Solving 1-D Wave Equation Using Finite Difference

Bhavay Luthra
Chirag Chand
Rahul Srivastava
Sheetal

Contents

Chapter 1

Introduction

In this article, we focus on the simple one-dimensional Wave Equation. We show how the equation can be solved using the finite difference method. We illustrate this by showing two solutions of the wave equation, a standing wave and a Gaussian pulse .By changing the general fortran formula , the code can also be applied to other fundamental one-dimensional equations such as Heat equation, and the Schrodinger equation.

1.1 Wave Equation

The wave equation is a second-order linear partial differential equation for the description of waves—as they occur in classical physics—such as mechanical waves (e.g. water waves, sound waves and seismic waves) or electromagnetic waves (including light waves). It arises in fields like acoustics, electromagnetism, and fluid dynamics.
The scalar wave equation is:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2} \right)$$

where c is a fixed non-negative real coefficient which describes the speed of the wave.
Using the notations of Newtonian mechanics and vector calculus, the wave equation can be written more compactly as:

$$\ddot{u} = c^2 \nabla^2 u$$

1.2 Finite difference method

In numerical analysis, finite-difference methods (FDM) are a class of numerical techniques for solving differential equations by approximating derivatives with finite differences. Both the spatial domain and time interval (if applicable) are discretized, or broken into a finite number of steps, and the value of the solution at these discrete points is approximated by solving algebraic equations containing finite differences and values from nearby points.

Finite difference methods convert ordinary differential equations (ODE) or partial differential equations (PDE), which may be nonlinear, into a system of linear equations that can be solved by matrix algebra techniques. Modern computers can perform these linear algebra computations efficiently which, along with their relative ease of implementation, has led to the widespread use of FDM in modern numerical analysis. Today, FDM are one of the most common approaches to the numerical solution of PDE, along with finite element methods.

Chapter 2

Wave Equation

Waves are everywhere. A plethora of physical phenomena can be thought of as waves. And we are not talking about the water waves we all created when we were kids by throwing rocks in the lakes — although these were pretty cool —. Light, for example, is an electromagnetic wave propagating through space Sound is also an example of a mechanical wave and, in particular, a pressure wave.

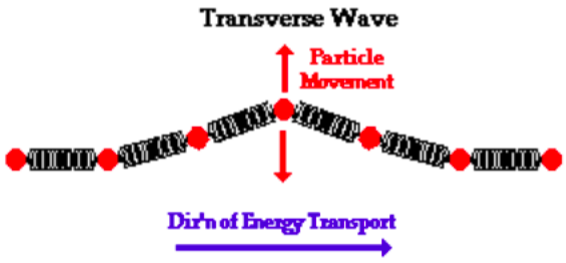
2.1 Waves in physics

Before we dive into the intricacies of the math behind the behavior of waves let's begin by asking ourselves a simpler question. *What is a wave?*

> *In physics, a wave is a propagating disturbance of one or more quantities that travels through a medium from one location to another. Wave motion transfers energy from one point to another, usually without permanent displacement of the particles of the medium.*

Mechanical waves, such as sound, require a medium through which to travel, while electromagnetic waves do not require a medium and can be propagated through a vacuum. Propagation of a wave through a medium depends on the medium's properties. Based on that, the speed at which a wave travels depends solely on the medium itself and not on the wave at all! It does not matter how hard or soft or at what angle you throw a rock in a lake. The waves will spread out at exactly the same speed.
A transverse wave propagating in a string/rope

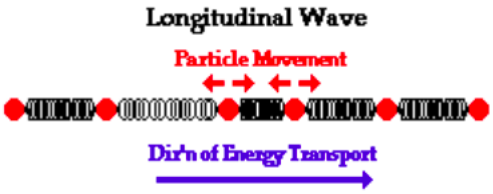
Figure 2.1: Particle movement is perpendicular to the propagation in a transverse wave



A longitudinal wave is a wave in which particles of the medium move in a direction parallel to the direction that the wave moves. An example of a longitudinal wave is a wave on a slinky as shown below.

A longitudinal wave propagating in a slinky

Figure 2.2: Particle movement is parallel to the propagation in a longitudinal wave



Finally, when two waves moving in opposite directions, having the same amplitude and frequency, are combined we get a standing wave — also known as a stationary wave :

A standing wave (black) as a result of two ways moving in opposite directions (red and blue)

Okay, we now know some of the fundamental ideas regarding waves in physics. We are ready to dive into the mathematics behind the equation that describes the behavior of any wave known as the wave equation.

2.2 The Wave Equation

The one-dimensional equation that describes how any kind of disturbance propagates through a medium i.e. the wave equation is the following:

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

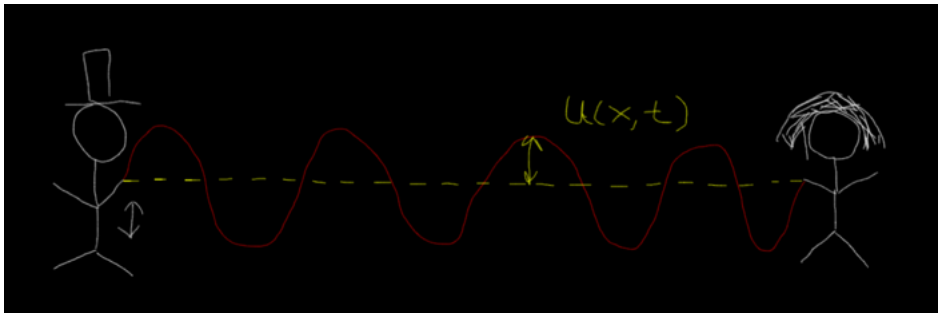
where $\mathbf{u} = \mathbf{u}(\mathbf{x}, \mathbf{t})$ is the displacement of the particle of the wave located at position x at time t and c is the speed at which the wave is propagating.

We will not present a formal mathematical proof of the aforementioned equation in this project. Instead, we will try to present an intuitive interpretation of what it represents. Let's begin!

2.3 Developing Intuition

Suppose that two friends, Alice and Bob are playing with a string. Alice is holding one end of the string while Bob is holding the other. They are both moving their hands up down trying to create some sort of wave in that string.

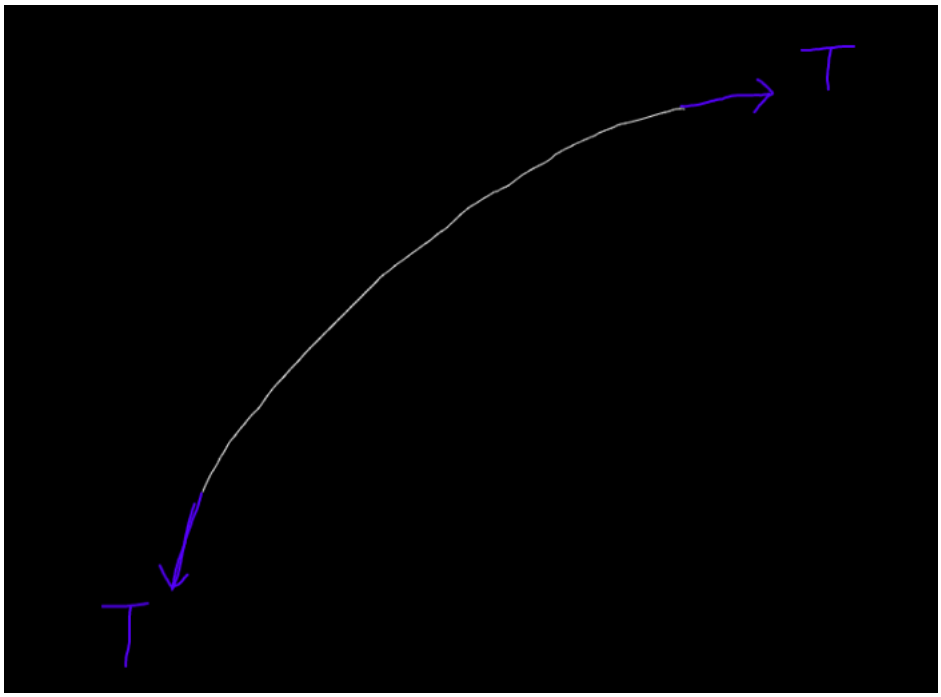
Figure 2.3: Bob and Alice playing with a string



We now take our binoculars and we zoom in in that string trying to figure out what exactly is going

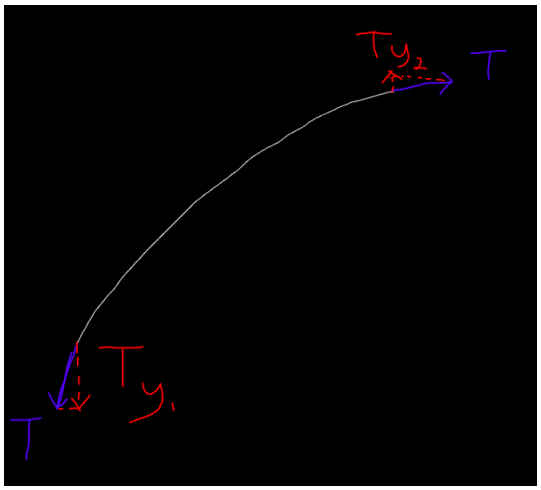
on. Bob and Alice create a wave with their hand motion which propagates through the string. In order for that to happen, a force must be exerted to the particles of the string which constantly move up and down. That force is tension.

Figure 2.4: An element of the string



The tension acts tangent to the direction of the string it pulls the string apart as seen in the image above. For simplicity purposes, we will assume that the tension is constant throughout the string — which can be easily proven for small disturbances .

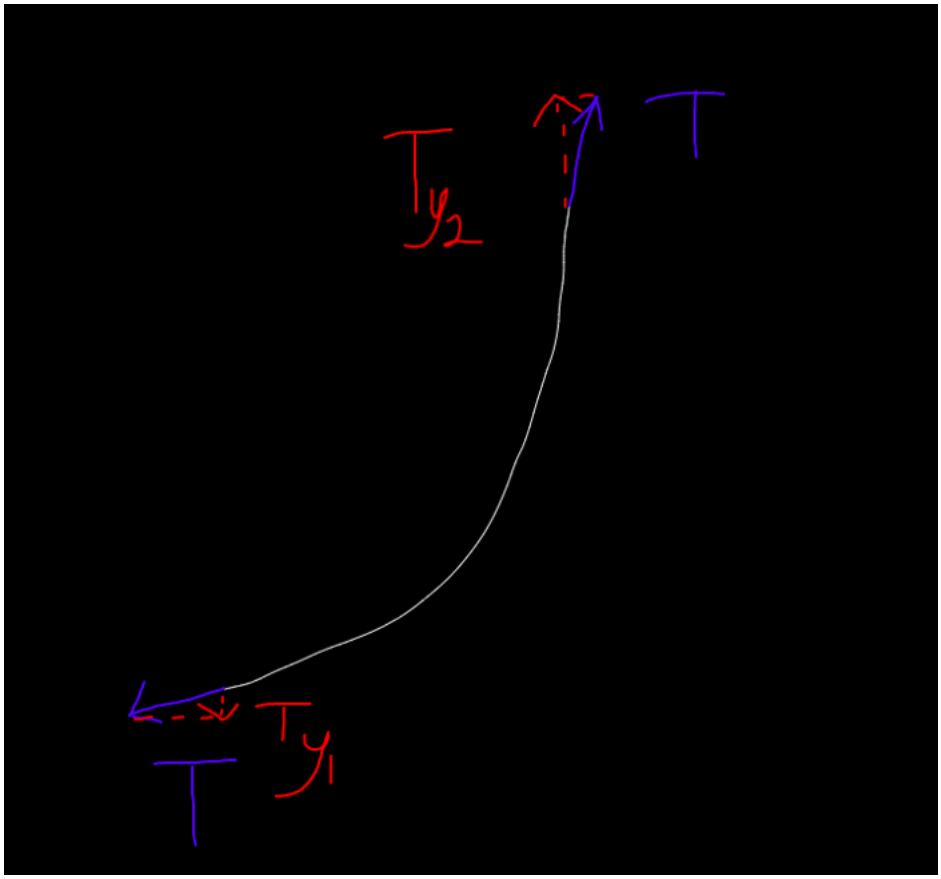
Now, notice that the part of the string that we zoomed in is curved downwards. The vertical components of both of the tensions at the end of the string can be seen in the image below.



It is obvious that $Ty_1 > Ty_2$ and thus, the net tension force is directed downwards. On the other hand, if we had zoomed in on another element of the string that was curved the other way, the net vertical

tension would be directed upwards.

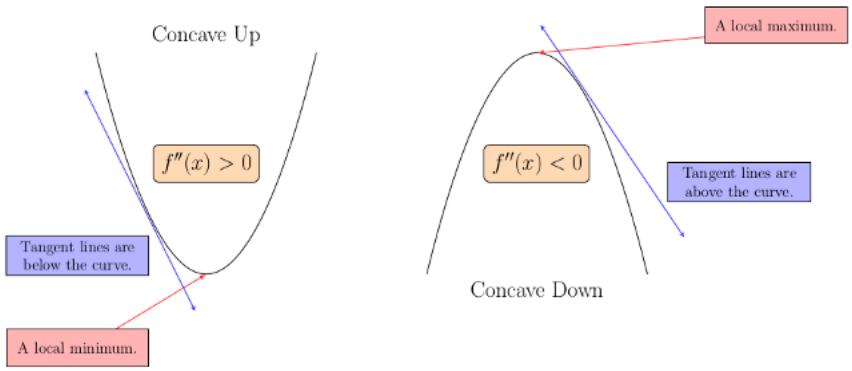
Figure 2.5: $Ty_1 > Ty_2$ The net vertical tension is directed upwards



We can conclude that the concavity of the string i.e. the way it is curved is related to the net tension applied to the string.

As we know from calculus, the concavity of a function is indicated by the sign of its second derivative. If the function's second derivative is positive, the graph concaved up while if it's negative, the graph is concaved down.

Figure 2.6: Concavity of a function f = f(x)

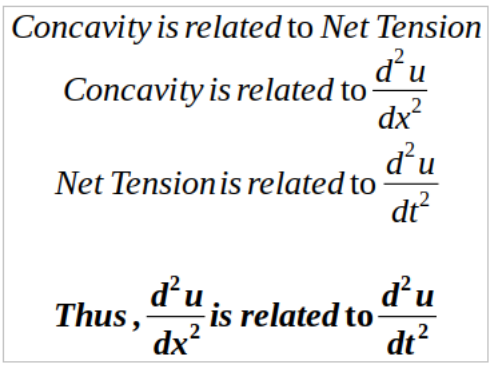


From Newton's second law, we can relate the net vertical tension to the acceleration of the string i.e. the second time derivative of its vertical displacement

$$\Sigma F = ma = m\frac{d^2u}{dt^2}$$

With all this in mind, it is evident that the second spatial derivative of the vertical displacement of the string is related to its second time derivative. To summarize:

Figure 2.7



It can be easily proven mathematically that not only are these two quantities related but they are proportional to each other. If we name the proportionality constant c^2 then we arrive at the wave equation!

2.4 Solution

If we now perform some factorization in the wave equation we can reach the following form:

$$\frac{d^2u}{dt^2} = c^2\frac{d^2u}{dx^2} \rightarrow c^2\frac{d^2u}{dx^2} - \frac{d^2u}{dt^2} = 0 \rightarrow (c\frac{du}{dx} - \frac{du}{dt})(c\frac{du}{dx} + \frac{du}{dt}) = 0$$

We can see that $u(x,t) = u(x-ct)$ and $u(x,t) = u(x+ct)$ are solutions respectively. The point here is that the wave equation describes any disturbance that has a left and right component. Since c is positive, the term $x - ct$ becomes smaller as time passes while $x + ct$ becomes larger.

Thus, whatever u describes, the term $u(x-ct)$ corresponds to a disturbance moving to the right while the term $u(x+ct)$ corresponds to one moving to the left. Thus, the equation for a standing wave can easily be produced by adding two solutions moving in opposite directions $u = u(x-ct) + u(x+ct)$ which have the same amplitude and frequency.

Chapter 3

Finite Difference Method

To solve the 1-D wave equation we have a couple of methods like Method of lines , Finite element method, Gradient discretization method , etc but in this docment we will be using the Finite difference method solve out 1-D wave equation.
In this method, functions are represented by their values at certain grid points and derivatives are approximated through differences in these values.

3.1 Finite Difference Method

This method use the neat property of Taylor ploynomial to calculate the derivate of a function by using taylor series. Thus we can calculate the nth order deivate with n intial condition.

Recall that the derivative of a function was defined by taking the limit of a difference quotient:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \rightarrow (8.1)$$

Now to use the computer to solve differential equations we go in the opposite direction - we replace derivatives by appropriate difference quotients. If we assume that the function can be differentiated many times then Taylor's Theorem is a very useful device to determine the appropriate difference quotient to use. For example, consider:

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f^{(2)}(x_0)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}h^n + R_n(x) \rightarrow (8.2)$$

Re-arranging terms in (8.2) and dividing by Δx we obtain:

$$\frac{f(a+h)}{h} = \frac{f(a)}{h} + f'(a) + \frac{R_1(x)}{h}$$

Solving for $f'(a)$:

$$f'(a) = \frac{f(a+h) - f(a)}{h} - \frac{R_1(x)}{h}$$

ssuming that $R_1(x)R_1(x)$ is sufficiently small, the approximation of the first derivative of "f" is:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

This is, not coincidentally, similar to the definition of derivative (eqn 8.1).

3.2 1-D Wave Equation using FDM

First we define the Shortcut Notation:

$$u_t = \frac{\partial u}{\partial t}$$

$$u_{xx} = \frac{\partial^2 u}{\partial x^2}$$

Then our 1-D wave equation can be written as :

$$u_{tt} = c^2 u_{xx}$$

3.2.1 Formulae Used

1. First Derivates:

$$\text{Forward Difference} : f'(x) \approx \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$
$$\text{Backward Difference} : f'(x) \approx \lim_{\Delta x \to 0} \frac{f(x) - f(x+\Delta x)}{\Delta x}$$
$$\text{Central Difference} : f'(x) \approx \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x-\Delta x)}{\Delta x}$$

2. Second Derivates:

$$\text{Central Difference} : f''(x) \approx \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x) + f(x-\Delta x)}{2\Delta x}$$

3. First Order Derivates:

$$\text{Forward Difference} : u_t(x,t) \approx \lim_{\Delta t \to 0} \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t}$$
$$\text{Backward Difference} : u_x(x,t) \approx \lim_{\Delta x \to 0} \frac{u(x,t) - u(x-\Delta x,t)}{\Delta x}$$
$$\text{Central Difference} : u_t(x,t) \approx \lim_{\Delta t \to 0} \frac{u(x,t+\Delta t) - u(x,t-\Delta t)}{2\Delta t}$$

4. Second Order Derivates:

$$\text{Central Difference} : u_t(x,t) \approx \lim_{\Delta t \to 0} \frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{(\Delta t)^2}$$
$$: u_t(x,t) \approx \lim_{\Delta x \to 0} \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2}$$

3.3 Boundary & Initial Conditions

- $u_{tt} = c^2 u_{xx}$ $0 < x < L$
- Boundary Conditions :$u(0,t) = 0$, $u(L,t) = 0$
- Initial Conditions: $u(x,0) = f(x)$, $u_t(x,0) = g(x)$

3.3.1 Solving

$$u_{tt} = c^2 u_{xx}$$
$$x_{i+1} = x_i + \Delta x$$
$$t_{j+1} = t_j + \Delta t$$
$$\frac{u(x_i,t_{j+1}) - 2u(x_i,t_j) + u(x_i,t_{j-1})}{(\Delta t)^2} = c^2 \frac{u(x_{i+1},t_j) - 2u(x_i,t_j) + u(x_{i-1},t_j)}{(\Delta x)^2}$$
$$u(x_i,t_{j+1}) = 2y(x_i,t_j) + \lambda * (y(x_{i+1},t_j) - 2y(x_i,t_j) + y(x_{i-1},t_j)) - y(x_i,t_{j-1})$$
$$\text{here: } \lambda = c^2 \frac{\Delta t^2}{\Delta x^2}$$

Chapter 4

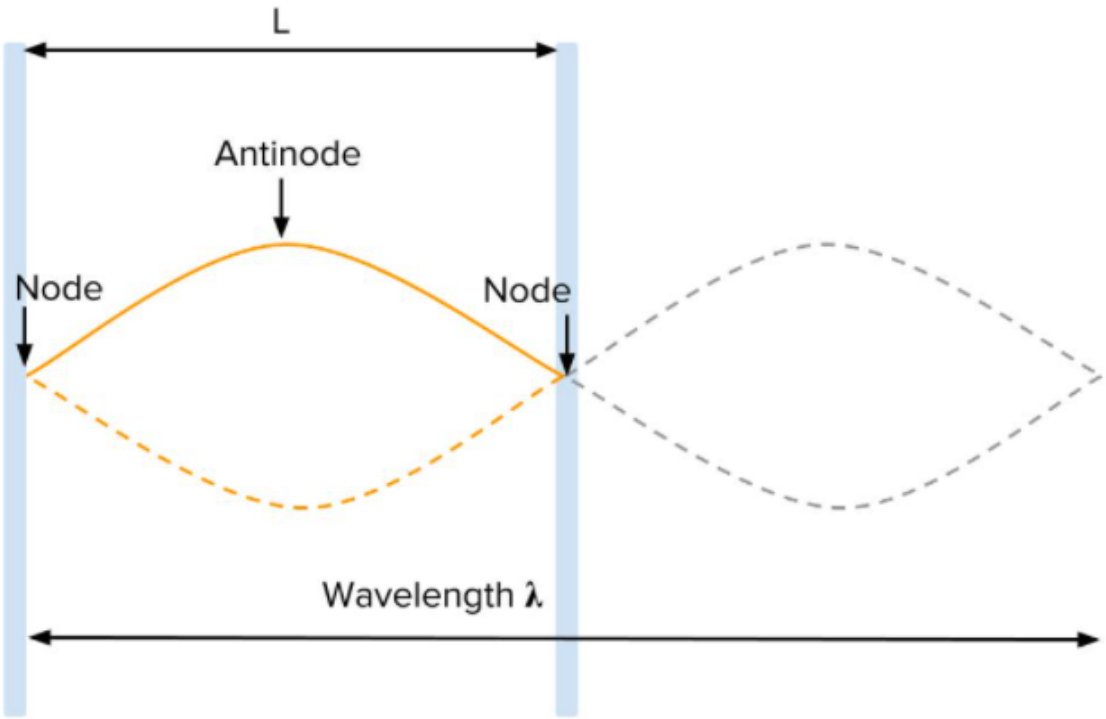
Standing Wave

In physics, a standing wave, also known as a stationary wave, is a wave that oscillates in time but whose peak amplitude profile does not move in space. The peak amplitude of the wave oscillations at any point in space is constant with respect to time, and the oscillations at different points throughout the wave are in phase. The locations at which the absolute value of the amplitude is minimum are called nodes, and the locations where the absolute value of the amplitude is maximum are called antinodes.

General Equation :

$$y(x,t) = 2y_{\max} \sin\left(\frac{2\pi x}{\lambda}\right) \cos(\omega t)$$

Figure 4.1: For the fundamental frequency of a standing wave between two fixed ends, the wavelength is double the length of the string.



4.1 Fortran Code

```
!Program is to calculate the postion of a wave in space and time using 1-D wave equation
!u_tt = A^2 u_xx, where u(x,t) is a function of x and t that tells the dispacement of wave
at postion x and time t
!The program calculating the wave genrated by a tightly streched string at both ends of
length L
!Also it is assumed that at time t = 0 the wave is at equilibrium
program wave
        real,parameter :: PI = 3.14159
        real :: string_length, time_end, x_step, t_step, phase_vel ,lambda
        real :: wave_vector, ang_vel

        !Here string_length is the ending point of string i.e. the length of string
        !time_end is the final time
        !x_step is the step size of x
        !t_step is the step size of t
        !phase_vel is the constant in the 1D wave equation

        integer :: nt, nx
        !nt and nx are the number of steps in t and x respectively

        real, dimension(:,:), allocatable :: y
        !y will be containing the the displacement of the x at time t

        real, dimension(:,:), allocatable :: ya
        !ya is the analytic solution: ya(x,t) = sin(kx)cos(wt)

        integer :: i, j
        !just two variable to move through the matrix

        write(*, '("The following program is to calculate the position of a wave genrated
by a streched string is space-time")')
        write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^2 u_xx", /, /)')
        write(*, '("Enter the length of the string : ")', advance = "no")
        read *, string_length
        write(*, '("Enter the final time t : ")', advance = "no")
        read *, time_end
        write(*, '("Enter the step size in x : ")', advance = "no")
        read *,x_step
        write(*, '("Enter the step size in t : ")', advance = "no")
        read *, t_step
        write(*, '("Enter the constant A : ")', advance = "no")
        read *, phase_vel

        ! Calculate the number of time and distance points
        nt = time_end/t_step + 0.5
        nx = string_length/x_step + 0.5
        print *, "nx = ", nx, ", nt = ", nt

        ! Wave vector and angular velocity
        ! We are assuming the wave is the fundamental so the wavelenght is 2L , k =
2pi/lambda , 2L/n = lambda , k = nPI/L :
        wave_vector = 7*PI/string_length
        ang_vel = phase_vel*wave_vector
        print *, "wave vector = ", wave_vector, ", angular velocity = ",ang_vel

        !Make the sure the value is 1 or less than 1 otherwise it will shoot to infinity
        lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
        print *,"Lambda = ", lambda

        ! This does the analytic calculation
        allocate(ya(0:nx, 0:nt))
        do i = 0, nx
            do j = 0, nt
               ya(i, j) = sin(wave_vector*i*x_step)*cos(ang_vel*j*t_step)
            end do
        end do

        ! Write the analytic data
```

```
        open(10, file = "wave_a.txt")
        do j = 0, nt
            do i = 0, nx
                write(10,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, ya(i,j)
            end do
            write(10, '(/)')
        end do
        close(10)

        ! Now the numerical calculation
        !the 1 is added to both nt and nx because the initial points are also needed to be
mapped
        !as the string is streched from the both ends and initially the string is at
equilibrium thus u(0,t) = u(L,t) = 0
        !this is boundry value
        allocate(y(0:nx, 0:nt))

        do j = 0, nt
            y(0,j) = 0
            y(nx, j) = 0
        end do

        !set the 1st initial values: y(x,0) = sin(kx)
        do i = 1, nx - 1
            y(i,0) = sin(wave_vector*i*x_step)
        end do

        !set the 2nd initial values: y(x,t) = sin(wt + kx)
        do i = 1, nx-1
            y(i,1) = sin(wave_vector*i*x_step)*cos(ang_vel*t_step)
        end do

        !calculating the value of u(x,t)
        do j = 1, nt-1
            do i = 1, nx-1
              y(i,j+1) = 2*y(i,j) +  lambda*(y(i+1,j)-2*y(i,j) + y(i-1,j)) - y(i,j-1)
            end do
        end do

        open(19, file = "wave.txt")
        do j = 0, nt
            do i = 0, nx
                write(19,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, y(i,j)
            end do
            write(19, '(/)')
        end do

        ! All finished so free the arrays we allocated
        deallocate(ya)
        deallocate(y)

    end program wave
```

4.2 Gnu script

Figure 4.2: Snapshot of the GNU Script

waveequation.plt

```
set terminal gif animate delay 15
set output "wave.gif"
stats "wave.txt" name "wave"
set xrange[wave_min_x:wave_max_x]
set yrange[wave_min_y:wave_max_y]
set xlabel "Length(x)"
set ylabel "Displacement(u)"
do for[i=0:wave_blocks-2]{plot "wave.txt" index i w l title sprintf("time = %.3f", i*0.005)}

set terminal gif animate delay 15
set output "wave_a.gif"
stats "wave_a.txt" name "wave"
set xrange[wave_min_x:wave_max_x]
set yrange[wave_min_y:wave_max_y]
set xlabel "Length(x)"
set ylabel "Displacement(u)"
do for[i=0:wave_blocks-2]{plot "wave_a.txt" index i w l title sprintf("time = %.3f", i*0.005)}
```

The first para creates the file wave.gif which is generated from the points in the wave.txt. This is the numerical solution.
The Second para creates the file wave_a.gif which is generated from the points in the wave_a.gif . This is the analytical solution.

4.3 Ouputs

4.3.1 Analytical & Numerical

We first run the fortran code to generate the text file. Input as follows:

Figure 4.3: Input for the fortran code

```
> Executing task: C:\Users\bhavay\OneDrive - Keshav Mahavidyalaya\Documents\Study\KMV\Phy\Sem 4\SEC\fortran\wave.exe  <

The following program is to calculate the position of a wave genrated by a streched string is space-time
Using 1-D Wave Equation
              u_tt = phase_vel^2 u_xx


Enter the length of the string : 1
Enter the final time t : 2
Enter the step size in x : 0.01
Enter the step size in t : 0.01
Enter the constant A : 1
 nx =          100 , nt =          200
 wave vector =    21.9911308     , angular velocity =    21.9911308
 Lambda =    1.00000000

Press any key to close the terminal.
```

Next we run the gnu script file to create the two gif files. To look at the actual solution , I have uploaded the gif files on imgur. Here , I only will be post the stills from the two gif.

Figure 4.4: Still from the Numerical solution

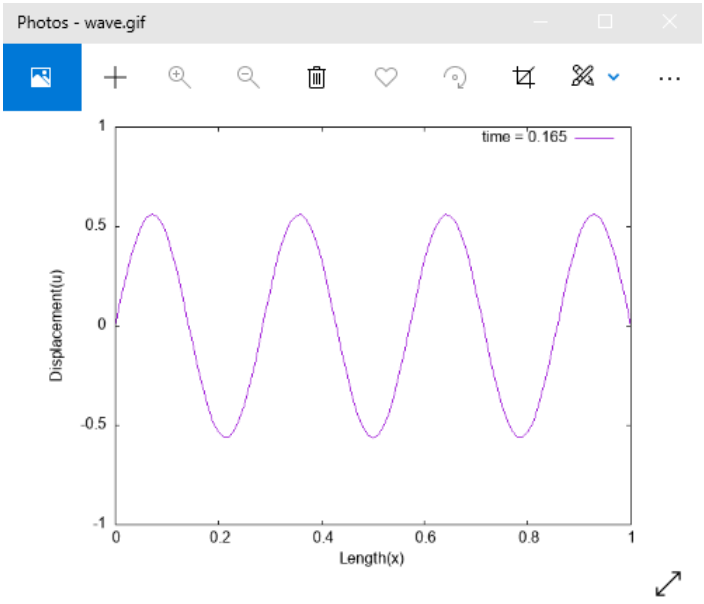
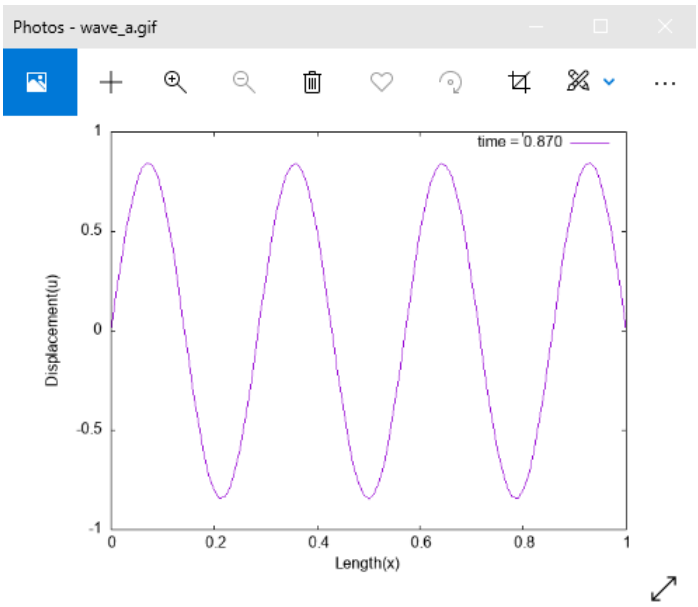


Figure 4.5: Still from the analytical solution



Here is the link for the actual gifs:
Analytical Solution
Numerical Solution

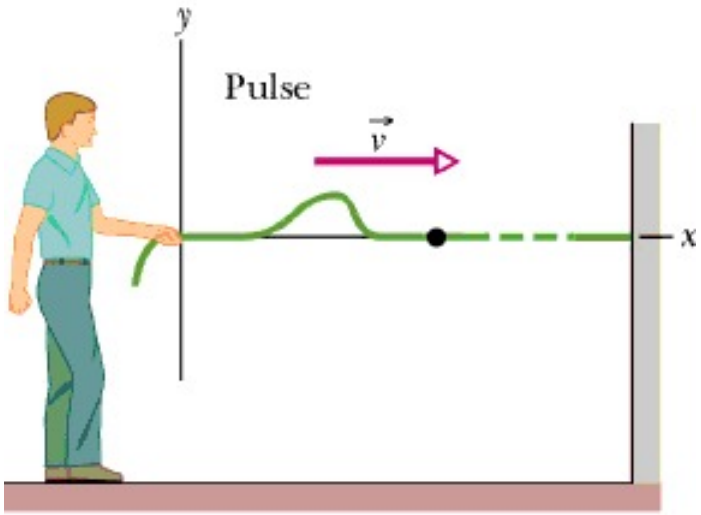
BONUS!!
I also uploaded the text file on the google drive which is generated from running fortran code.
Analytical Solution
Numerical Solution

Chapter 5

Pulse Wave

In chapter 4 we discussed using the wave equation to model a standing wave. This is a particularly simple solution since the standing wave is formed from a single sinusoidal wave reflected from the ends of the string, and it admits a simple analytical solution that can be compared with the numerical calculation. However we can use the finite difference method to calculate the time evolution of more complex waves, and to illustrate this in this chapter we consider a Gaussian pulse moving along the string. This will be familiar to many of us as the wave we observe when flicking the end of a rope.

Figure 5.1: Plot of a harmonic wave at t = 0 and t = Δt.



The equation for the pulse is:

$$y(x,t) = A\exp(-(x-vt)^2/a^2)$$

where A is the peak amplitude, v the wave velocity and a a constant that describes the pulse width. This pulse moves in the positive x direction at velocity v.

For the numerical solution we need only change the initial conditions. We set $y(x,0)$ and $y(x,dt)$, where dt is the time increment, using the equation above, and as before the ends of the string are clamped so the wave reflects from them. The code is given in the first part of section 5.1.

This time the analytic solution is more complex as we need to reflect the wave from the ends of the string and sum the incident and reflected pulses. Due to this added complexity the analytic solution has been done in a separate program given in the second part of section 5.1.

Sections 5.3 onwards give the results and associated discussion.

5.1 Fortran Code

Numerical

```
! This version of the program models a gaussian pulse travelling along the
string
program wave
    real :: string_length, time_end, x_step, t_step, phase_vel ,lambda
    real :: init_pos, width
    !Here string_length is the ending point of string i.e. the length of
string
    !time_end is the final time
    !x_step is the step size of x
    !t_step is the step size of t
    !phase_vel is the constant in the 1D wave equation
    integer :: nt, nx
    !nt and nx are the number of steps in t and x respectively
    real, dimension(:,:), allocatable :: y
    !y will be containing the the displacement of the x at time t

    integer :: i, j
    !just two variable to move through the matrix

    write(*, '("The following program is to calculate the position of a
wave genrated by a streched string is space-time")')
    write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^2
u_xx", /, /)')
    write(*, '("Enter the length of the string : ")', advance = "no")
    read *, string_length
    write(*, '("Enter the final time t : ")', advance = "no")
    read *, time_end
    write(*, '("Enter the step size in x : ")', advance = "no")
    read *,x_step
    write(*, '("Enter the step size in t : ")', advance = "no")
    read *, t_step
    write(*, '("Enter the constant A : ")', advance = "no")
    read *, phase_vel
    ! Calculate the number of time and distance points
    nt = time_end/t_step + 0.5
    nx = string_length/x_step + 0.5
    print *, "nx = ", nx, ", nt = ", nt
    !Make the sure the value is 1 or less than 1 otherwise it will shoot
to infinity
    lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
    print *,"Lambda = ", lambda
    ! Now the numerical calculation
    allocate(y(0:nx, 0:nt))
    ! The ends have to be at 0 or all times
    do j = 0, nt
        y(0, j) = 0
        y(nx, j) = 0
    end do
    ! Set the 1st initial values: y(x,0) = exp(-(x - x0)^2/a^2)
    ! We will start with x0 = L/2 i.e. halfway along the string
    init_pos = string_length/2
    width = string_length/5
    do i = 1, nx - 1
```

```
        y(i,0) = exp(-(i*x_step - init_pos)**2/width**2)
    end do
    ! For the second initial values the pulse has moved a distance v dt
along the string
    init_pos = init_pos + phase_vel*t_step
    do i = 1, nx-1
        y(i,1) = exp(-(i*x_step - init_pos)**2/width**2)
    end do
    ! Now do the calculation
    do j = 1, nt-1
        do i = 1, nx-1
          y(i,j+1) = 2*y(i,j) +  lambda*(y(i+1,j) - 2*y(i,j) + y(i-1,j)) -
y(i,j-1)
        end do
    end do
    open(19, file = "wavepulse.txt")
    do j = 0, nt
        do i = 0, nx
            write(19,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, y(i,j)
        end do
        write(19, '(/)')
    end do
    ! All finished so free the arrays we allocated
    deallocate(y)
end program wave
```

Analytical

```
program ana
    implicit none
    real :: string_length, time_end, x_step, t_step, phase_vel,
lambda, initial_pos, width, g
    common initial_pos, width, phase_vel, string_length
    integer :: nt, nx, i, j
    real, dimension(:,:), allocatable :: ya

    write(*, '("The following program is to calculate the position
of a wave genrated by a streched string is space-time")')
    write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^
2 u_xx", /, /)')
    write(*, '("Enter the length of the string : ")', advance =
"no")
    read *, string_length
    write(*, '("Enter the final time t : ")', advance = "no")
    read *, time_end
    write(*, '("Enter the step size in x : ")', advance = "no")
    read *,x_step
    write(*, '("Enter the step size in t : ")', advance = "no")
    read *, t_step
    write(*, '("Enter the constant A : ")', advance = "no")
    read *, phase_vel

    initial_pos = string_length/2
    width = string_length/5
```

```
    nt = time_end / t_step + 0.5
    nx = string_length / x_step + 0.5
    lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
    allocate(ya(0:nx, 0:nt))
    open(19, file = "wavepulse_a.txt")
    !Analytical Methos
    do j = 0, nt
        write(19,'(a,f0.4)') "# Time: ", j*t_step
        do i = 0, nx
            ya(i,j) = g(i*x_step, j*t_step)
            write(19, '(F0.4, 2X, F0.4)') i*x_step, ya(i,j)
        end do
        write(19,'(/)')
    end do
    close(19)

    deallocate(ya)
end program ana
real function g(x,t)
    implicit none
    real :: x, t
    real :: x2
    real :: initial_pos, width, phase_vel, string_length
    common initial_pos, width, phase_vel, string_length
    g = 0
    ! This is the right moving pulse
    x2 = -x + t*phase_vel + initial_pos
    ! x2 needs to be in the range -string_length < x2
< +string_length
    ! to simulate an infinite train of pulses
    x2 = mod(x2 + string_length, 2*string_length) - string_length
    g = g + exp(-(x2/width)**2)
    ! And we subtract the left moving pulse
    x2 = x + t*phase_vel - initial_pos - string_length
    ! x2 needs to be in the range -string_length < x2
< +string_length
    ! to simulate an infinite train of pulses
    x2 = mod(x2 + string_length, 2*string_length) - string_length
    g = g - exp(-(x2/width)**2)
    return
end function g
```

5.2 GNU Script

5.2.1 Analytical

Figure 5.2: Snapshot of the GNU Script (Analytical)

wavepulse_analytical.plt

```
set terminal gif animate delay 5 optimize
set output "wavepulse_a.gif"
stats "wavepulse_a.txt" name "wavepulse_a"
set xrange[wavepulse_a_min_x:wavepulse_a_max_x]
set yrange[wavepulse_a_min_y:wavepulse_a_max_y]
set xlabel "Length(x)"
set ylabel "Displacement(u)"
do for[i=0:wavepulse_a_blocks-2]{plot "wavepulse_a.txt" index i w l title sprintf("Step %d", i)}
```

5.2.2 Numerical

Figure 5.3: Snapshot of the GNU Script (Numerical)

wavepulse.plt

```
set terminal gif animate delay 5 optimize
set output "wavepulse.gif"
stats "wavepulse.txt" name "wavepulse"
set xrange[wavepulse_min_x:wavepulse_max_x]
set yrange[wavepulse_min_y:wavepulse_max_y]
set xlabel "Length(x)"
set ylabel "Displacement(u)"
do for[i=0:wavepulse_blocks-2]{plot "wavepulse.txt" index i w l title sprintf("time = %.3f", i*0.005)}

```

5.3 Output

We first run the fortran code to generate the text file. Input as follows:

Figure 5.4: Input for the fortran code

```
Enter the length of the string : 1
Emter the final time : 2
Enter the constant A : 1
Enter the x step size : 0.001
Enter the t step size : 0.001

Press any key to close the terminal.
```

Next we run the gnu script file to create the two gif files. To look at the actual solution , I have uploaded the gif files on imgur. Here , I only will be post the stills from the two gif.

Here is the link for the actual gifs:
Analytical Solution
Numerical Solution

Figure 5.5: Still from the Numerical solution

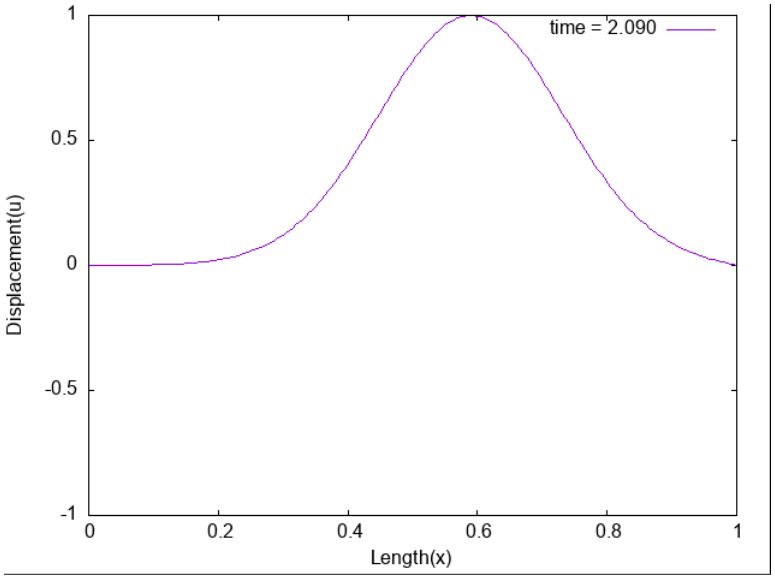
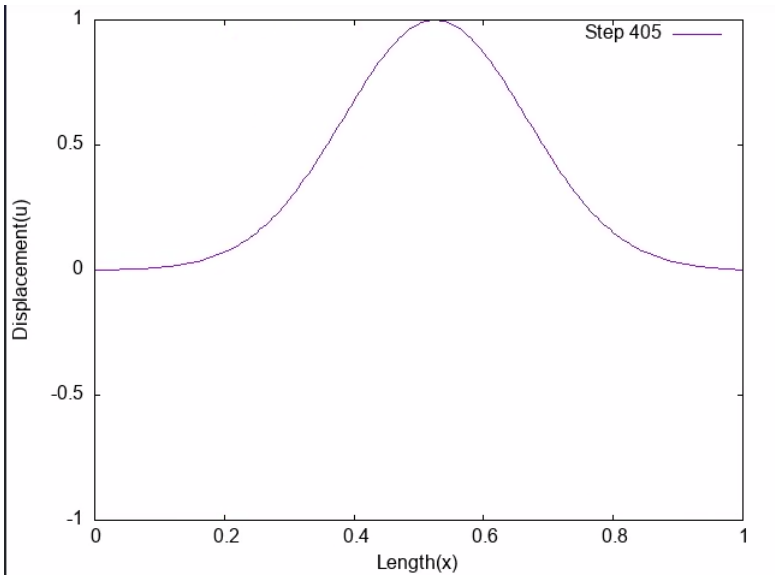


Figure 5.6: Still from the analytical solution



BONUS!!
I also uploaded the text file on the google drive which is generated from running fortran code.
Analytical Solution
Numerical Solution