need to approximate the partial differential equation using finite difference expressions. We can use both the forward and backward Taylor expansions to find the second-order central difference in time and space [5]. We can rearrange this to obtain the following approximation:

$$\Psi_i^{j+1} = c^2\frac{(\Delta t)^2}{(\Delta x)^2}(\Psi_{i+1}^j+\Psi_{i-1}^j-2\Psi_i^j)-\Psi_i^{j-1}+2\Psi_i^j \tag{2}$$

This finite difference discretization can be used to find a numerical solution to the one-dimensional wave equation if initial and boundary conditions are provided.

B. The Time-Dependent Diffusion Equation

The two-dimensional time-dependent diffusion equation describes diffusion over a two-dimensional spatial domain and is given by [5]:

$$\frac{\partial c(x,y,t)}{\partial t} = D\nabla^2 c(x,y,t) \tag{3}$$

Here, $c(x,y,t)$ represents the concentration as a function of the spatial coordinates x and y and time t. In addition, the ∇ symbol represents the Laplacian operator and D is the diffusion constant. Several numerical methods can be used to solve this partial differential equation.

1) Finite-Difference Discretization: To solve the two-dimensional time-dependent diffusion equation, we can discretise both the two dimensional spatial domain and the temporal domain [5]. We can divide each spatial axis into N intervals of length δx. In addition, we divide the time domain into N_t intervals of length δt. For each point in this grid, we can assign some value $i\delta x$ to refer to its x-position, where $i \in \mathbb{N}$. Similarly, we assign some $j\delta y,\ j\ \in \mathbb{N}$ and $k\delta t,\ k\ \in \mathbb{N}$ to refer to its y-position and t-value. We can then define a single point on the grid at a specific time value as follows:

$$c(i\delta x, j\delta y, k\delta t) \equiv c_{i,j}^k \tag{4}$$

A common explicit finite difference scheme is defined as follows:

$$c_{i,j}^{k+1} = c_{i,j}^{k} + \frac{\delta t D}{\delta x^2}(c_{i+1,j}^{k} + c_{i-1,j}^{k} + c_{i,j+1}^{k} + c_{i,j-1}^{k} - 4c_{i,j}^{k}) \tag{5}$$

This equation is stable if the diffusion coefficient $\alpha = \frac{\delta t D}{\delta x^2} \leq \frac{1}{4}$.

C. The Time-Independent Diffusion Equation

Now we will look at the time development of the concentration profile when it comes to the steady-state and how it reaches it. We can do this by removing the time variable from Equation 5 and making sure $\Delta^2 c = 0$ so that we can get the finite difference equation [5]:

$$c_{i,j} = \frac{1}{4}(c_{i+1,j} + c_{i-1,j} + c_{i,j+1} + c_{i,j-1}) \tag{6}$$

Since time is no longer relevant, this equation does not include the superscript k any longer.

There are multiple numerical methods to solve the time-dependent diffusion equation. In the next subsections we will be discussing three iterative methods.

1) Jacobi Method: The Jacobi method is denoted with the equation [5]:

$$c_{i,j}^{k+1} = \frac{1}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k} + c_{i,j+1}^{k} + c_{i,j-1}^{k}) \tag{7}$$

Where k denotes the k-th iteration of the method. This equation is quite similar to the time-dependent equation except for the fact that time is not accounted for. Therefore, there is a stopping condition where a solution is assumed to have converged for all (i, j) if:

$$\delta \equiv \max_{i,j} \left| c_{i,j}^{k+1} - c_{i,j}^{k} \right| < \varepsilon \tag{8}$$

Where $\varepsilon \ll 1$.

2) Gauss-Seidel Method: Guass-Siedel method offers an improvement in terms of convergence time over the Jacobi iteration, where a new value is used as soon as it has been assigned [5]. Assuming that the iteration goes along the rows this method is denoted by:

$$c_{i,j}^{k+1} = \frac{1}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}) \tag{9}$$

Although the Gauss-Siedel method provides a faster convergence time over the Jacobi method, it does not offer a great improvement.

3) Succesive Over-Relaxation Method: A great improvement in terms of convergence time over both Gauss-Siedel and Jacobi iteration is the Successive Over Relaxation Method (SOR) [5]. This formula is achieved by adding an over-correction (ω) into the iterate:

$$c_{i,j}^{k+1} = \frac{\omega}{4}(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}) + (1-\omega)c_{i,j}^{k} \tag{10}$$

This method shows convergent behavior only for $0 < \omega < 2$ [5].

III. METHOD

1) Wave Equation: Finite-Difference Discretization: To find a numerical solution to the one-dimensional wave equation as described in Equation 1, the boundary conditions must first be defined. The spatial domain is set to size $0 \leq x \leq L$, with $L = 1$. The boundary conditions used in this research are fixed boundaries: $\Psi(x = 0, t) = \Psi(x = L, t) = 0$.

To initialize the wave profile, we use a Taylor expansion to set the first time step. This is necessary because we need two time steps for the explicit finite-difference propagation scheme. We assume that the initial velocity of wave propagation is zero. The process is described in Algorithm 1.

Algorithm 1 Initialization of First Time Step

```
1: Input: Initial displacement $u_i^0$, grid size $N_x$, constant $r = c\frac{(\Delta t)}{(\Delta x)}$
2: Output: Updated array $u_i^1$
3: for $i = 1$ to $N_x - 1$ do
4:     $u_i^1 \leftarrow u_i^0 + \frac{r^2}{2}\left(u_{i+1}^0 - 2u_i^0 + u_{i-1}^0\right)$
5: end for
6: $u_0^1 \leftarrow 0$                    ▷ Left boundary
7: $u_{N_x}^1 \leftarrow 0$                ▷ Right boundary
8: return $u^1$
```

Next, we must update the wave over time. This is essentially done using an explicit second-order central difference scheme (leapfrog scheme), in which we use two previous time steps to approximate the next wave in time. This method provides a greater accuracy compared to first-order methods such as forward Euler. Additionally, it is straightforward to implement because it is an explicit scheme. We applied this in Algorithm 2.

We applied our algorithms to three different wave scenarios: $\Psi = \sin(2\pi x)$, $\Psi = \sin(5\pi x)$ and $\Psi = \sin(5\pi x)$ if $\frac{1}{5} < x < \frac{2}{5}$ else $\Psi = 0$. All code was written in the Python programming language, using the packages `numpy` and `matplotlib`. We applied a time step of size $\Delta t = 0.001$ and ran it for 1000 steps, whereas we

Algorithm 2 Explicit Finite-Difference Propagation of 1D Wave Equation

```
 1: Input: Initial data $u_i^0$, $u_i^1$, grid size $N_x$, number of time steps $N_t$, constant $r = c\frac{(\Delta t)}{(\Delta x)}$
 2: Output: Solution $u_i^j$ for $j = 0, \ldots, N_t$
 3: for $j = 1$ to $N_t - 1$ do
 4:     for $i = 1$ to $N_x - 1$ do
 5:         $u_i^{j+1} \leftarrow 2u_i^j - u_i^{j-1} + r^2(u_{i+1}^j - 2u_i^j + u_{i-1}^j)$
 6:     end for
 7:     $u_0^{j+1} \leftarrow 0$                    ▷ Left boundary
 8:     $u_{N_x}^{j+1} \leftarrow 0$                 ▷ Right boundary
 9: end for
10: return $u$
```

used 1000 spatial steps of size $\Delta x = 0.001$. In addition, the propagation speed was set to $c = 1$.

2) Diffusion Equation: Finite-Difference Discretization: In the computational implementation of the finite-difference scheme for the two-dimensional time-dependent diffusion equation, the spatial domain is defined as $0 \leq x, y \leq 1$. The spatial domain is discretized into $N_x = 50$ grid points in the x-directions and $N_y = 50$ grid points in the y-direction, with indices $i = 0, \ldots, N_x - 1$ and $j = 0, \ldots, N_y - 1$.

The explicit five-point stencil in Equation (5) is applied at stencil points $0 \leq i \leq N_x - 1$, $1 \leq j \leq N_y - 2$. In addition, the equations at the boundaries of the domain must be specified. The boundaries on the x-axis are periodic, such that the leftmost and rightmost points wrap around: $c_{0,j} = c_{N_x-1,j}$ for all $j = 0, \ldots, N_y - 1$. Moreover, the boundary conditions for the y-direction are such that the bottom boundary is fixed at $c_{i,0} = 0$, and the top boundary at $c_{i,N_y-1} = 1$, for all $i = 0, \ldots, N_x - 1$. We ran the simulation for 10000 time steps of size $\Delta t = 0.0001$

The boundary conditions are explicitly:

$$c_{i,0}^{k+1} = 0, \qquad i = 0, \ldots, N_x - 1 \quad \text{(bottom)} \tag{11}$$

$$c_{i,N_y-1}^{k+1} = 1, \qquad i = 0, \ldots, N_x - 1 \quad \text{(top)} \tag{12}$$

$$c_{-1,j}^{k} = c_{N_x-1,j}^{k}, \quad j = 1, \ldots, N_y - 2 \tag{13}$$

$$c_{N_x,j}^{k} = c_{0,j}^{k}, \qquad j = 1, \ldots, N_y - 2 \quad \text{(periodic in } x\text{)} \tag{14}$$

Since the initial condition and the periodic boundary conditions are uniform in the x- direction, the solution $c(x, y, t)$ remains independent of x and depends only on y and t. We therefore validate our numerical solution by comparing $c(y, t)$ with the analytical solution given in Equation (15) for $D = 1$.

$$c(x,t) = \sum_{i=0}^{\infty} \left[\text{erfc}\left(\frac{1 - x + 2i}{2\sqrt{Dt}} \right) - \text{erfc}\left(\frac{1 + x + 2i}{2\sqrt{Dt}} \right) \right] \tag{15}$$

We ran our simulation as described in Algorithm 3.

Algorithm 3 Finite-Difference Update for 2D Diffusion Equation

```
1: Input: Current matrix $C^k$, diffusion coefficient $\alpha$, grid size $N_x \times N_y$
2: Output: Updated matrix $C^{k+1}$
3: for $j = 1$ to $N_y - 2$ do
4:     for $i = 0$ to $N_x - 1$ do
5:         $C_{i,j}^{k+1} \leftarrow C_{i,j}^k + \alpha\Big(C_{i+1,j}^k + C_{i-1,j}^k + C_{i,j+1}^k + C_{i,j-1}^k - 4C_{i,j}^k\Big)$    ▷ Periodic boundaries in $x$-direction (implemented via modulo in code)
6:     end for
7: end for
8: $C_{i,0}^{k+1} \leftarrow 0 \ \forall i$          ▷ $y = 0$ boundary condition
9: $C_{i,N_y-1}^{k+1} \leftarrow 1 \ \forall i$      ▷ $y = 1$ boundary condition
```

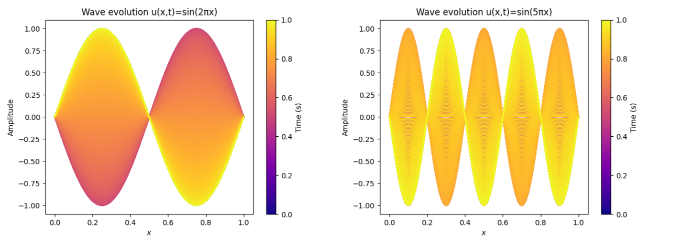
3) Implementation of Objects for the Time Independent Diffusion: For our experiment, we placed two different types of objects of size 2x1 randomly into the domain to see how the SOR iterative method is affected by them. One of the types of objects we added are sink objects. To implement this, we created a secondary matrix initialized to zero, in which sink object locations were marked with a value of 1. We made sure to keep cells that are occupied by a sink object at a concentration of 0.

The second type of object we added is the insulating object, which is denoted by value 2 in the secondary matrix. For this object, we kept the concentration of the cell where the object is placed constant. We also made sure if the neighbor of a cell is an insulating object then we use the value of the current cell for the neighbor calculation. Since we have placed the objects randomly, we made sure to run with the same number of objects multiple times to get average values.

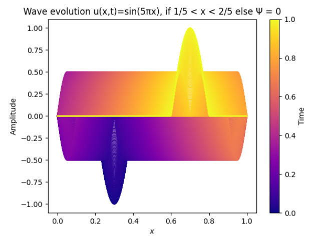
IV. Results

A. Wave Equation

We created a plot of the wave amplitude over the spatial coordinate x for three different initial conditions. Each plot depicts multiple time steps with each line representing a specific time. The time values of the waves are indicated by their color. Some time steps may be hidden because later waves overlap them. Rhe full wave propagation for each plot can be found in the Github repository [4].



(a) (b)



(c)

Fig. 1: Numerical solutions of the 1D wave equation for different initial conditions. a) $\sin(2\pi x)$ b)$\sin(5\pi x)$ c) $\sin(5\pi x)$ in $\frac{1}{5} < x < \frac{2}{5}$, else 0. The x-axis shows the spatial coordinate, the y-axis shows the amplitude, and the color of the lines denotes time.

B. Comparison between the Analytical Solution and Numerical Solution

Figure 2 shows that the numerical solution matches the analytical solution at $t = 0$, 0.001, 0.01, 0.1, and 1 for $D = 1$. The discretized profiles reflect the boundary conditions ($c = 0$ at $y = 0$ and $c = 1$ at $y = 1$), initial condition, and x-periodicity, approaching the expected steady-state $c(y, \infty) = y$ at $t = 1.0$.

C. Diffusion in the 2D Domain

We plotted the concentration field $c(x, y, t)$ at $t = 0$, 0.001, 0.01, 0.1, and 1 (see Figure 3). At $t = 0$, the concentration is close to one at the top, and the rest of the grid has the concentration value 0, indicating that diffusion is highly concentrated. As time increases, the concentration spreads downwards and the spread becomes smoother. By $t = 1$, the concentration forms a smooth gradient between the lower and upper boundaries, consistent with the expected steady-state behavior. An animation of the diffusion process is provided in the project repository [3].

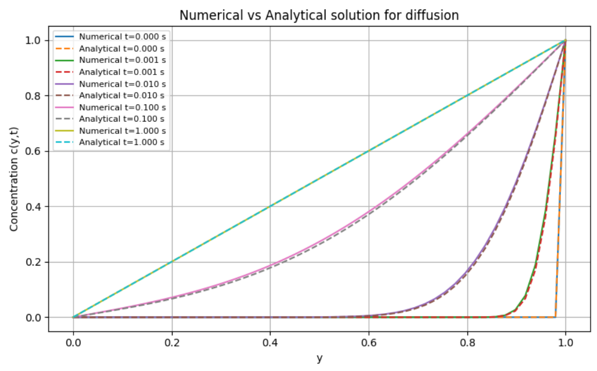
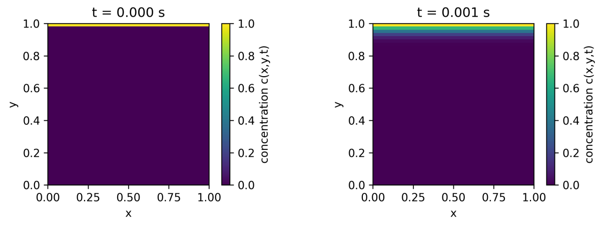
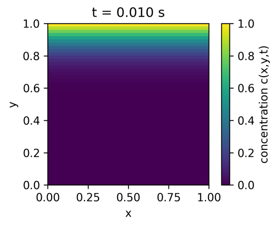


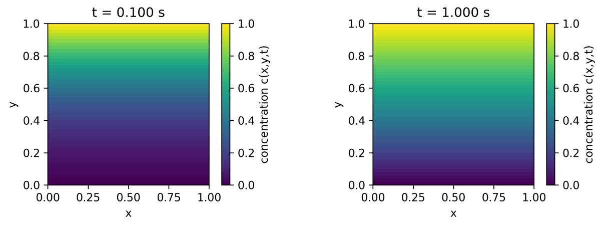
Fig. 2: Comparison between the numerical solution (solid line) and the analytical solution (dashed line) for the concentration, $c(y, t)$ at $t = 0$, 0.001, 0.01, 0.1, and 1 respectively for $D = 1$.



(a) $t = 0$ (b) $t = 0.001$



(c) $t = 0.01$

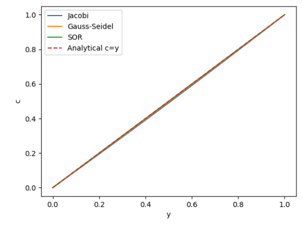


(d) $t = 0.1$ (e) $t = 1$

Fig. 3: Spatial distribution of the concentration field $c(x, y, t)$ at selected times, where color scale denotes the concentration value.

D. Time Independent Diffusion

For our first plot we plotted all the three different iterative methods together to see the linear dependence of the concentration on the y-value which can be seen in Figure 4a. We also plotted the analytical solution and found the error values through comparing the analytical solution against the experimental one which can be seen in the table I. Then we have plotted the δ (convergence measure) against the iteration count, with ω values $[1.75, 1.8, 1.85, 1.9, 1.95]$ for SOR, for all three iterative methods, which can be seen in 4b.


(a) Comparison between Jacobi, Gauss-Seidel, and SOR methods against the analytical solutions

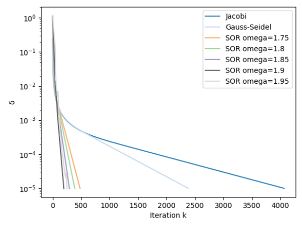
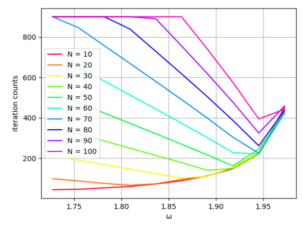

(b) Convergence comparison between Jacobi, Gauss-Seidel, and SOR methods

Fig. 4

Jacobi	0.00972
Gauss-Seidel	0.00485
SOR	0.000364

TABLE I: Errors of Jacobi, Gauss-Seidel, and SOR methods from the analytical solutions

We then wanted to see for SOR method what the optimal ω values are and how different board sizes (N) affect this optimal value in Figure 5a. For this we made a plot with different lines representing different N values and ran our code with different ω values to see how each iteration count compares. We also did the same with board size 50x50 and adding different number of randomly placed sink objects to visualize the optimal ω values in Figure 5b. For the different


(a) Comparing different ω values against different board sizes (N) and iteration count for SOR method

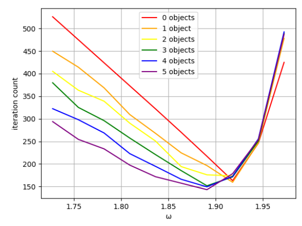
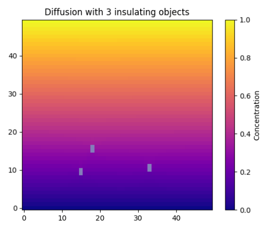
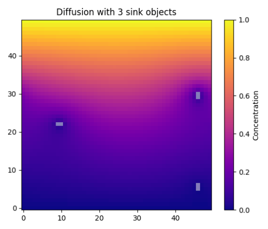

(b) Comparing different ω values against different number of sink objects and iteration count for SOR method

Fig. 5

number of sink objects we have compared iteration count against the convergence measure to see how quickly the concentration reaches steady-state, which can be seen in Figure 6c. We also decided to visualize how the concentration looks after reaching steady-state with 3 sink objects in 6b. In this image, we highlighted where the objects are placed but in our repository we visualized how the objects are acting throughout the iterations [2]. We have also done the same for 3 different insulating objects which is in Figure 6a. The gif for this can be seen in our repository through the link in the references [1].



(a) Showing the effects of 3 insulating objects in the concentration field at steady-state for SOR



(b) Showing the effects of 3 sink objects in the concentration field at steady-state for SOR

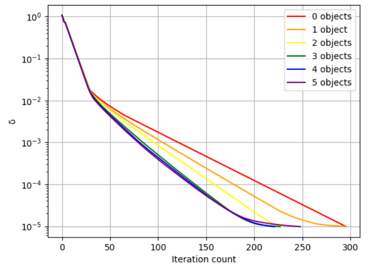

(c) Comparing the iteration count and convergence measure with different number of sink objects for SOR method

Fig. 6

V. Discussion

A. Wave Equation

The results show that the second-order central difference in time and space provides an accurate numerical solution to the one dimensional wave equation. The plots in Figure 1 show that the combination of the chosen initial conditions and time stepping captures the wave propagation accurately. This is apparent from the fact that the shape and amplitude of the wave are preserved over time and that the numerical solution closely follows that of the behavior expected from the analytical solution.

B. Comparison between the Analytical Solution and Numerical Solution

By setting $\Delta t = \frac{0.25\Delta x^2}{D}$, we obtain $\alpha = \frac{D\Delta t}{\Delta x^2} = 0.25$, guaranteeing numerical stability without oscillations.

Moreover, the modulo operator ensured seamless x-periodicity by wrapping grid indices, while fixed boundary conditions $c(i, 0) = 0$ and $c(i, N_Y - 1) = 1$ were maintained throughout the simulation.

These choices ensure a physically meaningful evolution of the concentration field and allow a direct comparison with the analytical solution by extracting the one-dimensional profile $c(y, t)$.

C. Diffusion in the 2D Domain

The results illustrate that over time the concentration gradient spreads and becomes smoother. This behavior is expected for diffusion because the concentration is transported from a high concentrated region (top boundary) toward the low concentration region (bottom boundary), which smooths out steep gradients. Over time, the concentration field approaches equilibrium between the two fixed boundaries and becomes more evenly distributed across the domain (see Figure 3e).

D. Time Independent Diffusion

When we look at our Figure 2 and our table I, we can see the our experimental results are very close to the analytical ones as we are getting very small errors and the lines are very close to each other. When we look at the Figure 4b we can see that Jacobi has the most iteration count, which was expected. Then next we see that Gauss-Siedel has less iteration count compared to Jacobi, but as can be seen converges much later than SOR method. As expected SOR converges the quickest out of all three. When we look at our Figure 5a, we can see that as the board size increases the optimal ω value becomes higher, this could be due to the fact that since there are more cells then we need to have a higher correction count in between the iteration, which is the point of adding the ω value into the equation. We can also see that the iteration counts become higher, which is expected as there are more cells to cover. The reason as we approach $\omega = 2$ for all the lines could be explained by the fact that as we are approaching this value the method stops being able to converge since it only converges for values $0 < \omega < 2$ [5]. When we look at Figure 5b we can see that as we add more sink objects the optimal ω becomes also smaller and the diffusion converges more quickly. This is also evident in Figure 6c, where the higher object count converges earlier. This could be explained by the fact that sink objects pull the concentration into the themselves, leading to the concentration going to the steady-state much quicker. In Figure 6a and the gif the Github respository, we can see that when we add in insulating objects, these objects just stay stable and do not affect the area around them unlike the sink objects[1]. They stay in concentration 0 from beginning to end (which is the value they start out as) and reflect the diffusion around them.

VI. Conclusion

Throughout this report, we have explored several concepts. We have investigated the one-dimensional wave equation and the time-dependent diffusion equation. we have looked into how the finite-difference model works and have demonstrated visually by looking into different wave equations with boundaries to see that through the initial conditions and time stepping we can accurately capture the propagation of the waves. Through looking at diffusion over a 2D domain, we have explored a concentration gradient and visualized the behavior of the concentration and have seen that as it reaches equilibrium how the gradient smooths out. Then we looked into iterative methods such as Jacobi, Gauss-Seidel and Successive Over-Relaxation methods for time independent diffusion. Through this exploration, we have seen that Successive Over-Relaxation is a method that runs and converges faster. We have also explored adding sinks and insulating objects into the field and have seen the effects of it. Through the sink objects, we have seen that the iteration converges faster towards equilibrium when we have more objects. In the future, it would be a good research path to explore the effects of adding more insulating objects on the convergence and the optimal ω value.

References

[1] Gileesa McCormack. *3 insulating objects gif.* URL: https://github.com/Gileesa/ScientificComputing/blob/main/Figures/1.3/insulating_3_obj.gif.

[2] Gileesa McCormack. *3 sink objects gif.* URL: https://github.com/Gileesa/ScientificComputing/blob/main/Figures/1.3/sink_3_obj.gif.

[3] Gileesa McCormack. *Diffusion over Time gif.* URL: https://github.com/Gileesa/ScientificComputing/blob/main/Figures/1.2/diffusion_over_time.gif.

[4] Gileesa McCormack. *Wave Figures Folder.* URL: https://github.com/Gileesa/ScientificComputing/tree/main/Figures/1.1.

[5] Gábor Závodszky. *Assignments for Scientific Computing.* 2026. URL: https://canvas.uva.nl/courses/56607/pages/assignments?module_item_id=2741264 (visited on 02/22/2026).