# The Wave Equation

Leonardo Werneck

December 26, 2019

## Contents



## 1 The Wave Equation

Here we will explore the wave equation and discuss how to solve it in the computer. For the sake of this discussion, we will be using the C++ programming language.

For a given function $u(t,\vec{x})$, where $\vec{x} = (x,y,z)$, the *wave equation* is given by

$$\partial_t^2 u(t,\vec{x}) = c^2 \nabla^2 u(t,\vec{x}) \ , \tag{1}$$

where $\nabla^2$ is the Laplacian operator, which in Cartesian coordinates reads

$$\nabla^2 = \partial_x^2 + \partial_y^2 + \partial_z^2 \ . \tag{2}$$

Above we have used the fairly standard notation:

$$\partial_t \equiv \frac{\partial}{\partial t} \ , \quad \partial_t^2 \equiv \frac{\partial}{\partial t^2} \ , \quad \text{etc} \ . \tag{3}$$

### 1.1 Solving the wave equation in 1-dimensional Cartesian coordinates using finite differences

We will start studying the wave equation by focusing on the simplest case, the 1D wave equation. Considering that the one spatial dimension is the $x$-direction, we have

$$\partial_t^2 u(t,x) = c^2 \partial_x^2 u(t,x) \ . \tag{4}$$

A standard technique when dealing with an equation like this is to define a new auxiliary function

$$v(t,x) \equiv \partial_t u(t,x) \ , \tag{5}$$

so that we trade one second-order differential equation by a system of coupled, first-order differential equations:

$$\begin{aligned} \partial_t u(t,x) &= v(t,x) \ , \\ \partial_t v(t,x) &= c^2 \partial_x^2 u(t,x) \ . \end{aligned} \tag{6}$$

One could also simplify the Laplacian operator by introducing yet another auxiliary function $w_x(t,x) \equiv \partial_x u(t,x)$, but we won't do that here.

### 1.1.1 The evolution equations

We will use second-order, centered finite differences to solve this problem. Introducing the notation

$$f\left(n \cdot \Delta t, i \cdot \Delta x\right) \equiv f_i^n \ , \tag{7}$$

we have thus the approximations

$$\begin{aligned} \partial_t f_i^n &= \frac{f_i^{n+1} - f_i^{n-1}}{2\Delta t} \ , \\ \partial_x f_i^n &= \frac{f_{i+1}^n - f_{i-1}^n}{2\Delta x} \ , \\ \partial_x^2 f_i^n &= \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{\Delta x^2} \ , \end{aligned} \tag{8}$$

which allow us to write down the wave equation as

$$\begin{aligned} \frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} &= v_i^n \ , \\ \frac{v_i^{n+1} - v_i^{n-1}}{2\Delta t} &= c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \ , \end{aligned} \tag{9}$$

leading to the iterative relations

$$\boxed{\begin{aligned} u_i^{n+1} &= u_i^{n-1} + 2\Delta t v_i^n \\ v_i^{n+1} &= v_i^{n-1} + \frac{2c^2 \Delta t}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \end{aligned}} \ . \tag{10}$$

## 1.2 The initial data problem

As can be seen from the boxed equation above, this is a scheme that involves *three time levels*, since the left-hand sides (LHS) are terms that depend on the time level $n+1$, while the right-hand sides (RHS) contain terms on the time levels $n-1$ and $n$.

The wave equation is a second-order differential equation, so it is indeed expected that we provide *two* initial conditions in order to specify a solution. Say we wish to specify the initial conditions $u(0,x) = f(x)$ and $\partial_t u(0,x) = v(0,x) = 0$. These are easily implemented using

$$\begin{aligned} u_i^0 &= f(x) \ , \\ v_i^0 &= 0 \ . \end{aligned} \tag{11}$$

This is all very well, but now how do we evolve the system in time? We currently have access to $\left(u^0\_i, v^0\_i\right)$, but if we use our evolution equations with $n = 1$, we will need to have access to $\left(u^1\_i, v^1\_i\right)$ *before* we can compute $\left(u^2\_i, v^2\_i\right)$. Obtaining $\left(u^1\_i, v^1\_i\right)$ then becomes a crucial part of the initial data specification known as the ***initial data problem***.

To make it clear, when the user specify $\left(u^0\_i, v^0\_i\right)$ we are giving the *initial conditions* required by the differential equation. However, due to our choice of numerical scheme, a different, *artificial* initial condition is also required, that is $\left(u^1\_i, v^1\_i\right)$. The combination of these two initial conditions is referred to as the *initial data*.

A common trick to obtain this initial data is the following. Consider a *half-step forward* iteration

$$\begin{aligned} \partial_t u_i^{n+\frac{1}{2}} &= \frac{u_i^{n+\frac{1}{2}} - u_i^n}{\Delta t/2} = v_i^n \ , \\ \partial_t v_i^{n+\frac{1}{2}} &= \frac{v_i^{n+\frac{1}{2}} - v_i^n}{\Delta t/2} = \frac{c^2}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \ , \end{aligned} \tag{12}$$

followed by a *half-step* centered iteration

$$\begin{aligned} \partial_t u_i^{n+\frac{1}{2}} &= \frac{u_i^{n+1} - u_i^n}{\Delta t} = v_i^{n+\frac{1}{2}} \ , \\ \partial_t v_i^{n+\frac{1}{2}} &= \frac{v_i^{n+1} - v_i^n}{\Delta t} = \frac{c^2}{\Delta x^2}\left(u_{i+1}^{n+\frac{1}{2}} - 2u_i^{n+\frac{1}{2}} + u_{i-1}^{n+\frac{1}{2}}\right) \ . \end{aligned} \tag{13}$$

Notice that these derivatives are centered at the $n+\frac{1}{2}$ point, as opposed to the usual $n$. Thus, to obtain initial data we use the following algorithm (assuming $n = 0$ corresponds to the initial data level)

$$
\boxed{
\begin{aligned}
u_i^0 &= f(x) \\
v_i^0 &= 0 \\
u_i^{\frac{1}{2}} &= u_i^0 + \frac{\Delta t}{2} v_i^0 \\
v_i^{\frac{1}{2}} &= v_i^0 + \frac{c^2 \Delta t}{2\Delta x^2} \left( u_{i+1}^0 - 2u_i^0 + u_{i-1}^0 \right) \\
u_i^1 &= u_i^0 + \Delta t v_i^{\frac{1}{2}} \\
v_i^1 &= v_i^0 + \frac{c^2 \Delta t}{\Delta x^2} \left( u_{i+1}^{\frac{1}{2}} - 2u_i^{\frac{1}{2}} + u_{i-1}^{\frac{1}{2}} \right)
\end{aligned}
} \quad . \tag{14}
$$

### 1.2.1 Outer boundary conditions

The wave equation is defined everywhere, that is $-\infty < x, y, z < \infty$. However, when solving the problem numerically, we must restrict our attention to a finite domain, which in turn introduces the necessity of imposing *outer boundary conditions*.

Outer boundary conditions specify the behaviour of the functions $u(t, x)$ and $v(t, x)$ at the spatial boundaries of the computational domain. To see why we need this, notice that the spatial derivatives with respect to $x$ require us to know the function $u$ at the points $\left(u_{i-1}^n, u_i^n, u_{i+1}^n\right)$. However, if we have $N_x + 1$ points along the $x$-direction discretization, i.e. $i = 0, 1, \ldots, N_x$, then we cannot evaluate the following quantities:

$$
\begin{aligned}
\partial_x^2 u_0^n &= \frac{u_1^n - 2u_0^n + u_{-1}^n}{\Delta x^2} \ , \\
\partial_x^2 u_{N_x}^n &= \frac{u_{N_x+1}^n - 2u_{N_x}^n + u_{N_x-1}^n}{\Delta x^2} \ ,
\end{aligned} \tag{15}
$$

since we do not have the values $u^n\_-1$ and $u^n\_N_x + 1$.

In order to be able to use our numerical scheme throughout the entire numerical grid, we introduce *artificially* the points $i = -1$ and $i = N_x + 1$ to the grid. The portion of the numerical grid which contains *only* the points $i = 0, 1, \ldots, N_x$ is referred to as the ***interior grid*** while the points $i = -1$ and $i = N_x + 1$ are referred to as ***external grid points*** or ***ghostzones***. Figure 1 illustrates the numerical grid.

The number of ghostzones is scheme dependent. For example, choosing to evaluate the spatial derivative $\partial_x u(t, x)$ using $\mathcal{O}\left(\Delta x^2\right)$ centered finite differences only requires 1 ghostzone at each boundary of the computational domain. On the other hand, a scheme evaluating $\partial_x u(t, x)$ using $\mathcal{O}\left(\Delta x^4\right)$ centered finite differences would required 2 ghostzones at each boundary, and one using $\mathcal{O}\left(\Delta x^6\right)$ centered finite differences would required 3.

The idea behind ghostzones is to be able to naturally evaluate all points that belong to the interior grid, without modifying our scheme. Then, af-

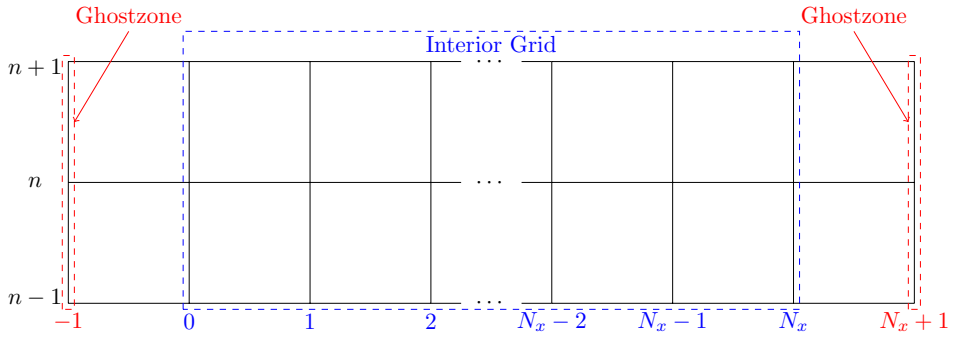


Figure 1: Illustration of computational grid. The interior grid is composed of the blue spatial points and the ghostzones are composed of the red spatial points. The illustration also shows the three time levels required by the numerical scheme described here.

ter all interior grid points have been determined, we apply outer boundary conditions to the ghostzones, filling up the entire grid before moving on to the next time step.

Since our numerical scheme of choice only requires one ghostzone at each outer boundary, we will choose to impose Dirichlet boundary conditions of the form

$$\boxed{\begin{aligned} u^n_{-1} &= 0 = u^n_{N_x+1} \\ v^n_{-1} &= 0 = v^n_{N_x+1} \end{aligned}} \,. \tag{16}$$

### 1.2.2 Outline of the algorithm

We now outline the numerical algorithm we will use to solve the wave equation. It is a combination of all the equations we have boxed this far:

1. Set the initial condition

$$\boxed{u^0_i = f(x) \ , \ v^0_i = 0} \,,$$

   where $f(x)$ is a function of our choosing. This initial condition is applied ¡em¿to the interior grid¡/em¿.

2. Set boundary conditions to the initial condition

$$\boxed{u^0_{-1} = u^0_{N_x+1} = v^0_{-1} = v^0_{N_x+1} = 0} \,.$$

3. Find the initial data.

3.1. Start by computing the "half-step forward" approximation

$$\boxed{\begin{aligned} u_i^{\frac{1}{2}} &= u_i^0 + \frac{\Delta t}{2} v_i^0 \\ v_i^{\frac{1}{2}} &= v_i^0 + \frac{c^2 \Delta t}{2\Delta x^2} \left( u_{i+1}^0 - 2u_i^0 + u_{i-1}^0 \right) \end{aligned}} .$$

3.2. Next, apply boundary conditions

$$\boxed{u_{-1}^{\frac{1}{2}} = u_{N_x+1}^{\frac{1}{2}} = v_{-1}^{\frac{1}{2}} = v_{N_x+1}^{\frac{1}{2}} = 0} .$$

3.3. Then, compute the "half-step centered" approximation

$$\boxed{\begin{aligned} u_i^1 &= u_i^0 + \Delta t v_i^{\frac{1}{2}} \\ v_i^1 &= v_i^0 + \frac{c^2 \Delta t}{\Delta x^2} \left( u_{i+1}^{\frac{1}{2}} - 2u_i^{\frac{1}{2}} + u_{i-1}^{\frac{1}{2}} \right) \end{aligned}} .$$

3.4. Finally, apply boundary conditions

$$\boxed{u_{-1}^1 = u_{N_x+1}^1 = v_{-1}^1 = v_{N_x+1}^1 = 0} .$$

4. Start the main time loop. While $n$ ¡ $N_t^{\max}$, do:

4.1. Step the interior grid forward in time using our $\mathcal{O}\left(\Delta t^2 + \Delta x^2\right)$ centered finite differences scheme, namely

$$\boxed{\begin{aligned} u_i^{n+1} &= u_i^{n-1} + 2\Delta t v_i^n \\ v_i^{n+1} &= v_i^{n-1} + \frac{2c^2 \Delta t}{\Delta x^2} \left( u_{i+1}^n - 2u_i^n + u_{i-1}^n \right) \end{aligned}} .$$

4.2. Apply boundary conditions

$$\boxed{u_{-1}^{n+1} = u_{N_x+1}^{n+1} = v_{-1}^{n+1} = v_{N_x+1}^{n+1} = 0} .$$