Scientific Computing Exercise Set 1

Sander Broos (11895616) & Nick van Santen (11857846)

***Abstract*—Both the wave equation and the diffusion equation are differential equations. This report simulates the wave equation using a finite difference method and investigates different methods for simulating the diffusion equation. First, a finite difference equation is used that relies on the time-dependent diffusion equation. Three other methods are investigated which are time-independent and work iteratively: Jacobi iteration, Gauss-Seidel iteration, and Successive Over Relaxation (SOR). These methods are compared to the analytical solution of a specific configuration of the diffusion field. All four methods converge to the analytical solution for $t \to \infty$, although SOR generally has the lowest error and converges in fewer iterations than Gauss-Seidel, with Jacobi taking the most iterations to converge. However, the speed of the convergence of SOR depends significantly on a parameter ω, with the optimal ω varying for different resolutions of the simulation. Lastly, objects are added to the simulation which act as sinks in the diffusion field. These result in lower values of the optimal ω and faster convergence, but further research is suggested to investigate if this holds for other configurations of objects.**

I. Introduction

DIFFERENTIAL equations appear everywhere once you start looking for them. They can describe the movement of liquids, the transfer of heat, describe the evolution of quantum-mechanical systems, etc. However, most of these problems do not have an analytical solution. These problems can only be numerically approximated. One of the numeric approximation methods is the finite difference method. This method tries to approximate the solutions with the help of finite difference equations. These equations approximate the derivative of the differential equations. By iteratively applying the finite difference equations the solution can be approximated.

This report will apply finite difference equations on a 1D wave equation and a 2D time-dependent diffusion equation. For the diffusion equation, it will look into three different iteration methods: Jacobi iteration, Gauss-Seidel iteration, and Successive Over Relaxation (SOR). These methods will be compared with each other by looking into their convergence rate. Furthermore, this report will look into the effect of objects on the diffusion.

II. Theory

FINITE difference equations are used to numerically solve differential equations. This section will compose the finite difference equations for the 1D wave equation and the 2D diffusion equations. Furthermore, it will describe three methods to increase the convergence rate of these finite difference equations.

A. Finite difference equations

Finite difference equations work by discretising the derivatives of the differential equations. This is demonstrated on the 1-dimensional wave equation [1]

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

with $\psi(x,t)$ the amplitude of the wave at a position x and a time t. c is a constant related to the characteristics of the wave. First the wave function $\psi(x,t)$ is discretized to $u(i,j)$, where u is a 2D grid of values where the indices i and j refer to the amplitude at timestep i at position j. Next, the first-order spatial derivative is approximated by computing the difference between neighbouring cells and dividing by the spatial resolution Δx. Thus this gives

$$\frac{\partial \psi}{\partial x} = \frac{u(i+1,j) - u(i,j)}{\Delta x} \tag{2}$$

$$\frac{\partial \psi}{\partial x} = \frac{u(i,j) - u(i-1,j)}{\Delta x} \tag{3}$$

The second-order derivative is then computed by subtracting Equation 3 from Equation 2 and again dividing by the spatial resolution Δx. Thus the second-order derivative becomes

$$\frac{\partial^2 \psi}{\partial x^2} = \frac{u(i+1,j) + u(i-1,j) - 2u(i,j)}{(\Delta x)^2} \tag{4}$$

and similarly, the second-order time derivative can be computed

$$\frac{\partial^2 \psi}{\partial t^2} = \frac{u(i,j+1) + u(i,j-1) - 2u(i,j)}{(\Delta t)^2} \tag{5}$$

Solving Equation 1 for $u(i, j+1)$ by substituting Equations 4 and 5 gives

$$u(i,j+1) = \left(c\frac{\Delta t}{\Delta x}\right)^2 \Big[u(i+1,j) + u(i-1,j) \\ -2u(i,j)\Big] - u(i,j-1) + 2u(i,j) \tag{6}$$

This is the finite difference equation of the 1D wave equation. This equation calculates the amplitude of the wave at the next time step and thus makes it possible to compute the time evolution of the 1-dimensional wave.

This approach can also be used to compute the 2D time-dependent diffusion equation

$$\frac{\partial C}{\partial t} = D\nabla^2 C \tag{7}$$

where $C(x, y; t)$ is the concentration at coordinates x and y at a time t. D is the diffusion constant. After again discretizing the equation by defining $c(i, j; k) = C(i\Delta x, j\Delta x; k\Delta t)$, the finite difference equation for Equation 7 becomes

$$c(i,j;k+1) = c(i,j;k) + D\frac{\Delta t}{\Delta x^2}\Big[c(i+1,j;k) + \\ c(i-1,j;k) + c(i,j+1;k) + \\ c(i,j-1;k) - 4c(i,j;k)\Big] \tag{8}$$

where i and j refer to the x and y coordinate and k refers to the time. This method is stable for $\frac{4\Delta tD}{\Delta x^2} \leq 1$. Thus, the time step Δt can not be too large, resulting in a simulation that takes more iterations to compute.

B. Different iteration methods

Instead of Equation 7, there is also a time-*independent* diffusion equation. This could be a helpful equation if the time development of the simulation is not relevant and the steady-state of the simulation is the only wanted result. Since the equation does not depend on time, there is no issue of choosing a Δt for which the equation is stable, like for the method in Equation 8. Here, the goal is to solve

$$\nabla^2 c = 0 \tag{9}$$

for which three different methods are tested in this report: Jacobi iteration [2, p.469], Gauss-Seidel iteration [2, p.470], and Successive Over Relaxation [2, p.470].

1) Jacobi iteration: In the Jacobi iteration for the time independent diffusion equation, the concentration of the point at coordinates i and j after $k+1$ iterations is simply given by the average of the concentration of its four neighbours:

$$c(i,j;k+1) = \frac{1}{4}\Big[c(i+1,j;k) + c(i-1,j;k) + \\ c(i,j+1;k) + c(i,j-1;k)\Big] \tag{10}$$

This formula is equal to Equation 8 when the $\frac{4\Delta tD}{\Delta x^2} = 1$, which is the highest value of Δt for which the equation is stable. As the algorithm updates concentrations at increasing coordinates i and j, it can also need the concentrations of points that have already been updated and thus already have their value at iteration $k+1$, even though their value at iteration k is needed. Thus, the algorithm must save both the old and new values of all the coordinates at every time step.

The algorithm stops when it has come very close to convergence and the concentrations change extremely little between iterations. This happens when $\delta = \max|c(i,j;k+1) - c(i,j;k)| < \epsilon$ - when the biggest change in a concentration value of any coordinate (δ) is smaller than a previously chosen threshold (ϵ).

2) Gauss-Seidel iteration: The Gauss-Seidel iteration is the same as the Jacobi iteration with one important difference - the old concentration values are not saved and instead the updating calculation uses the most recent value whether that is from the current or the previous iteration. This results in the equation

$$c(i,j;k+1) = \frac{1}{4}\Big[c(i+1,j;k) + c(i-1,j;k+1) + \\ c(i,j+1;k) + c(i,j-1;k+1)\Big] \tag{11}$$

with the same stopping condition as the Jacobi iteration.

3) Successive Over Relaxation: The Successive Over Relaxation (SOR) algorithm uses the Gauss-Seidel iteration as its base. However, here the updating calculation "over-corrects" its difference using the old value of the concentration, with a parameter ω:

$$c(i,j;k+1) = \frac{\omega}{4}\Big[c(i+1,j;k) + c(i-1,j;k+1) \\ + c(i,j+1;k) + c(i,j-1;k+1)\Big] \\ + (1-\omega)c(i,j;k) \tag{12}$$

When $\omega = 1$, this method is equal to the Gauss-Seidel iteration. For $0 < \omega < 2$, this method is stable [2, p. 470]. The question is for which value of ω the method converges the fastest.

III. Method

THIS section will describe the implementations of the finite difference equations for the 1D wave equation and the 2D time-dependent diffusion equation.

A. Vibrating string

The 1-dimensional wave equation describes the evolution of a 1-dimensional wave. This report looks into simulating a uniform string. The evolution of the string is described by Equation 6. The rope is fixed at both ends, such that $u(0, j) = u(N - 1, j) = 0$, where N is the number of discrete spatial steps. The rope has a length of 1, so the size of the discrete spatial steps is $\Delta x = 1/N$. The rope has no initial velocity, therefore $u(i, 0) = u(i, 1)$. Three different initial configurations are tested.

i $u(i, 0) = \sin 2\pi i \Delta x$
ii $u(i, 0) = \sin 5\pi i \Delta x$
iii $u(i, 0) = \sin 5\pi i \Delta x$ if $1/5 < i\Delta x < 2/5$ else 0

B. Diffusion field

This report investigates the diffusion equation in a 2D square for $0 \leq x, y \leq 1$ with the boundary conditions $c(x, y = 1; t) = 1$ and $c(x, y = 0; t) = 0$. All points on the grid for $0 < y < 1$ are initialised to 0 at $t = 0$. Lastly, the bounds of the x-axis have the boundary condition $c(x = 0, y; t) = c(x = 1, y; t)$. Because of this, the boundaries use a variation of Equation 8. At $x = i = 0$, the equation needs the concentration at $i-1$, which from the periodic boundary condition is then taken from the point at $N = 1/\Delta x$, the right side of the field. For points at $i = 0$, Equation 8 can be rewritten to

$$\begin{aligned} c(0, j; k+1) = c(0, j; k) + D\frac{\Delta t}{\Delta x^2}\Big[c(1, j; k) + \\ c(N, j; k) + c(0, j+1; k) + \\ c(0, j-1; k) - 4c(0, j; k)\Big] \end{aligned} \tag{13}$$

The opposite boundary condition is true for points at $x = 1$, or $i = N$. These need the concentration at the point right of them, at an x-coordinate of $N + 1$. They use the points at $x = 0$, and Equation 8 can be rewritten to

$$\begin{aligned} c(N, j; k+1) = c(N, j; k) + D\frac{\Delta t}{\Delta x^2}\Big[c(0, j; k) + \\ c(N-1, j; k) + c(N, j+1; k) + \\ c(N, j-1; k) - 4c(N, j; k)\Big] \end{aligned} \tag{14}$$

Because of the symmetry from the boundary conditions at $x = 0$ and $x = 1$, the concentration does not depend on the x-coordinate and can be compared to a 1D case of the diffusion equation over the x-axis with $c(x = 0, t) = 0$ and $c(x = 1, t) = 1$. This has an analytical solution of

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

This sum theoretically increases i until infinity, but the code used for this report increases i until the term inside the sum falls below a certain threshold, which would mean the formula has reached a certain precision as the sum barely increases anymore.

For $t \to \infty$ the analytical solution becomes

$$\lim_{t\to\infty} c(y, t) = y \tag{16}$$

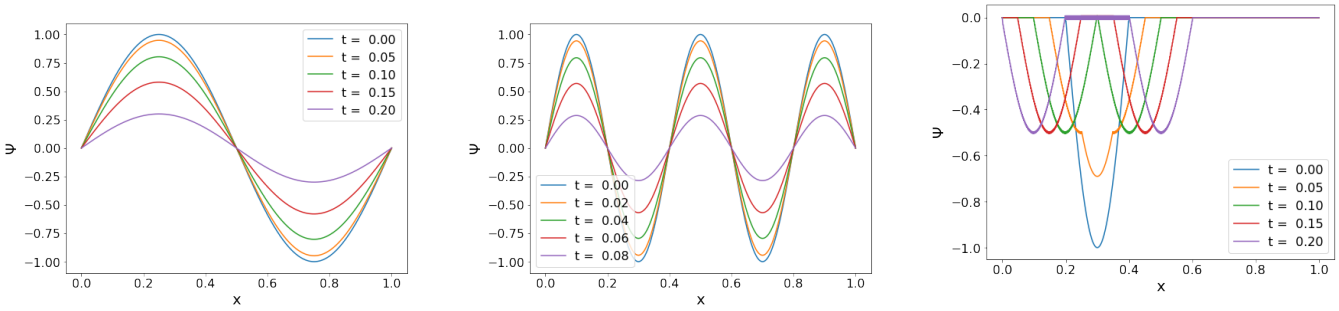
which means the concentration profile becomes a straight line.

The convergence rate of SOR depends on the chosen ω. For this diffusion field, ω should be somewhere between 1.7 and 2. The optimal ω should have the fastest convergence rate and should thus need the fewest iterations to find a solution (when $\delta < \epsilon$). Golden section search [2, p. 269] is used to find this optimal ω.

1) Objects: The diffusion field is further tested by including objects. These objects act as sinks. The cells which overlap with the object are marked as sinks. These sinks are not updated throughout the simulation and thus remain at a concentration of zero. This report will only look at one configuration of objects. There are three objects placed; 2 rectangles and 1 circle. Both rectangles have a width of 0.2 and a height of 0.05. One rectangle has its left bottom corner placed at coordinates (0.2, 0.8), while the other rectangle has coordinates of (0.4, 0.6). The circle is placed at (0.8, 0.8) and has a radius of 0.1. A visual representation of these objects and their effects can be seen in Figure 8.

IV. Results

THE results will first show the vibrating string for all three starting configurations. Thereafter, the diffusion field will be tested against the analytical solution. Jacobi iteration, Gauss-Seidel iteration, and SOR will be applied and compared against each other for their respective errors and convergence rates. At last, the optimal ω is computed for different values of N in the case with objects and without objects.



(a) $u(i,0) = \sin 2\pi i \Delta x$ (b) $u(i,0) = \sin 5\pi i \Delta x$ (c) $u(i,0) = \sin 5\pi i \Delta x$ if $1/5 < i\Delta x < 2/5$ else 0

Fig. 1: The time evolution of a vibrating string for 3 different starting conditions. All three strings used $\Delta x = 0.001$ and $\Delta t = 0.001$, and $c = 1$.

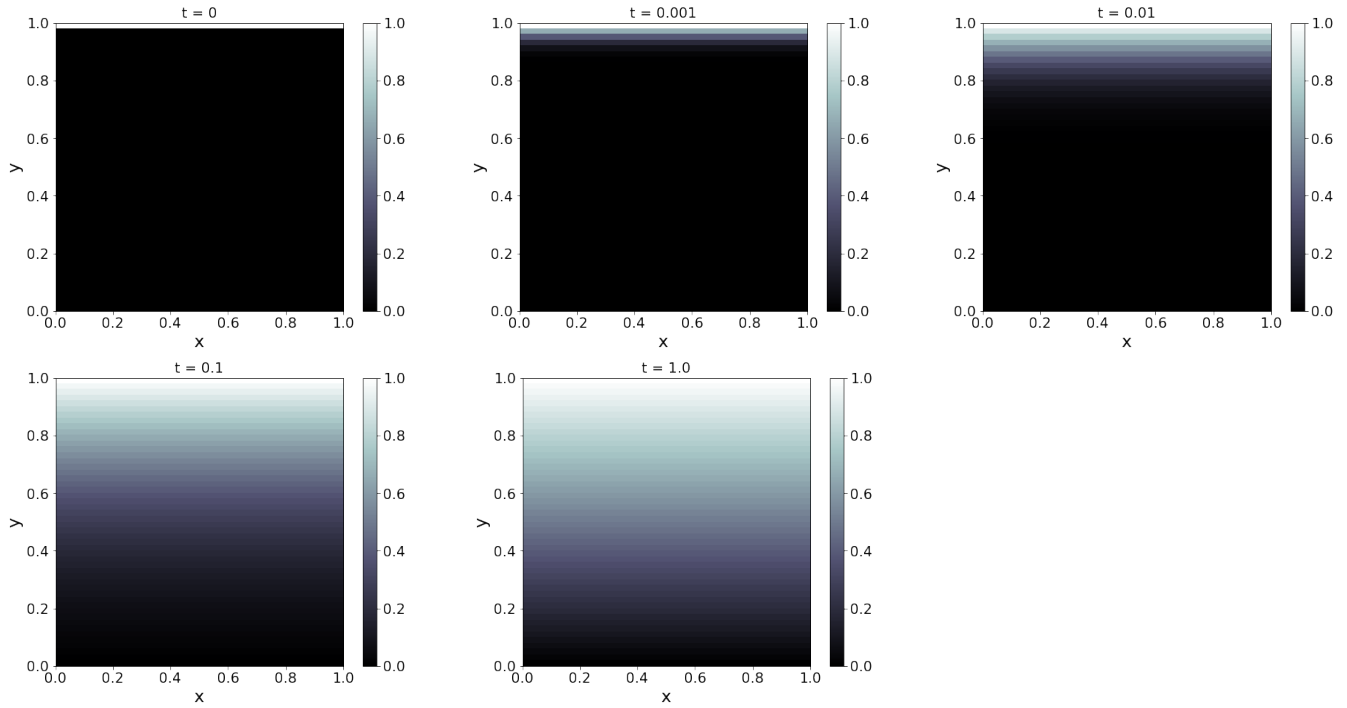


Fig. 2: The time evolution of a 2D diffusion field with boundary conditions of $c(x, y = 1; t) = 1$ and $c(x, y = 0; t) = 0$. All points on the grid for $0 < y < 1$ are initialized to 0 at $t = 0$. The bounds of the x-axis have the periodic boundary condition $c(x = 0, y; t) = c(x = 1, y; t)$. The diffusion field has parameters $N = 50$ and $D = 1$.

A. Vibrating string

The vibrating string is plotted at different time values for all three starting configurations. These can be seen in Figure 1. The vibrating strings all used $\Delta x = 0.001$, $\Delta t = 0.001$, and $c = 1$.

B. Diffusion field

The diffusion field is simulated using the finite difference method from Equation 8, plotted at different time values in Figure 2. It shows that at the start ($t = 0$) the boundary conditions are met; a concentration of 1 at $y = 1$ and the rest has a concentration of 0. Over time the concentration at the top diffuses downward.

The results of the diffusion field are compared with the analytical result of Equation 15. The computed concentration as a function of y is shown in Figure 3 for different time values. For this plot, the value at a certain y-coordinate is calculated by taking the average over all the x-coordinates at that value of y, although this finite difference method is symmetrical so all those values are equal (for some other methods discussed later this is not true anymore so the average becomes relevant there). At first there is only concentration at $y = 1$. This spreads

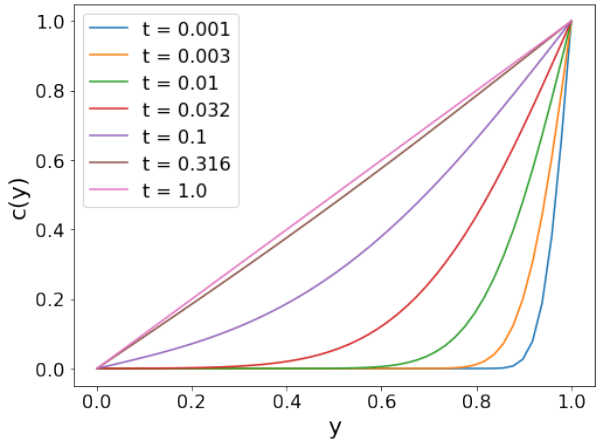


Fig. 3: The average concentration across the y-axis for different time points, using the standard finite difference simulation. The diffusion field has parameters $N = 50$ and $D = 1$.

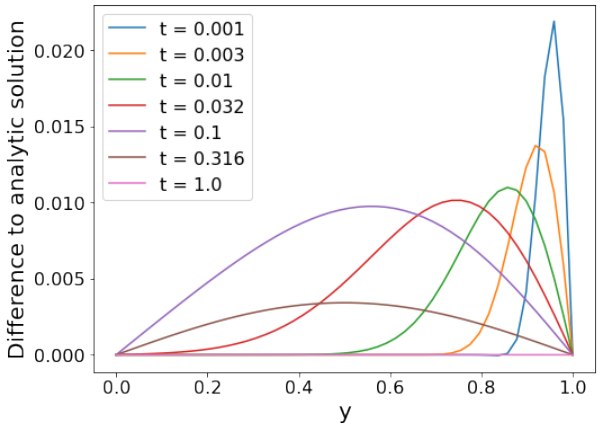


Fig. 4: The difference across the y-axis between the analytical 1D solution and the standard finite difference model, for different time points. The diffusion field has parameters $N = 50$ and $D = 1$.

to lower y over time. At $t = 1$ the concentration profile is simply (indistinguishable from) a straight line. The error between the analytical solution and the computed results is shown in Figure 4. The analytical solution was calculated using Equation 15 with a precision of 10^{-8}. No error is observed at $y = 0$ and $y = 1$, due to the fixed concentration at the boundaries. The error is initially the largest at y slightly lower than 1. Over time the error diminishes.

Each of the three iteration methods is tested at convergence, and compared against the analytical solution which is $c(y) = y$ at the stable state, from Equation 16. The results for the Jacobi iteration, Gauss-Seidel iteration, and SOR are shown in Figure 5. When convergence is reached no immediate visible difference is seen.

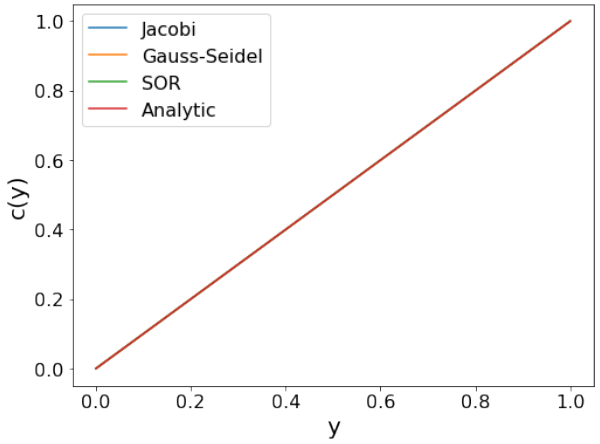


Fig. 5: The average concentration across the y-axis after converging, using different iterative methods and the analytical solution. The diffusion field has parameters $N = 50$ and $D = 1$, with the SOR method using $\omega = 1.85$. The iterative methods used a stopping parameter $\epsilon = 10^{-8}$.

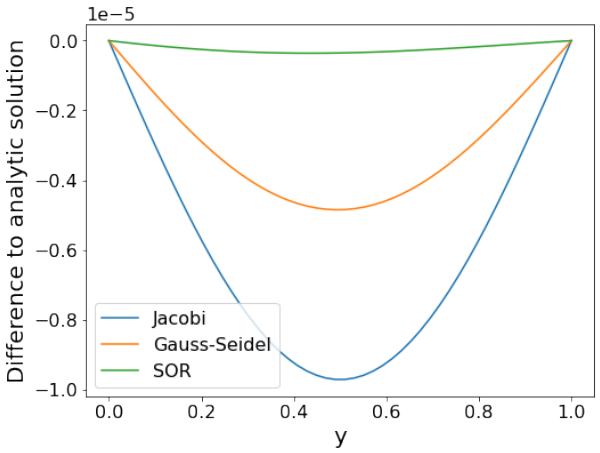


Fig. 6: The difference between the analytical formula and the average concentration across the y-axis after converging using different iterative methods. The diffusion field has parameters $N = 50$ and $D = 1$, with the SOR method using $\omega = 1.85$. The iterative methods used a stopping parameter $\epsilon = 10^{-8}$.

The absolute error, shown in Figure 6, however, does show a difference between the three methods. SOR has the lowest difference when compared to the analytical solution, while Jacobi iteration has the largest. The error for all three iteration methods is the largest halfway between $y = 0$ and $y = 1$, since there is no error there due to the boundary conditions. Note that this difference depends on the chosen value of ϵ. When a lower ϵ is chosen, the differences to the analytical solution also become smaller for all the methods, although they still

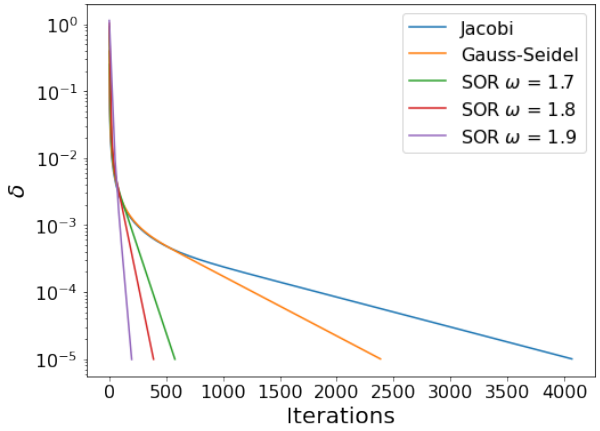


Fig. 7: The value of δ after a number of iterations of different iterative methods, with different values of ω for the SOR method. The diffusion field has parameters $N = 50$, $D = 1$ and a stopping parameter $\epsilon = 10^{-5}$.

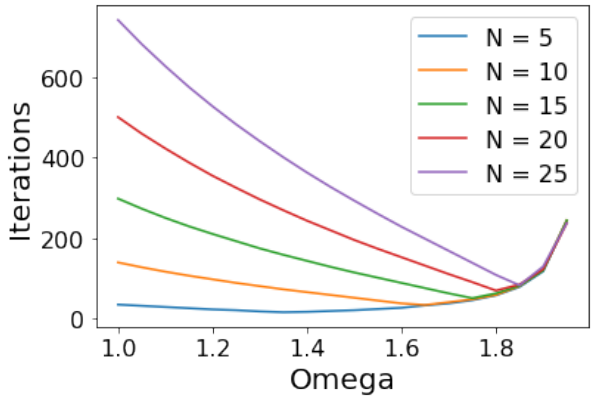


Fig. 9: The number of iterations needed before the stopping condition is met. This is performed for various N values. It shows that optimal ω is dependent on N. The diffusion field used a stopping parameter of $\epsilon = 10^{-5}$.

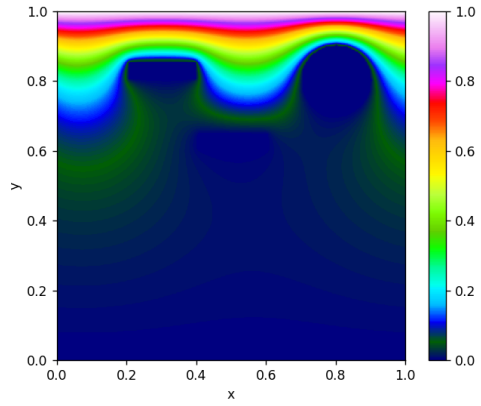


Fig. 8: The concentration of the diffusion field at the steady-state solution. 3 objects are placed in the field, which act as sinks. The top row is a source, while the bottom row is a sink.

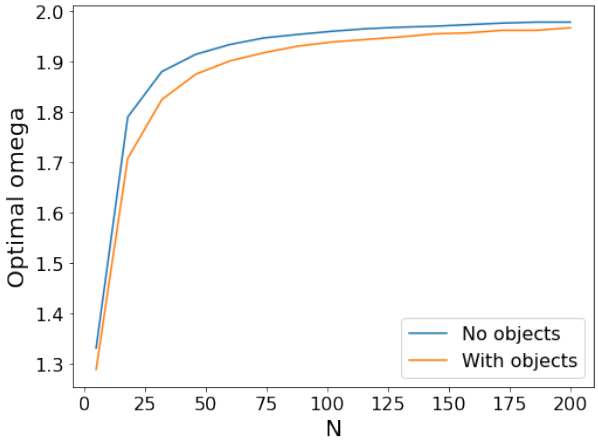


Fig. 10: The optimal ω for SOR as a function of N. The diffusion field has a stopping parameter $\epsilon = 10^{-5}$. The configuration of the objects is shown in Figure 8.

show the same order of difference (SOR having the smallest difference).

The three iteration methods each have different convergence rates. In Figure 7, the value of δ is plotted for each iteration. SOR has the fastest convergence rate in general for all three used values of ω, followed by Gauss-Seidel iteration, and then Jacobi iteration.

Figure 8 shows the steady-state solution with the objects present. Here, a different colour scale is used to more clearly show the diffusion behaviour. The colourbar on the right can be used to read the concentration values, but also represents the analytical steady-state solution *without* objects, where the concentration scales linearly with height, which makes it useful to compare. Seemingly, the diffusion spread gets damped by the presence of objects, with the concentration falling off much quicker as y approaches zero. Since the objects act as sinks their surroundings will diffuse into the sink. Thus at the barrier of the objects the concentration becomes zero. These objects then have a somewhat rippling effect at lower heights.

The convergence rate of SOR depends on ω. This can be seen in Figure 9. There is an optimal ω for which the diffusion field converges the fastest, which can be found using Golden section search. Figure 10 shows how the optimal ω depends on the resolution of the diffusion field with and without objects. It shows that the optimal ω increases when the resolution of the field increases. This is the case with and without objects, although the optimal ω is always lower in the case with objects. For large

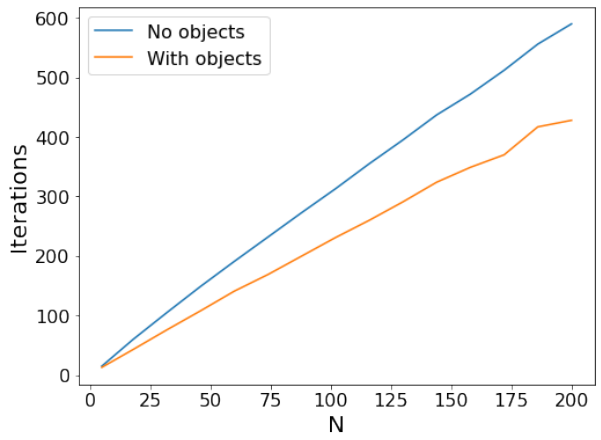


Fig. 11: The number of iterations needed when running the SOR simulation with the optimal ω before the stopping condition is met as a function of N. The diffusion field has a stopping parameter $\epsilon = 10^{-5}$. The configuration of the objects is shown in Figure 8.

N the optimal ω seems to asymptotically converge to a particular value. Figure 11 shows how many iterations were performed for the optimal ω until convergence. The number of iterations seems to scale linearly with respect to N both in the case with and without objects, although the addition of objects seems to result in faster convergence (fewer iterations needed) for all resolutions.

V. Discussion

FIGURE 5 showed that Jacobi iteration, Gauss-Seidel, and SOR converge toward the analytical solution. These methods provide a numerical method for finding the steady-state of the diffusion equation. From Figure 6 it was noted that SOR has the smallest error compared to the analytical solution from the three methods. The performance of SOR depends however on ω, this can be seen in Figure 7. This figure also shows that SOR converges faster than Jacobi, and Gauss-Seidel iteration. Figure 10 shows that the optimal ω increases when the spatial resolution increases. Including objects, the optimal ω is lower than when no objects were included. This might however be because of the specific placement configuration. Different placement configurations might have resulted in different values for the optimal ω. The number of objects, the size of the objects, and the shape of the objects could all have an influence on the optimal ω. Future research might look into what extent these attributes affect the optimal ω and the iterations needed to reach convergence.

References

[1] J. l. R. d'Alembert, "Recherches sur la courbe que forme une corde tendue mise en vibration," 1747.

[2] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed. Boston: McGraw-Hill, 2002.