University of Amsterdam

Report

Excercise Set 1

February 24, 2023

Students:
Gelieza Kötterheinrich
gelieza.kotterheinrich@student.uva.nl
14567040

Aaron De Clercq
aaron.de.clercq@student.uva.nl
14483610

Group:
2

Lecturer:
Jaap Kaandorp

Course:
Scientific Computing

Course code:
5284SCCO6Y

1 Introduction

In this paper, we will investigate numerical solutions to two of the most prominent partial differential equations: the wave equation and the diffusion equation. Both equations can be applied to describe a wide range of physical phenomena. For instance, the one-dimensional wave equation can be used to predict the transport of voltage and current along a lossless transmission line, the pressure and flow rate of a compressible liquid or gas in a pipe, sound waves in gases or liquids, and optical waves as well as the vibration of uniform strings.

First, we will examine the numerical solution to the one-dimensional wave equation. We will explore our numerical solution by simulating the vibration of a bounded string as an example for the one-dimensional wave equation. Secondly, we will discuss the numerical solution of the time dependent two-dimensional diffusion equation. We will experimentally test our numerical solution by simulating the development of a concentration profile on a two-dimensional square surface. Furthermore, we will validate our numerical results against the analytical solution. Finally, we analyze three different approaches to solve the time independent diffusion equation (*Laplace* equation): 1) the Jacobi iteration, 2) the Gauss-Seidel iteration, and, 3) Successive Over Relaxation.

The rest of this report is structured as follows: Section 2 gives a short theoretical introduction into the wave equation, the time dependent diffusion equation and the time independent diffusion equation. Next, Section 3 outlines the methods used to numerically solve these equations. The results and discussion of our experiments are presented in Section 4. Finally, Section 5 provides a conclusion of our results.

2 Theory

2.1 Vibrating string

The vibration of a uniform string can be described by the one-dimensional wave equation

$$\frac{\partial^2 \Psi}{\partial t^2} = c^2 \frac{\partial^2 \Psi}{\partial x^2}, \tag{2.1}$$

where c is a fixed non-negative real coefficient. The solution $\Psi(x,t)$ is the wave amplitude expressed as a function of space and time. We assume that the string is bounded so that $\Psi(x=0,t) = 0 = \Psi(x=L,t)$, where L is the length of the string.

We attempt a numerical solution method by introducing a uniform discretization of the spatial and temporal domains, and approximating the partial differential equation by finite difference expressions. Let

$$\Psi(x,t) \equiv u(i,j) \tag{2.2}$$

denote the discretized wave function that gives the wave amplitude at a given position i

and timestep j. In order to approximate the wave function by our discretized version, we first approximate the first order spatial derivative using finite differences. The approximation can be obtained as

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

or

$$\frac{\partial \psi}{\partial x} \equiv \frac{u(i,j) - u(i-1,j)}{\Delta x}.$$

We can apply the same approximation again to the discretized first order derivative to obtain the second order derivatives

$$\frac{\partial^2 \psi}{\partial x^2} \equiv \frac{u(i+1,j) + u(i-1,j) - 2u(i,j)}{(\Delta x)^2}, \tag{2.3}$$

and similarly,

$$\frac{\partial^2 \psi}{\partial t^2} \equiv \frac{u(i,j+1) + u(i,j-1) - 2u(i,j)}{(\Delta t)^2}. \tag{2.4}$$

Finally, after substituting the second order derivative terms with their approximations in Equation 2.2, we get

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

$$c^2 \frac{u(i+1,j) + u(i-1,j) - 2u(i,j)}{(\Delta x)^2}. \tag{2.5}$$

We can rewrite this to the form of our numerical scheme,

$$u(i,j+1) =$$

$$c^2 \frac{(\Delta t)^2}{(\Delta x)^2}[u(i+1,j) + u(i-1,j) - 2u(i,j)]$$

$$- u(i,j-1) + 2u(i,j). \tag{2.6}$$

For the simulation of the vibrating string, we will iterate over a certain number of time steps and perform updates of the discretized wave function like above. Thus, we need to store both the values of the current time step $u(i,j)$ and the values of the last time step $u(i,j-1)$.

2.2 Time dependent diffusion equation

The two-dimensional time dependent diffusion equation is

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right), \tag{2.7}$$

where D is the diffusion coefficient. The solution $c(x,y,t)$ is the concentration as a function of the two spatial coordinates x and y and the time t. In order to solve this equation numerically, we formulate a discretized version of the function

$$c(x,y,t) \equiv u(i,j,k) \tag{2.8}$$

such that $u(i,j,k)$ gives the approximated concentration at the two-dimensional position i,j at timestep k. First of all, we can directly derive the discretized approximation of the left-hand side of Equation 2.7:

$$\frac{\partial c}{\partial t} \equiv \frac{u(i,j,k+1) - u(i,j,k)}{\Delta t}. \tag{2.9}$$

Secondly, the spatial first-order and second-order derivations can be obtained as in Section 2.1:

$$\frac{\partial^2 c}{\partial x^2} \equiv \frac{u(i+1,j,k) + u(i-1,j,k) - 2u(i,j,k)}{(\Delta x)^2}, \tag{2.10}$$

and, similarly,

$$\frac{\partial^2 c}{\partial y^2} \equiv \frac{u(i,j+1,k) + u(i,j-1,k) - 2u(i,j,k)}{(\Delta x)^2} \tag{2.11}$$

Substituting this into Equation 2.7, we get

$$u(i,j,k+1) =$$

$$D\frac{\delta t}{(\delta x)^2}[u(i+1,j,k) + u(i-1,j,k)$$

$$+ u(i,j+1,k) + u(i,j-1,k) - 4u(i,j,k)]$$

$$+ u(i,j,k). \tag{2.12}$$

Using this iterative scheme, we can update the concentration based on the concentration of surrounding positions at the previous timestep.

2.3 Time independent diffusion equation

The time independent diffusion equation only considers the steady state of the system, that is where

$$0 = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right). \tag{2.13}$$

Taking the same discretization as for the time-dependent diffusion equation, we can substitute the first-order temporal derivative with 0 in Equation 2.12. The result is the iterative scheme

$$c_{i,j}^{k+1} = \frac{1}{4}\left(c_{i+1,j}^{k} + c_{i-1,j}^{k} + c_{i,j+1}^{k} + c_{i,j-1}^{k}\right), \tag{2.14}$$

where the superscript k denotes the k-th iteration and $c_{i,j}$ denotes the concentration at position i, j. This iterative scheme is also called the *Jacobi iteration*. In order to evaluate if the steady state is reached, some stopping condition needs to be introduced. Assume that the solution is assumed to be converged if for all values of i, j

$$\delta \equiv \max_{i,j} |c_{i,j}^{k+1} - c_{i,j}^{k}| < \epsilon, \tag{2.15}$$

where ϵ is some small number. The Jacobi iteration can be improved in order to reduce the number of iterations needed for convergence. Furthermore, the Jacobi iteration currently needs the storage of two separate matrices for $c_{i,j}^{k}$ and $c_{i,j}^{k+1}$.

An improvement over the Jacobi iteration is the *Gauss-Seidel iteration*, where during the iteration a new value is used as soon as it has been calculated. Assuming that the iteration proceeds along the rows (i.e. incrementing i for fixed j), the Gauss-Seidel iteration is

$$c_{i,j}^{k+1} = \frac{1}{4}\left(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}\right). \tag{2.16}$$

Although the Gauss-Seidel iteration does not reduce the number of iterations needed for convergence substantially, it requires less storage since the update is performed in place. The Gauss-Seidel iteration can be further improved by adding an over-correction of the new iterate:

$$c_{i,j}^{k+1} = \frac{\omega}{4}\left(c_{i+1,j}^{k} + c_{i-1,j}^{k+1} + c_{i,j+1}^{k} + c_{i,j-1}^{k+1}\right) + (1-\omega)c_{i,j}^{k} \tag{2.17}$$

The new resulting scheme is called *Successive Over Relaxation (SOR)*. This method only converges for $0 < \omega < 2$. For $w < 1$, the method is called under-relaxation. The new value is then a weighted average of the Gauss-Seidel method and the previous concentration value. For $\omega = 1$, the SOR iteration is equivalent to the Gauss-Seidel iteration. Previous research shows that different values of ω yield different numbers of iterations before convergence [1]. Hence, one of the ways to optimize this iterative scheme is to search for an optimal value of ω.

3 Experimental Method

3.1 Vibrating string

In the implementation of the vibrating string, the position of the string is represented as a 1-dimensional array. Updating the position at every point along the string can be done using Equation 2.6. As the values from the current time step and the previous step are necessary to calculate the new position, both the current and previous state of the string needs to be stored.

To test the effect of the initial condition on the behavior of the string, we will simulate the system under three different initial conditions,

$$\Psi(x, t=0) = sin(2\pi x), \tag{3.1}$$

$$\Psi(x, t=0) = sin(5\pi x), \tag{3.2}$$

$$\Psi(x, t=0) = sin(5\pi x) \text{ if } 0.2 < x < 0.4. \tag{3.3}$$

In this experiment, the value for c in Equation 2.6 is set to 1 and the time step is equal to 0.001. As mentioned in section 2.1, the boundary conditions of the string are $\Psi(x = 0, t) = 0 = \Psi(x = L, t)$. The length of the string is set to 1 and the string is divided into 200 intervals.

3.2 Time dependent diffusion

To simulate a diffusion process using the discretized time dependent diffusion equation, we start by initializing an N by N grid. In our experiments, N is equal to 40, the boundary conditions are

$$c(x, y=1, t) = 1, \tag{3.4}$$

$$c(x, y=0, t) = 0, \tag{3.5}$$

$$c(x=0, y, t) = c(x=1, y, t), \tag{3.6}$$

and the initial condition is

$$c(x, y, t = 0) = 0 \text{ for } 0 \leq x \leq 1, 0 \leq y < 1. \tag{3.7}$$

Using Equation 2.12, we can calculate the new concentration in every grid cell based on the concentration in the previous time step. During the experiments, the diffusion constant, D, was set to 1. Because $\frac{4D\delta t}{\delta x^2}$ needs to be significantly smaller than 1 to have a stable time propagation, we set δt equal to 10^{-4}.

In the comparison of the numerical solution of the diffusion process to the analytical solution, we let the simulation run until the time equals one and during the simulation, every 100 time steps, we store the grid in a txt file. After the simulation, we extract the numerical solution at times t = $\{0.01, 0.1,$ and $1\}$ from the txt file. At those times, we also calculate the analytical solution,

$$c(y, t) = \sum_{i=0}^{\infty} \text{erfc}\left(\frac{1 - y + 2i}{2\sqrt{Dt}}\right) - \text{erfc}\left(\frac{1 + y + 2i}{2\sqrt{Dt}}\right) \tag{3.8}$$

In this equation, erfc is the error function and in the implementation the sum until infinity is limited to 1000. The y-coordinates at which this equation is calculated are linearly spaced between 0 and 1 with an interval of 10^{-3}. Note that due to the boundary and initial conditions, the final solution is independent of the x-coordinate on the grid. Therefore, the comparison of a single column of the grid to the analytical solution is sufficient.

As the concentration on every gridpoint is stored every 100 time steps, it is also possible to visualize the diffusion process in a 2D plot where the intensity reflects the concentration. We made this plot at different time steps (t = $\{0, 0.001, 0.01, 0.1, 0.2$ and $1\}$) to gradually see the process reach an equilibrium.

To see the effect on the diffusion process of an insulating material, we set the diffusion constant in the cells located in the middle 5 rows equal to 0.05 and ran the same experiments.

3.3 Time independent diffusion

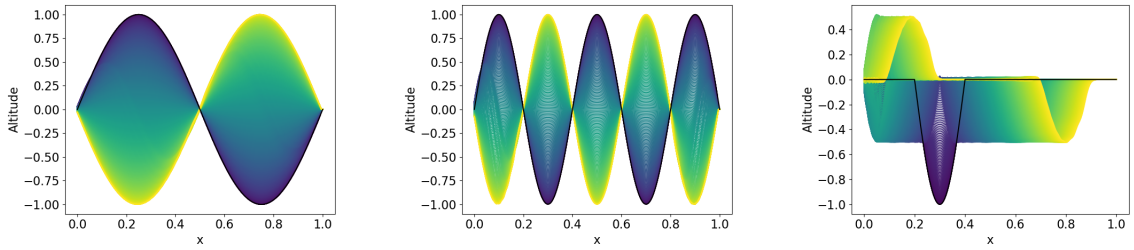
Solving the discretized version of the time independent diffusion equation boils down to solving a system of linear equations, for which three possible iterative schemes were presented in section 2.3. Similar to the time dependent case, we start by initializing an N by N grid and the boundary and initial conditions are given by Equations 3.4 - 3.7.

With the Jacobi scheme, we iteratively update the concentration in every gridpoint using Equation 2.14. Note that the entire grid needs to be copied before the updating step because the old value in an already calculated gridpoint is still necessary to calculate the new value in a neighboring point. In the Gauss-Seidel scheme, we use Equation 2.16 instead, where the new values are used for two of the neighboring cells. When iterating over the grid one row at a time from left to right, this allows us to update the grid in place and a copy of the grid is no longer required. The approach to implementing the SOR scheme is similar to the Gauss-Seidel scheme. The only difference is that we use Equation 2.17 instead in the updating step, where a relaxation parameter is taken into account. For all three methods, the convergence of the scheme is checked at every iteration by comparing the new value at every gridpoint with the old value. If the difference in concentration between two iterations is smaller than a set threshold, δ, for every gridpoint, then the iterative process stops.

With the implementation of these three different schemes, we can compare them based on accuracy and computational cost. To determine the accuracy, we can solve the time independent diffusion equation with all three methods and compare the resulting concentration as a function of the y-coordinate because we know from the analytical solution that this should be a straight line. For this experiment, we used a 50 by 50 grid and set δ equal to $5 \cdot 10^{-5}$. In the SOR method, the relaxation parameter was set to 1.9.

To compare the computational cost of the different methods, we can look at the number of iterations it takes for each method to converge. Again, we used a 50 by 50 grid but we let the threshold vary from 10^{-3} to 10^{-8} to see how many iterations each method needs depends on the magnitude of the threshold. To get an idea of the influence that the relaxation parameter has on the number of iterations, we used the SOR method with both ω equal to 1.5 and 1.9.

The SOR method still contains a free parameter, ω, and the optimal value for it, i.e. the one that requires the least number of iterations, depends on several factors. To determine the dependence on the grid size, we counted the number of iterations the SOR method needed to converge, for a grid of 10 by 10, 30 by 30 and 50 by 50. The values for ω that were used in this experiment were linearly spaced between 1.5 and



(a) $\Psi(x, t = 0) = sin(2\pi x)$ **(b)** $\Psi(x, t = 0) = sin(5\pi x)$ **(c)** $\Psi(x, t = 0) = sin(5\pi x)$ if $0.2 < x < 0.4$

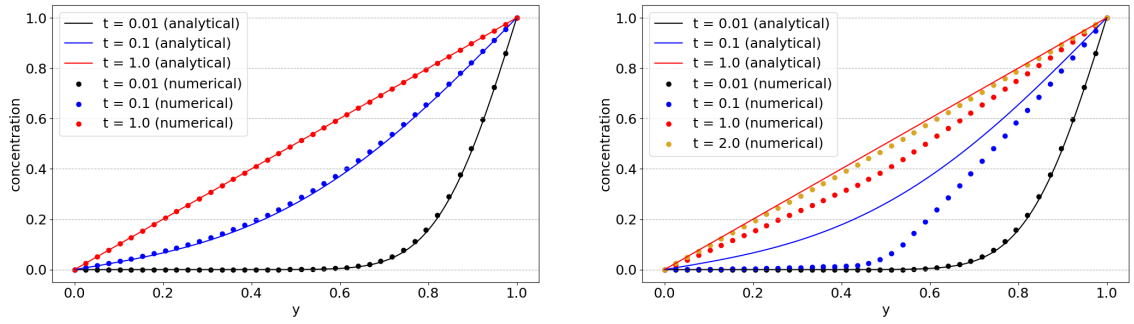
Figure 1. Change in the position of the string as a function of time. The black line corresponds to the initial position of the string and the position of the string changes from the darker color to the lighter color during the simulation. The time step is 0.001. The left and right graphs are shown for the first 500 iterations and the middle one for the first 200 iterations.

1.99 with an interval of 0.01. The threshold in these experiments was $5 \cdot 10^{-5}$. To test how the optimal ω would change if the grid contained objects, we determined the optimal ω for a grid with 0, 1 and 2 rectangular objects. For this experiment, we chose a 50 by 50 grid and the objects were 6 by 8 rectangles where the concentration was equal to zero during the entire simulation. The placement of the rectangles can be seen in Figure 9, where the final solution of the three grids is shown. Determining the optimal ω was done in the same way as the previous experiment.

4 Results and Discussion

4.1 Vibrating string

In Figure 1, the vibrating behavior of a string is presented for different initial conditions. The lightest color on the left graph is the position of the string after 500 iterations, while the lightest color on the middle graph is the position after 200 iterations. This indicates that with an initial condition of $\Psi(x, t = 0) = sin(5\pi x)$, the string will oscillate with a higher frequency. This follows from Equation 2.6 because the difference in altitude between neighboring points on a string will be larger when the initial condition is $\Psi(x, t = 0) = sin(5\pi x)$ compared to $\Psi(x, t = 0) = sin(2\pi x)$. Therefore, the difference in position between two time steps will be larger as well. The vibrating string in the right graph of Figure 1, no longer produces a standing wave but has a more complex behavior. Initially, the string is stretched out only between 0.2 and 0.4. This causes local oscillatory behavior and a traveling wave toward the two boundaries. When the traveling wave hits the left bound of the string, we can see a shift in the amplitude from negative to positive.



(a) Without insulation material **(b)** Insulation material in the middle 5 rows

Figure 2. The concentration as a function of the y-coordinate at different time steps. At y equals one, there is a heat source (c=1) and at y equals zero, there is a sink (c=0). The lines represent the analytical solution and the dots are from the numerical solution.

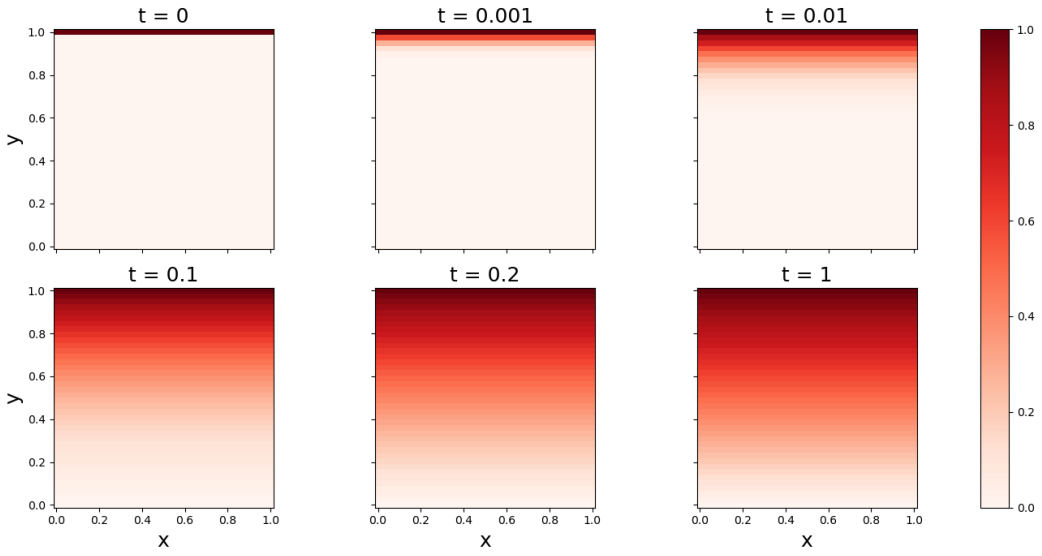


Figure 3. The concentration plotted on a 2-dimensional grid at time 0, 0.0001, 0.01, 0.1, 0.2 and 1. The intensity of the color reflects the magnitude of the concentration. The dark red color corresponds to a high concentration and the light red to a low concentration.

4.2 Time dependent diffusion

Figure 2a shows the concentration as a function of the y-coordinate at several moments in time for the numerical and analytical solution. At all three moments in time, the numerical solution is close to the analytical solution. At a y-position of 1, the concentration is always equal to one, and at a y-position of 0, the concentration is always equal to zero, as expected, given the boundary conditions. At the beginning of the simulation, the concentration drops fast as y decreases and over time, this gradually smooths out until the concentration as a function of the y-coordinate behaves like a straight line.

The same trend can be seen in Figure 3, which show the concentration plotted on a 2-dimensional grid. At time t = 0.001, the concentration increase due to the diffusion from the heat source at the top row is only present in the few rows below it. As time progresses, the concentration also increases in the rest of the grid until the equilibrium is reached.

If an insulating material is added to the system in the middle rows, we can see in Figure 2b that the diffusion process is normal at the beginning of the simulation. Once the concentration reaches non-zero levels at the insulation material, the diffusion slows down. When the insulation is present, the same equilibrium state is still reached but it does take longer. This process is also clearly visible in Figure 4, where the grid is plotted at several points in time.

4.3 Time independent diffusion

Figure 5 shows the concentration as a function of the y-coordinate at equilibrium, using the Jacobi, Gauss-Seidel and SOR iterative scheme. Even though the stopping criterion is the same for all three methods, the SOR method is closer to the analytical solution compared to the other two methods. The Jacobi method results in a relatively large deviation from the analytical solution because, during a single iteration, it only uses information from the previous iteration, while the other methods already use newly calculated information for half of the neighboring cells.

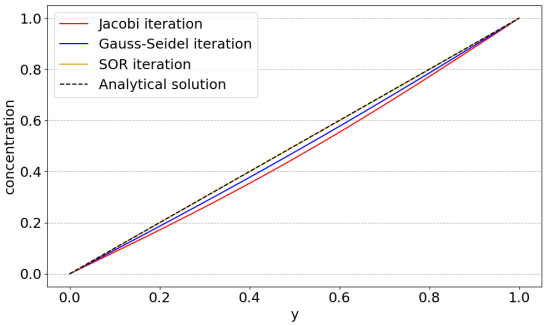


Figure 5. The concentration as a function of the y-coordinate at equilibrium, using the Jacobi, Gauss-Seidel and SOR iterative scheme. At y equals one, there is a heat source (c=1) and at y equals zero, there is a sink (c=0).

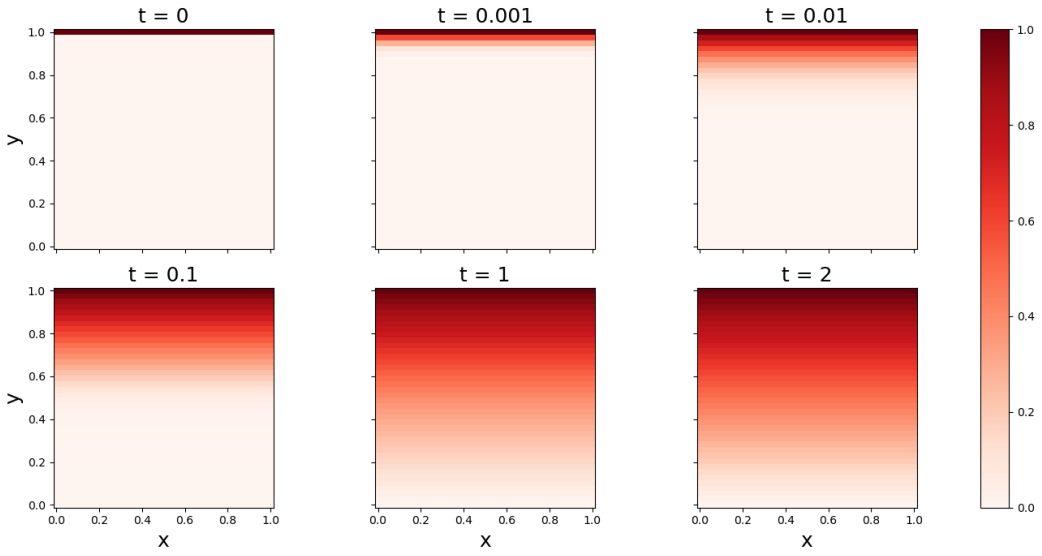


Figure 4. The concentration plotted on a 2-dimensional grid at time 0, 0.0001, 0.01, 0.1, 1 and 2. The intensity of the color reflects the magnitude of the concentration. The dark red color corresponds to a high concentration and the light red to a low concentration. The middle 5 rows contain a insulation material (D = 0.05).

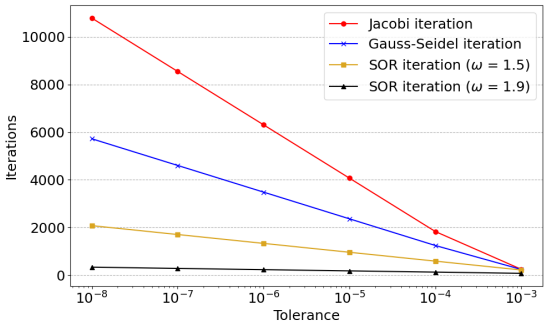


Figure 6. Number of iterations necessary for convergence as a function of the stopping criterion for the different iterative schemes.

Figure 6 depicts how the number of iterations for convergence depends on the iterative scheme and the stopping criterion. As the Jacobi iteration method needs the most number of iterations until convergence, it is computationally the most expensive method. Switching to the Gauss-Seidel method results in a number of iterations that is about twice as low. If we look at the first iteration for example, in the Jacobi only the second row will have new nonzero values because it is the only row that borders the top row. All the other cells in the grid will continue to have a zero value after the first iteration because they are all surrounded by points with a zero concentration. This is different in the Gauss-Seidel method because, for half of the neighboring cells, the newly updated concentration is used. Using the SOR iteration method leads to an even greater speed-up, especially when setting the relaxation parameter to 1.9. The reasoning is very similar to how the Gauss-Seidel method resulted in a speed-up. The effect is only larger because the part that contains the newly updated information has a larger weight when the relaxation parameter is above 1.

The interplay between the relaxation parameter and the grid size and their effect on the number of iterations is shown in Figure 7. For a denser grid, the number of iterations necessary to reach the converged equilibrium is higher because the number of rows for which the equilibrium concentration is above a certain value is larger. Initially, the number of iterations decreases while ω increases, but above the optimal ω, the number of iterations starts to increase significantly, because the newly calculated concentration can be above the equilibrium concentration. The optimal value for ω is lower for smaller grids because the equilibrium concentration in a specific row is lower than the corresponding row in a larger grid. For example, the equilibrium concentration in the second row for a 10 by 10 grid is 0.9, while it is 0.98 in a 50 by 50 grid.

Figure 8 shows the number of iterations necessary for convergence using the SOR method as a function of the relaxation parameter for a grid with 0, 1 and 2 rectangular objects. Adding more objects results in an overall lower number of iterations because there are more grid points

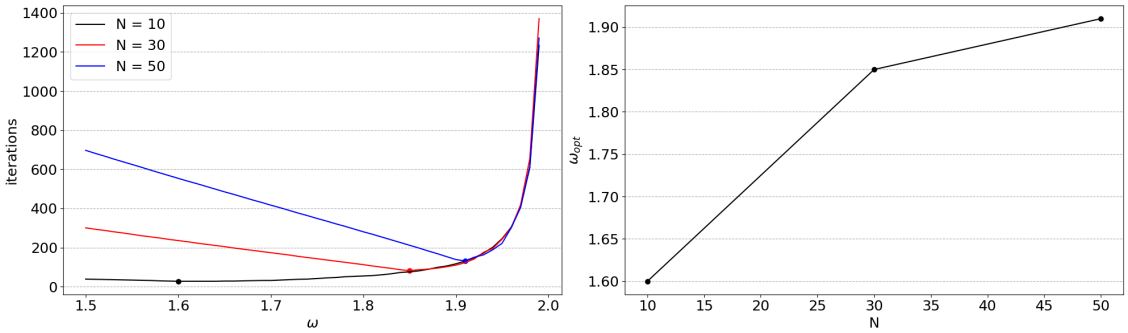


Figure 7. Left: Number of iterations as a function of the relaxation parameter in the SOR scheme for different grid sizes. The dots represent the optimal ω. Right: The optimal ω as a function of the grid size.

for which the concentration remains zero and thus fewer unknown concentrations in the system of linear equations. We can also see that the optimal value for ω decreases if more rectangular objects are placed on the grid. A similar reasoning can be applied here as for the dependence of the optimal ω on the grid size. With the presence of objects, there are areas in the grid where the gradient of the equilibrium concentration is higher. In Figure 9, this is visible in the middle and right graph, where the equilibrium concentration is very low near the objects and very high near the heat source at the top.

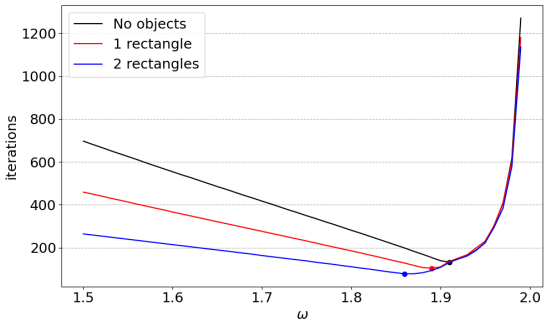


Figure 8. Number of iterations necessary for convergence using the SOR method as a function of the relaxation parameter for a grid 0, 1 and 2 rectangular objects.

5 Conclusion

From simulating a vibrating string by numerically solving the one-dimensional wave equation, we can conclude that the initial condition of the string has a significant effect on its behavior.

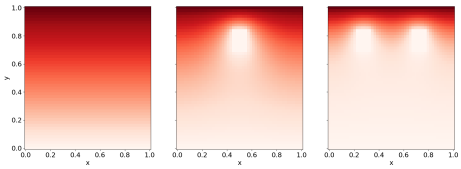


Figure 9. The equilibrium concentration for a grid with 0, 1 and 2 rectangular objects.

A more oscillating initial condition will result in a string that vibrates faster and a non-symmetric initial condition can lead to more complex behavior instead of a standing wave.

Simulating a diffusion process by numerically solving the time dependent diffusion equation gives a clear picture of how the concentration spreads away from a source until an equilibrium is reached.

Out of the three iterative schemes that were used to solve the time independent diffusion equation, the Jacobi iteration method is computationally the most expensive, while the SOR method needs relatively few iterations to converge if the right value is chosen for the relaxation parameter. When increasing the density of the grid, the optimal value for ω increases as well. On the other hand, placing objects on the grid results in a lower optimal value for the relaxation parameter.

References

[1] D. M. Young and T.-Z. Mai, "The search for omega," in *Iterative Methods for Large Linear Systems*, Elsevier, 1990, pp. 293–311.