Johnathan von der Heyde – 2019

# P5 – Waves on a String

Once again we explore the computational techniques behind simulations of seemingly simple physics. Here specifically we explore the 1D motion, of waves propagating on a string. There is much one can do here, and lots of possible nuance to be had, but for the sake of brevity and education, only the basic theory is explored.

We begin, as we should, with the fundamental *wave equation*

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2}$$

This equation tells us that the height of the wave $y$ changes over time in the same way that it changes over space $x$ – proportional to the *wavespeed* $c$. Henceforth we consider only cases where $c = 1$, to simplify our methods. The fact that the wavespeed is squared is simply to keep up with the second derivatives, meaning the way we scale our dimensions of space and time is linearly equivalent to adjusting the wavespeed $c$.

Consider then our simplified wave equation

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial^2 y}{\partial x^2}$$

This implies that for each new time step $t_{i+1}$, our waveform merely shifts by $x_{i+1}$. This is equivalent to the idea

$$y(x_{i+1}, t_{i+1}) = w(x_i, t_i)$$

(where the same index $i$ is used for both $x$ and $t$ for aforementioned reasons)

Which is all to say something like *it will be what it was*, just shifted along in space. But of course, the astute may notice that if we kept with this simple method, we'd end up just cloning the wave continuously over all space. What is missing is to get rid of the old waveform, so that the illusion of motion may manifest.

To get our head around this, we should first set up the scene. What is our waveform? Imagining we take a rather loose piece of string tied to walls at both ends, and give it a pluck right in the middle. We should expect waves to be sent out in both directions, just as the initial middle wave disappears. This initial condition is a simplification called the *Gaussian pluck*, because it's none other than a Gaussian waveform

$$y(x_0, t_0) = A * e^{-\alpha x^2}$$

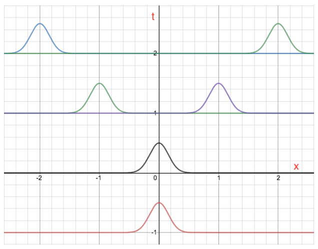
Here $A$ is the amplitude of the wave and $\alpha$ is the wavenumber, which in this context is equivalent to $1/\lambda$ – the wavelength.

To see how this system evolves, we could follow a full derivation, but I don't think such a derivation is very illuminating. Instead, I think it's best to analyze the solution. Since the wave equation is so far idealized, it's not very complicated, and rather easy to understand with retrospect. So here is our numerical solution for the wave equation

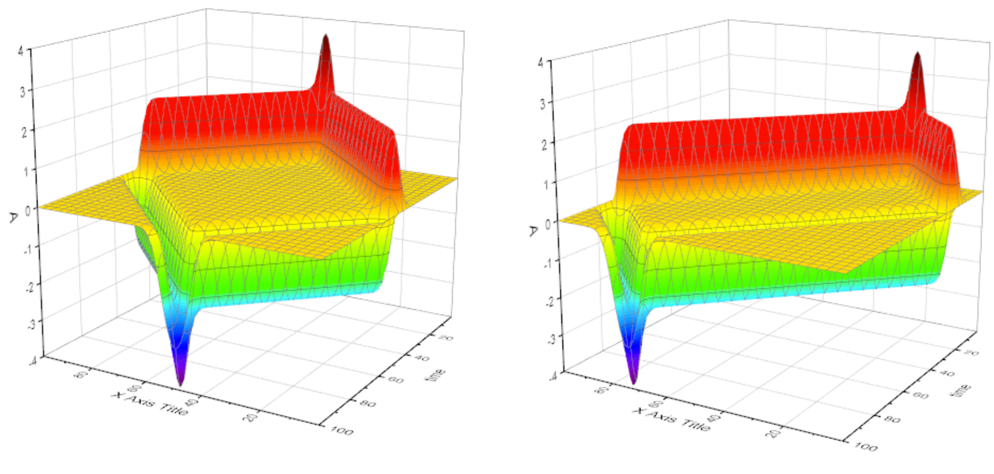
$$y(x_i, t_{i+1}) = y(x_{i+1}, t_i) + y(x_{i-1}, t_i) - y(x_i, t_{i-1})$$

The first thing to notice is that our wave is not a function of here and now $y(x_i, t_i)$, rather it's a function of here and *then* $y(x_i, t_{i+1})$. Of course the equation could equivalently be represented as a function of each term, but I believe the current form is the most instructive (it's the form we use in our simulation).

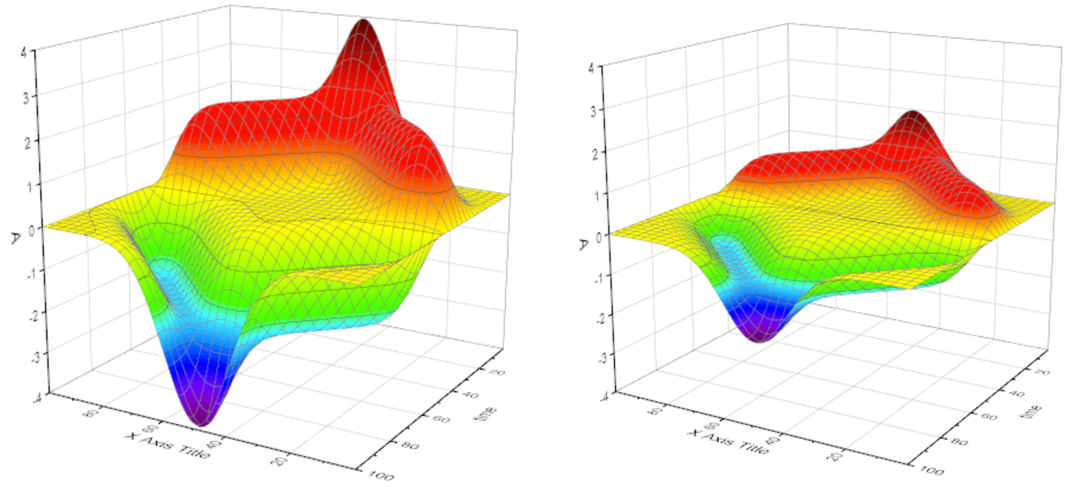
Basically, this equation tells us that, from the starting position, the next waveform will be the sum of that to the left and that to the right, minus the previous waveform. And since this is a second order equation, we require two initial conditions. The easiest way is to hold the initial Gaussian for the first 2 time steps. From here, the equation results in waves manifesting only along diagonals of a space-time diagram.



The above plot is only a representation, since realistically, subsequent waveforms should have half the area of the initial two, as conservation of information demands. Moreover, with fine enough time steps, we should see an initial splitting of the fist wave. This we can examine with the actual simulations, plotted below.

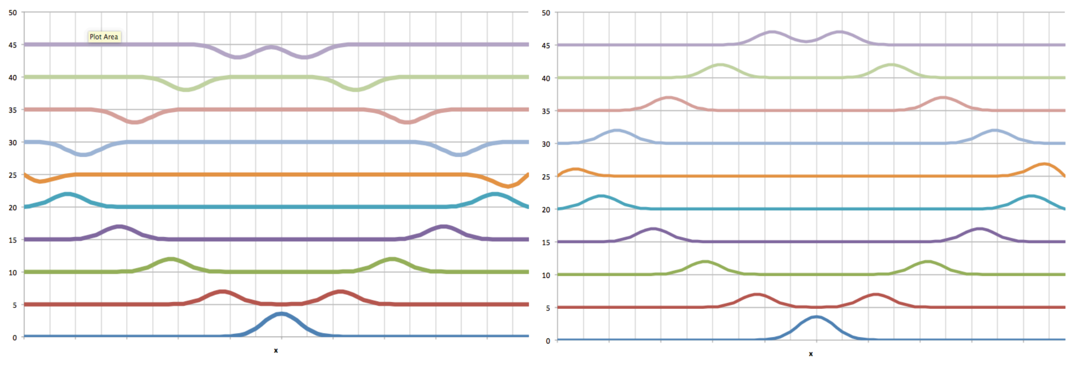


Above we see the back ends, with displacement plotted against time, where color represents amplitude $A$. To the left we see the initial Gaussian pluck bifurcate and bounce off the walls – flipping sign as they do. This is always what happens when we fix the end points of a string. To the right, we see the same dynamics but with a different starting position.



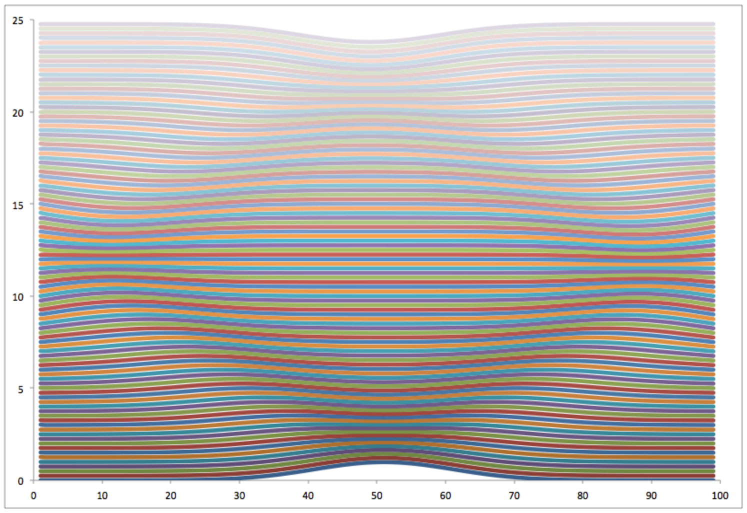
Here we see some very pretty plots, of what happens when we first increase the wavelength – decrease $\alpha$ (left), and then drop the amplitude (right).

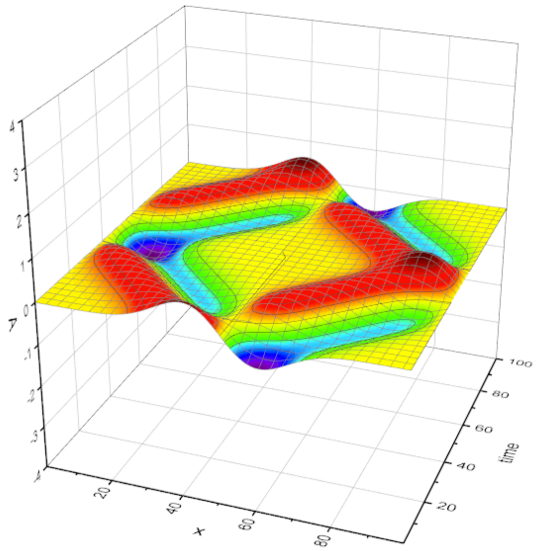
By staggering the time dimension, we can also see what happens in 2D.

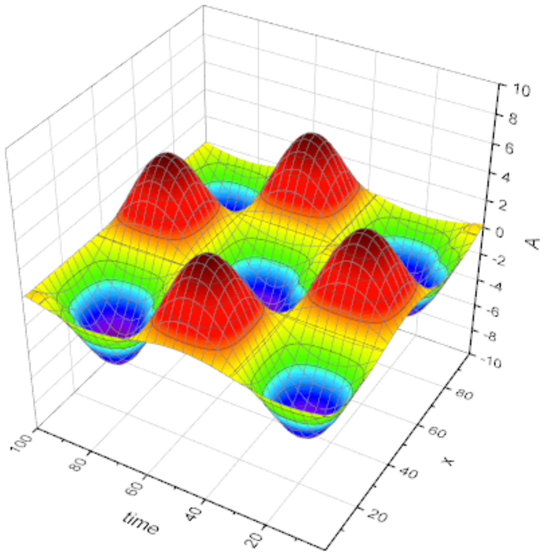


The left plot shows fixed boundary conditions, whereas the right plot shows 'loose' bounds, e.g., if the string were attached between poles and tied with loops which slid freely along the poles, as opposed to being pinned between fixed walls. We see that in this case the string doesn't reverse its phase, and is instead reflected as it came.

Below are exploratory plots, including what happens when you change your initial waveform to be either the derivative of a Gaussian, or a sinusoidal. The sin wave is of course not a wave pulse, but more of a continuous oscillation – like is the case for water waves. Below that is my C code implementation, which produced the plots.


25
20
15
10
5
0
0
10
20
30
40
50
60
70
80
90
100


4
3
2
1
0
-1
-2
-3
-4
A
20
40
60
80
x
20
40
60
80
100
time


10
8
6
4
2
0
-2
-4
-6
-8
-10
A
20
40
60
80
x
100
80
60
40
20
time

```
//Johnathan von der Heyde - 2019
//Simulates 1D wave propagation on string over time.
#include <stdio.h>
#include <math.h>
#define length 100
#define height 3

double w[length+1][height+1];

int main(){

   int x, y, t = 0;
   double amp = 0.5;
   double alpha = 0.1;  // 1 / wavelength or frequency
   double xi, x0 = 5;  //xi / starting position

 //initial conditions
   for (x=1; x<length; x++){
      xi = 0.1 * (x-1);
      //w[x][1] = amp*sin(alpha*(x));     //sinusoidal
      w[x][1] = amp*exp(-alpha*(pow((xi-x0),2)));  //Gaussian
      //w[x][1] = -amp*(xi-x0)*exp(-alpha*(pow((xi-x0),2)));  //derivative
      w[x][2] = w[x][1];
   }

 //boundary case
   w[0][2] = w[0][height] = 0.0;
   w[length][2] = w[length][height] = 0.0;

   FILE* f = fopen("wave", "w");

 //main loop
   while(t<100){

      for (x=1; x<length; x++)
         w[x][height] = w[x+1][2] + w[x-1][2] - w[x][1];

      for (x=1; x<length; x++){
         w[x][1] = w[x][2];
         w[x][2] = w[x][height];
      }
      for (x=1; x<length; x++)              // adjustible time extension
         fprintf(f, "%.6lf ", w[x][height]); // + (double)((double)t/4.0));

      fprintf(f, "\n");
      t++;
   }

   fclose(f);

   return 0;
}
```