CS 489

Dr. Richard Mann

University of Waterloo

SIMULATING THE 2-D WAVE EQUATION

Christopher Finn Plummer

14-04-2021

Contents

1 Introduction

The 2-D wave equation is a widely studied topic with a wide variety of applications in physics and applied mathematics. How we can solve the equation numerically or analytically changes with every context that it is used in. This report focuses on a numerical solution for developing a simulation environment to use as a toolkit to explore various topics of the wave equation in acoustics. Initially, we construct a time-stepping algorithm from the 2-D wave equation which acts as the foundation of the simulation environment, from which numerous tools are introduced to aid in exploration of the 2-D wave equation and acoustics. Furthermore, this report demonstrates some of those applications in acoustics and we discuss some of the results we have created using this toolkit.

Additionally, find my presentation for this project and report at:
`https://www.youtube.com/watch?v=EC4wLg-_Kz8`

2 Constructing a Time-Stepping Algorithm:

2.1 Discretization of Wave Equation

We are focused on simulating the 2-D wave equation as follows:

$$\frac{\partial^2 s}{\partial t^2} = c^2 \frac{\partial^2 s}{\partial x^2} + c^2 \frac{\partial^2 s}{\partial y^2} + f \tag{1}$$

where s denotes a 2-d space and f is a source function on s. As follows in [4] we assert that our temporal step size is consistent throughout the simulation and we denote this value as Δt, additionally, we also assert that the spatial step size is consistent in both the x and y axis, denoted as Δl. Therefore, we can use the central difference formula as outlined in [4], equation (17) and (18), and we get:

$$\frac{\partial^2 s(t,x,y)}{\partial t^2} = \frac{s(t+\Delta t,x,y) - 2s(t,x,y) + s(t-\Delta t,x,y)}{\Delta t^2} + o(\Delta t^2) \tag{2}$$

$$\frac{\partial^2 s(t,x,y)}{\partial x^2} = \frac{s(t,x+\Delta l,y) - 2s(t,x,y) + s(t,x-\Delta l,y)}{\Delta l^2} + o(\Delta l^2) \tag{3}$$

$$\frac{\partial^2 s(t,x,y)}{\partial y^2} = \frac{s(t,x,y+\Delta y) - 2s(t,x,y) + s(t,x,y-\Delta l)}{\Delta l^2} + o(\Delta l^2) \tag{4}$$

2.2 Introduce a Simulation Mesh

Let u be a 2-d matrix representation mesh of the space, s. We have that $u^t_{i,j}$ denotes the value at the i^{th} row, j^{th} column of u at the t^{th} timestep of the simulation. We can then use the approximation of these dropping the $o(\Delta t^2)$ and $o(\Delta l^2)$ terms, and using (1) and (2)-(4) we get that:

$$\frac{u^{t+1}_{i,j} - 2u^t_{i,j} + u^{t-1}_{i,j}}{\Delta t^2} = c^2 \frac{u^t_{i+1,j} - 2u^t_{i,j} + u^t_{i-1,j}}{\Delta l^2} + c^2 \frac{u^t_{i,j+1} - 2u^t_{i,j} + u^t_{i,j-1}}{\Delta l^2} + f^t_{i,j} \tag{5}$$

$$\Rightarrow u^{t+1}_{i,j} = 2u^t_{i,j} - u^{t-1}_{i,j} + \frac{c^2 \Delta t^2}{\Delta l^2}(u^t_{i+1,j} + u^t_{i-1,j} + u^t_{i,j+1} + u^t_{i,j-1} - 4u^t_{i,j}) + \Delta t^2 f^t_{i,j} \tag{6}$$

and so now we have a way to increment a time-step of each mesh-point on u.

2.3 Introduce CFL Condition

We let $CFL = \frac{c\Delta t}{\Delta l}$, which is our Courant-Friedrichs-Lewy condition, and is used to help optimize the timestep used in the simulation [1]. From [1] we conclude that for a 2-dimensional wave equation we require that $CFL \leq 0.5$. To enforce this condition in our simulation environment we provide the CFL as a parameter into our simulation and compute $\Delta t = \frac{CFL \Delta l}{c}$, rather than computing CFL. However, we can now subsitute $CFL = \frac{c\Delta t}{\Delta l}$ into (6). From this we get that:

$$u^{t+1}_{i,j} = 2u^t_{i,j} - u^{t-1}_{i,j} + CFL^2(u^t_{i+1,j} + u^t_{i-1,j} + u^t_{i,j+1} + u^t_{i,j-1} - 4u^t_{i,j}) + \Delta t^2 f^t_{i,j} \tag{7}$$

which is what we will use as the basis for our time-marching numerical simulation algorithm.

3 Creating a Simulation Environment

Code is attached as *waves.jl* for reference.

3.1 Implement Equation (7)

We are first concerned with lines 159-166 where we implement this algorithm to timestep forward the simulation. In the code we denote u as timestep $t+1$, u_1 as timestep t and u_2 as timestep $t-1$. n denotes the size of u. We have the following code:

```
159.      u = 2*u_1 - u_2
160.      c = -4*u_1[2:end-1,2:end-1]
161.      c += u_1[3:end,2:end-1]
162.      c += u_1[1:end-2,2:end-1]
163.      c += u_1[3:end,3:end]
164.      c += u_1[3:end,1:end-2]
165.      c = CFL^2*c
166.      u[2:n-1,2:n-1] += c
```

which implements (7).

3.2 Boundary Conditions

However, we notice that this doesn't account for the boundaries of u. So we need to account for different boundaries, the first boundary implementation is a reflection boundary on line 22 denoted η_1. This is created by simply setting all points on the boundary to 0.

The second implementation of boundaries uses Mur's absorbing conditions which is on line 33 denoted η_2. We start with the equations for Mur's absorbing boundary:

$$\frac{\partial s}{\partial t} = \frac{1}{c}\frac{\partial s}{\partial x}, \text{ for x = 0} \tag{8}$$

$$\frac{\partial s}{\partial t} = \frac{-1}{c}\frac{\partial s}{\partial x}, \text{ for x = end} \tag{9}$$

$$\frac{\partial s}{\partial t} = \frac{1}{c}\frac{\partial s}{\partial y}, \text{ for y = 0} \tag{10}$$

$$\frac{\partial s}{\partial t} = \frac{-1}{c}\frac{\partial s}{\partial y}, \text{ for y = end} \tag{11}$$

which we can discretize to, as discussed in [3] and demonstrated in [4]:

$$u_{1,j}^{t+1} = u_{2,j}^{t} + \frac{CFL-1}{CFL+1}[u_{2,j}^{n+1} - u_{1,j}^{n}] \tag{12}$$

$$u_{end,j}^{t+1} = u_{end-1,j}^{t} + \frac{CFL-1}{CFL+1}[u_{end-1,j}^{n+1} - u_{end,j}^{n}] \tag{13}$$

$$u_{i,1}^{t+1} = u_{i,2}^{t} + \frac{CFL-1}{CFL+1}[u_{i,2}^{n+1} - u_{i,1}^{n}] \tag{14}$$

$$u_{i,end}^{t+1} = u_{i,end-1}^{t} + \frac{CFL-1}{CFL+1}[u_{i,end-1}^{n+1} - u_{i,end}^{n}] \tag{15}$$

which we can then implement on our mesh to simulate Mur's absorbing conditions.

3.3 Incorperating Other Features

We now have a basis for the simulation. The next features we add allow us to create a flexible simulation environment. The code currently has the following features:

1. $B(u, u_1)$, which allows a function to define the boundary conditions
2. $I(u)$, which allows a function to define the initial state of u
3. $f(u)$, which allows a function to define a source within the environment
4. s, r, are samplers, this allows us to place a Sampler struct within the environment which records the values of each timestep at the position of the Sampler
5. FieldObj's, these are structs that act as an object within the environment. Currently these objects can behave with reflecting and absorbing surfaces

4 Demonstration

The demonstration discusses the output from the script *waves_ demo.jl*

4.1 Installing Packages and Run Demo

We first need to ensure that you have the Plots and IJulia packages for julia, so we will run:

```
julia
julia> using Pkg
julia> Pkg.add("Plots")
julia> Pkg.add("IJulia")
```

If this is the first time using these packages it will take a little time to precompile the packages.
Once the packages have installed we will want to run the demo file as follows:

```
julia> include("waves_demo.jl")
```

Discussing Results:
Unfortunately, I had difficulties including animations of the .gif files directly into the report so I will have to only refer to the outputs generated, find the pre-computed files in the figures folder. The following subsections discuss the outputs.

4.2 Default Examples

The following outputs help to showcase the parameters of the simulation environment.

2d-wave This serves as the default "out-of-the-box" simulation. It exhibits a 10x10 unit surface, a sinusoidal source function in the middle of the surface and reflecting boundaries.

objects This demonstrates the placement of objects onto the mesh for a simulation. We place an absorbing object in the bottom-left corner and a reflecting object in the top-right corner of the field respectively. We see with the assistance of the heatmap how the wave will interact differently with the two unique objects.

4.3 Visualization of Recording Audio

We use this to demonstrate the use of Samplers, and we can use this to show how absorbing objects can help when recording audio. If we allow for a large simplification, we could imagine the Sampler in the middle of the field being a microphone recording a source producing a 1Hz wave.

rfl_mic Here we have reflecting boundaries which we could interpret a recording studio made with concrete walls. We see that there is noticeable interference in the magnitude and shape of the tone from the original source of a 1Hz wave.

abs_mic In contrast, we have an absorbing boundary which we could interpret as an incredibly sought after material that absorbs all incoming waves that are placed alongside the walls of a recording studio. We see that there is almost no interference in the shape of the wave and only a scaled change in magnitude from the original tone.

4.4 Vibration Patterns

The next part of the demonstration is inspired by Chladni Patterns and the initial idea was to recreate these patterns using the toolkit. Originally, Chladni found these patterns when drawing a violin bow along the edge of a thin metal plate that is fixed in the center and is covered in salt. A demonstration of this can be found here at reference [5]. However, when starting to embark on recreating these patterns a problem became immediately apparent. The physics of how a violin bow interacts with the surface is in itself a highly complex interaction to model and simulate. So another popular way to demonstrate a Chladni pattern is instead to vibrate a thin metal plate at the center at various frequencies and similarily numerous patterns also appear. A demonstration of this can be found at reference [6]. However, another problem arose from this. As we saw from the video the frequencies of the metal plate range from 345-3000 Hz, and this is problematic because we must abide by our CFL coefficient for our simulation to run without numerical error. Furthermore, we need to be able to make the time-step small enough to accurately represent the frequency so for even something like a 345Hz source wave we need at least a time-step that is smaller than 1/700. And since this directly corresponds with the spatial step in our mesh we require a much larger computing power then I have access to. So instead of directly trying to replicate these experiments we can investigate the vibration patterns that form within our simulation environment. We want to be able to visualize what would happen to the sand if we placed sand on our mesh. To do so we can introduce a new plot of our simulation, that plots the magnitude of the derivative at each time step, or in other words, the absolute value of the difference between our mesh from time-steps t-1 and t. Then if we sum all these differences throughout the simulation we are left with the plots appended with "-tc.png" and these show the total change in magnitude of a point on the mesh (the darker an area the less it has moved).

bc This acts as our base case for the following simulations. This is a 1Hz wave with the parameters altered as shown in the demo file to allow for a 1/60 time-step. Accompanied with an animation rate of 60fps the output files this and the following are real-time simulations of the waves. In regards to the vibration pattern, we see it is a simply pattern. We could hypothesize that the this is a standing wave which result in such a pattern, however, we see from the 3d animation that rather than a standing wave it is following an oscillatory pattern which is resulting in such a pattern.

abs_edge This looks at if we set all boundaries to be reflecting except for a single absorbing edge. It is interesting to note that there are points of the mesh that have almost absolutely no movement even in field with a majority of reflecting boundaries.

rfl_edge This looks at if we set all boundaries to be absorbing except for a single reflecting. In the opposite way I find it interesting how much movement there is where almost all the boundaries are absorbing which dissipates all energy from the wave. However, it appears that the single reflecting wave allows for large regions of the mesh to still have a bigger change in magnitude.

clmp_ctr What is interesting about this vibration pattern is how little everything moves in comparison to the Chladni patterns with the violin which also have a source along the edge and a clamped center. Particularly, when the surface in our simulation has no stiffness incorporated, whereas, the metal plate is inherited stiff.

odd_src By placing our source at an arbitrary non-center point on the mesh we still see symmetric patterns from our patterns. This was unsuspected by myself particularly since it is symmetric about the line y=x which shows that it can be symmetric not just about the x or y axis.

lim_src This is equivalent to the base case however we the source function becomes null after 3 seconds. We see that this does reach a standing wave unlike the base case which is of note.

4hz This introduces a new symmetric function with a frequency of 4Hz. We see that the vibration patterns become immediately greatly complex in comparison to the previous figures and that this pattern is already on par with the patterns demonstrated at over 2000Hz in the second video linked. We note that this occurs because the lack of stiffness in our wave equation as well as there is a lack of dissipation in energy from the mesh.

4.5 1-D Wave Equation

That last thing in the demo that is illustrated is how we can use this 2-D toolkit to create a pseudo 1-D simulation of the wave equation with absorbing or fixed boundary points. This could potentially be used to emulate the strings of the guitar among other instruments.

1d This illustrates how by placing reflecting edges in our field we can create a pseudo 1-d wave equation simulation with either absorbing or fixed end points.

4.6 Conclusion

From the demonstration we saw a few ways we could use the toolkit to explore various topics such as how objects placed in a field alter the propagation of the wave, patterns found on the surfaces of the waves and using it for the 1-D case to emulate instruments.

5 Further Developments

Moving into the future there are three main goals I would like to build on:

1. Also include an option to simulate in the frequency domain as opposed to the time domain. This may allow for the ability to emulate at a higher frequency in the vibration pattern section without as much computing power
2. Introduce a stiffness parameter onto our surface which will allow for a better emulation of a thin metal plate, among other things.
3. Introduce various optimizations to the algorithm to allow for simulations at smaller time-steps without having a large computing power.

6 References

[1] Courant, Richard, Kurt Friedrichs, and Hans Lewy. "On the partial difference equations of mathematical physics." *IBM Journal of Research of Development* 11.2 (1967): 215:234

[2] Stephen, Haroon. "CEE 709 - Numerical Methods in Mechanics" *University of Nevada* https://hstephen.faculty.unlv.edu/teaching-2/cee-709/

[3] G. Mur, "Absorbing Boundary Conditions for the Finite-Difference Approximation of the Time-Domain Electormagnetic-Field Equations" *IEEE Transactions on Electromagnetic Compatibility* 23.4 (1981): 377-382

[4] Sayed, Saiduzzaman. "Numerical Simulation of Wave Equation" *Global Journal of Science Frontier Research: A Physics and Space Science* 14.7 (2014)

[5] The Royal Institution, *Demonstrating Resonance with Chladni Figures - Christmas Lectures with Charles Taylor*. Available at `https://www.youtube.com/watch?v=OLNFrxgMJ6E`. (Accessed April 12, 2021)

[6] brusspup, *Amazing Resonance Experiment!*. Available at `https://www.youtube.com/watch?v=wvJAgrUBF4w`. (Accessed April 12, 2021)