Waves on a string including simulation of the propagation of a wave along a string and standing wave:

Student number: 40291098
dgrant12@qub.ac.uk
10/11/23

Introduction:

The phenomenon of waves travelling along a string has been a subject of fascination and study for centuries. From the simple observation of a plucked guitar string to the intricate mathematics that describes such motion, understanding waves on a string is fundamental in both physics and various applications, it offers deep insights into the underlying principles of wave mechanics.

The behaviour of waves on a stretched string depends on numerous factors, from the string's tension and mass density to the nature of the external disturbance that initiates the wave. Under idealized conditions, with no external forces acting, the string can support simple harmonic waves. However, real-world scenarios often involve complications. One significant factor that affects wave propagation is the presence of external damping forces, such as air resistance, which leads to the attenuation of the wave over time.

While the fundamental principles governing waves on strings have been known for centuries, deriving an exact solution, especially in non-ideal situations, remains challenging. With the advent of computational methods, however, it has become feasible to study these waves' behaviour under a variety of conditions. Using techniques like the Fast Fourier Transform (FFT) and numerical simulations, one can delve deeper into understanding both the frequency components and the time evolution of waves on a string.

In essence, the objective is to understand, simulate, and compare the behaviour of waves on a string through exact mathematical derivations and numerical approximations, shedding light on both the elegance of wave physics and the power of computational methods. Through a combination of theory and simulation, our aim is to paint a comprehensive picture of wave behaviour on a string.

Physics / Method:

1. ***Wave Equation:*** For all models/programs written there are multiple similarities between them, firstly, this loop models wave propagation on a string using the finite difference method, a numerical technique to solve differential equations. Applying the method iteratively, calculating the new displacement for each string segment, excluding the fixed ends which remain stationary.

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

Eqn 1: The wave equation in one space dimension

```
for t in range(time_steps):
  for i in range(1, num_elements - 1):
    y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
  y_old = y.copy()
  y = y_new.copy()
```

2. ***Boundary conditions*** are critical in wave mechanics as they determine how waves behave at the boundaries of a medium, such as a string. They influence:
    - *Reflection and Transmission:* Dictate whether waves reflect into the medium or pass into another, affecting wave magnitude and phase.
    - *Standing Waves and Resonance:* When waves reflect off boundaries under certain conditions, they can interfere with incoming waves. This interference can lead to the formation of standing waves, where the wave appears to be stationary with points of no motion (nodes) and points of maximum motion (antinodes). If driven at the right frequency, this can result in resonance,
    - *Physical Realism*: Ensure simulations reflect real-world scenarios, like a string fixed or freely hanging at the end.
    - *Solution Stability:* Essential for numerical simulation stability and avoiding unphysical results.

    This setup corresponds to Dirichlet boundary conditions with zero displacement, effectively simulating a string held in place at its endpoints. The computational loop that updates the displacements of the string's elements deliberately avoids altering these boundary elements, thereby maintaining the fixed-end constraints at each timestep of the simulation.

3. ***Determining System Response***: The initial shape or displacement of the string dictates how the system will respond over time. A string with no initial displacement remains static, while one with an initial perturbation will display wave-like behaviour, transferring energy from one point to another.

```
def initial_displacement(x):
    # Gaussian displacement with maximum of 5mm at the centre
    return 0.005 * np.exp(-40 * (x - L/2)**2)

# Spatial discretization
dx = L / num_elements

# Arrays for displacements
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# Set the initial displacement
y += initial_displacement(np.linspace(0, L, num_elements))
y_old += y
```

4. ***Determining Numerical Stability:*** The numerical simulation adhered to the Courant-Friedrichs-Lewy (CFL) condition, ensuring stability by satisfying the criterion.

$$C = c\frac{\Delta t}{\Delta x} \leq 1$$

Eqn 2: CFL Condition: If the condition < 1 the simulation is likely stable (CFL condition is satisfied)

Unstructured changes:

1) <u>*When simulating real-world scenarios*</u>, such as a plucked guitar string or the ripple effect of a stone dropped in water, the initial conditions help recreate these situations authentically, ensuring that the simulation mirrors reality. To simulate a more realistic pluck of a string we change the initial displacement here set up a triangular profile for the string, which resembles the shape the string would take if it were plucked at a point along its length.

```
# For the middle pluck scenario
d0 = 0.01      # 1 cm displacement
d0_loc = 0.5   # For middle pluck

# Set the initial displacement
y[np.where(x <= d0_loc*x[-1])] = d0/(d0_loc*x[-1]) * x[np.where(x <= d0_loc*x[-1])]
y[np.where(x > d0_loc*x[-1])] = -d0/((1.0-d0_loc)*x[-1]) * (x[np.where(x > d0_loc*x[-1])] - x[-1])
```

2) To find the effect of having sting ends which are free to move up and down in the y direction.

```
# Free boundary condition (add to the for loop)
y_new[0] = y_new[1]
y_new[-1] = y_new[-2]
```

3) Behaviour of a whip (T (tension) and μ can vary along the string)

```
# Variable tension and mass per unit length along the string
tension = np.linspace(100, 10, num_elements)  # Tension decreasing from base to tip
mu = np.linspace(0.01, 0.02, num_elements)    # Mass per unit length increasing from base to tip
```

4) FFT (Fast Fourier Transform)

```
from scipy import fft

# FFT analysis using scipy
frequencies = fft.fftfreq(time_steps, dt)
positive_freq_idxs = np.where(frequencies > 0)  # Only consider positive frequencies
magnitudes = np.abs(fft.fft(mid_point_disp))
```

Results

Structured:

Part 1: Modelling the propagation of a wave on a string with fixed ends.

- The results showcase a wave on a string undergoing reflection at fixed boundaries. Initially, the wave is concentrated near the left end. As the simulation progresses, it moves to the left, reflects, and inverts at the fixed end, and then travels back.
- The wave's behaviour, including reflection and inversion at the boundaries, follows the expected physical principles for a wave on a string with fixed ends. Notably, the wave's amplitude remains constant, indicating an ideal, non-dissipative system.
- This inversion and reversion cycle continue, with the wave oscillating between the two fixed ends of the string.
- The amplitude of the wave does not diminish, implying an idealized scenario with no energy loss. In a real-world scenario, material properties and air resistance would lead to damping effects, causing the amplitude to decrease over time.

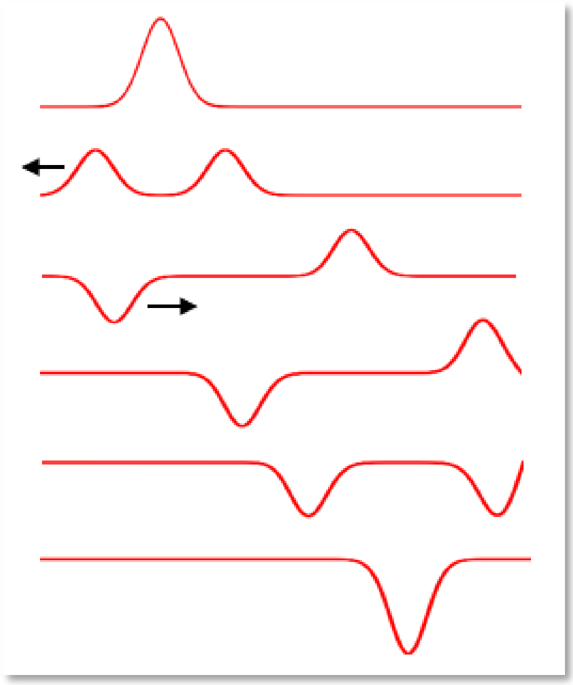


Fig 1: Dynamic Simulation of a Gaussian Wave Packet on a String*:* This figure presents the propagation of a Gaussian-shaped wave packet on a string of length 3 meters, as observed in a discrete simulation with 200 segments. The wave packet, initially centred at one-quarter the string's length, evolves over time with a simulation timestep of 0.001 seconds and a wave speed of 10 m/s. Courant number: 0.6666666666666667, the simulation is likely stable (CFL condition is satisfied).

Part 2/3: Temporal Evolution of a Gaussian Pulse on a String Under Tension and wave speed.

- The simulation captures the string's displacement at five different moments after the initial disturbance: at 0 ms (the moment of release), 2 ms, 4 ms, 8 ms, and 12 ms.
- The initial Gaussian profile, at the centre of the string, is evident in ***Fig 2*** (blue curve). As time advances, the pulse propagates towards the ends of the string while retaining its characteristic shape due to the non-dispersive medium in the simulation.
- At 4 ms (green curve), the wave begins interacting with the boundary, and by 8 ms (red curve), it exhibits reflection and inversion upon reaching the fixed end. This behaviour is in accordance with the theoretical prediction that a wave will invert upon reflecting from a fixed boundary.
- By 12 ms (purple curve), the wave has reflected and travelled back, moving towards the original position but with a reversed amplitude. The lack of damping in the system allows the wave to maintain its energy throughout the observed period, demonstrating the energy conservation in an idealized wave system.
- The simulation effectively illustrates the fundamental principles of wave propagation, reflection, and inversion in a medium constrained by fixed boundaries, providing a visual representation of these concepts.

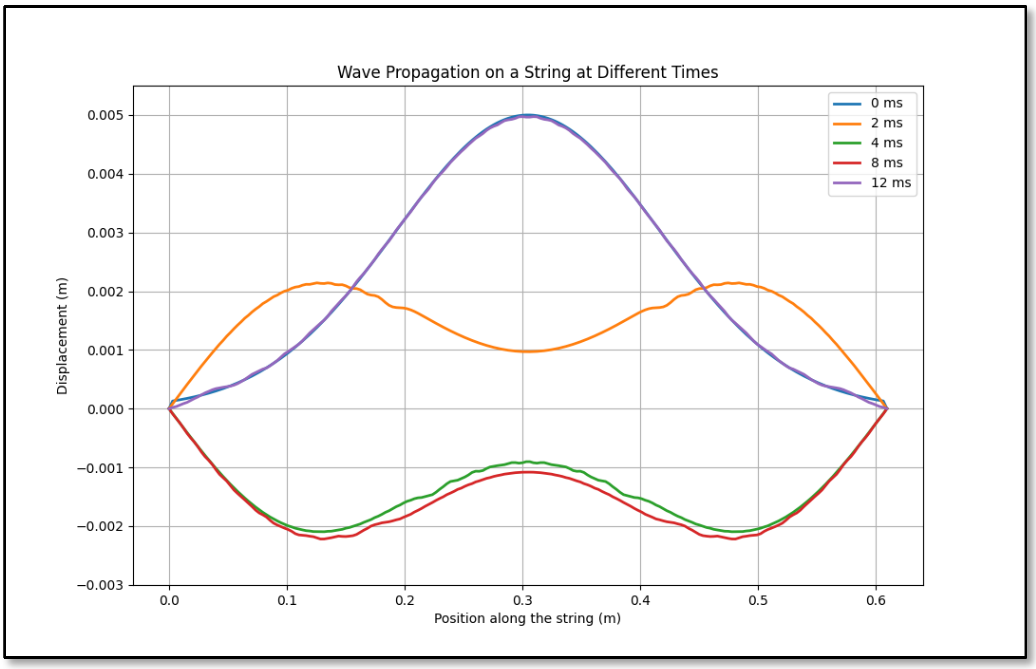


Fig 2: Temporal Evolution of a Gaussian Pulse on a String. The plot visualizes the displacement of a Gaussian wave pulse at successive time intervals (0 ms, 2 ms, 4 ms, 8 ms, and 12 ms) following its release. The string has a length of 0.51098 m (using student number), with the simulation parameters set to a wave speed (c) of 100 m/s and a tension of 100 N. The plot captures the wave's propagation and reflection at the fixed endpoints, displaying the characteristic inversion of the pulse upon reflection *Courant number:* 0.3279763857002 The simulation is likely stable (CFL condition is satisfied).

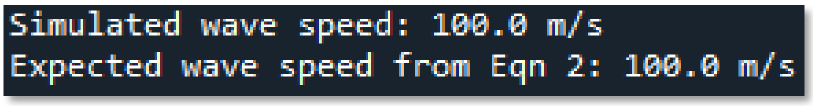
- The wave speed in the simulation, calculated using ***Eqn 3*** , is set at 100 m/s, derived from the string's tension of 100 N and its mass per unit length (μ) of 0.01 kg/m. This speed is a fundamental parameter governing the dynamics of the wave pulse as it propagates along the string. According to the wave equation for a string under tension, the theoretical wave speed *c* is expected to be exactly *Eqn 3* assuming a perfectly elastic and flexible string with no energy loss.

$$c = \sqrt{\frac{\text{tension}}{\mu}}$$

Eqn 3: The speed, c, depends on the tension in the string, T, and the mass per unit length, μ

- In the context of this simulation, the observed wave speed matches the expected theoretical value, indicating that the numerical model accurately represents the physical system. This in accordance validates the simulation's parameters and the computational model used to simulate wave propagation on the string. Any discrepancies between observed and expected wave speeds could suggest the presence of simulation artifacts or the need to account for factors such as damping, string stiffness, or discretization errors.

- The accuracy of the wave speed in the simulation is crucial, as it ensures that the wave behaviour particularly the timing of reflections and pulse shape preservation is correctly modelled. This accurate representation allows us to confidently use the simulation to predict and analyse wave behaviour on similar real-world strings."



```
Simulated wave speed: 100.0 m/s
Expected wave speed from Eqn 2: 100.0 m/s
```

Fig 3: Simulated and Expected wave speed from (Eqn 3) calculated in program.

Unstructured:

Part 1: Evolution of a Realistically Plucked String: From Initial Displacement to Progressive Wave

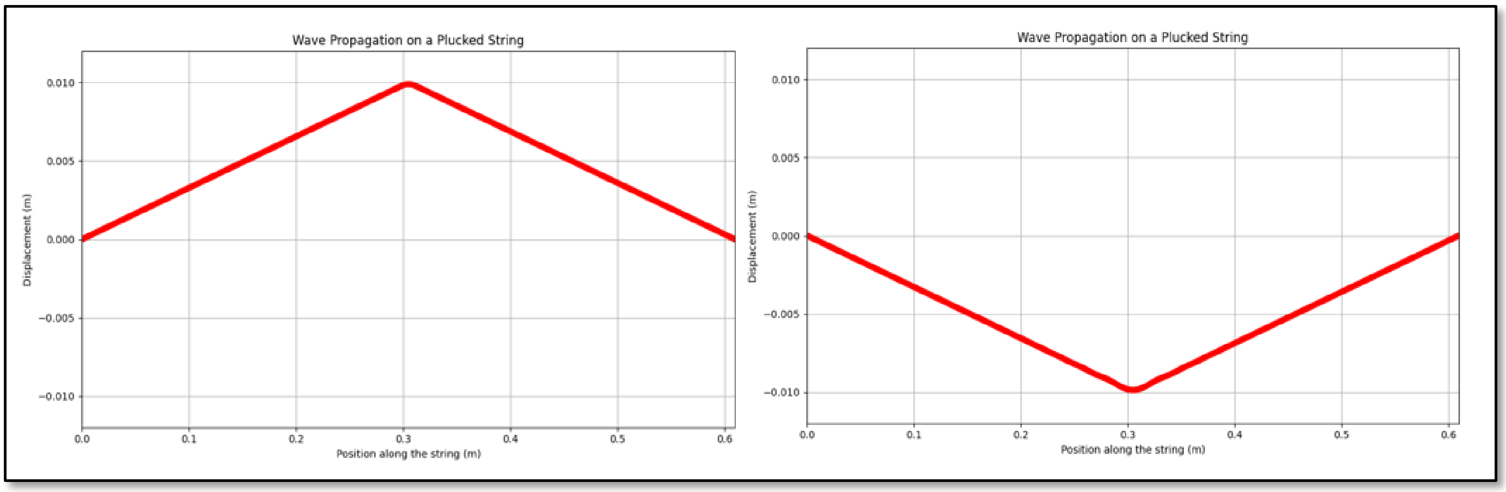


Fig 3/4: Early Stages of Wave Propagation in a Plucked String Simulation. The left panel shows the string immediately after being plucked at the centre, creating a symmetric V-shaped displacement. The right panel captures the string later, illustrating the inversion of the wave as it reflects off the fixed ends. Courant number: 0.3279763857002296, the simulation is likely stable (CFL condition is satisfied).

- The simulation results illustrate the wave propagation on a string that has been asymmetrically plucked, first in the centre and then near one end.
- In the initial scenario (***Fig 3/4***), the string was plucked centrally to a height of 1 cm, forming a linear displacement from the peak to the fixed ends. This led to a symmetrical wave propagation and expected reflection and inversion at the boundaries, aligning with fixed-end string behaviour.
- In the second scenario (***Fig 5/6/7***), plucking near one end produced an asymmetric wave that travelled mainly towards the nearest fixed boundary. Upon reflection and inversion, the waveform's shape was notably less uniform than that of the central pluck, underscoring the pluck position's effect on wave behaviour.

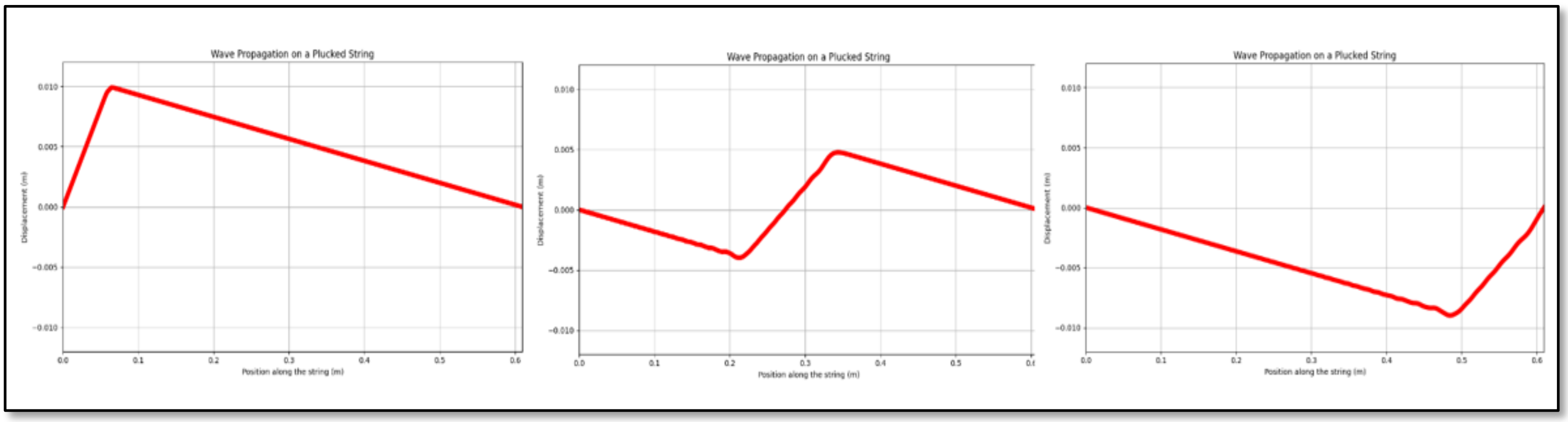


Fig 5/6/7: Sequential Wave Dynamics on a Plucked String. The three panels display the progression of a wave on a string plucked near one end, at three successive time intervals. The left panel shows the immediate linear displacement after the pluck. The middle panel captures the complex waveform as it travels and reflects, demonstrating the initial signs of wave distortion. The right panel shows the wave after reflection, with a clearly visible inversion. Courant number: 0.3279763857002296, the simulation is likely stable (CFL condition is satisfied).

- Simulations initially depicts a tightly configured wave, corresponding to the immediate aftermath of the plucking event. This tight waveform reflects the high degree of order and coherence expected immediately following the displacement. As the simulation continues, the wave begins to exhibit a 'wobbly' appearance, with the once-straight lines of the waveform developing undulations. This change could be attributed to several factors. Physically, it suggests the onset of nonlinear effects or the redistribution of energy across different harmonics of the string, leading to more complex motion.

- The undulating, wobbly motion observed in our string simulation closely aligns with the behaviour of a guitar string captured in a high-speed video, confirming the physical accuracy of our computational model.

Part 2: The effect of having string ends which are free to move up and down in the y direction.

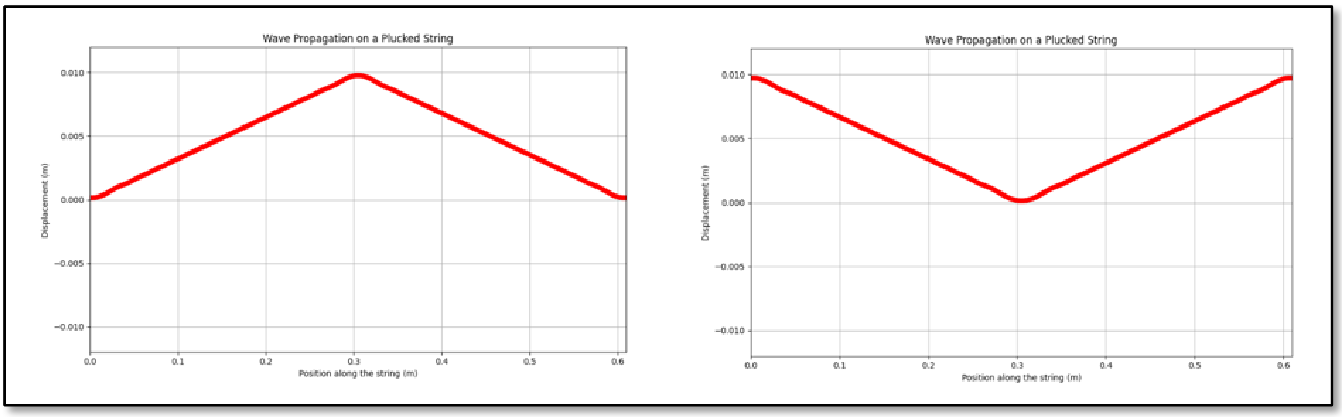


Fig 8/9: Temporal Evolution of a Plucked String with Free Ends: The left image depicts the initial symmetrical displacement immediately after the string is plucked at the centre, while the right image shows the subsequent propagation of the wave later. Both images illustrate the reflection and inversion of the wave upon reaching the free ends Courant number: 0.3279763857002296, the simulation is likely stable (CFL condition is satisfied).

- The results demonstrate wave reflection at free ends without a phase change. This is depicted by the wave crests and troughs reflecting and continuing in the same direction, a characteristic behaviour for waves on a string with free ends.

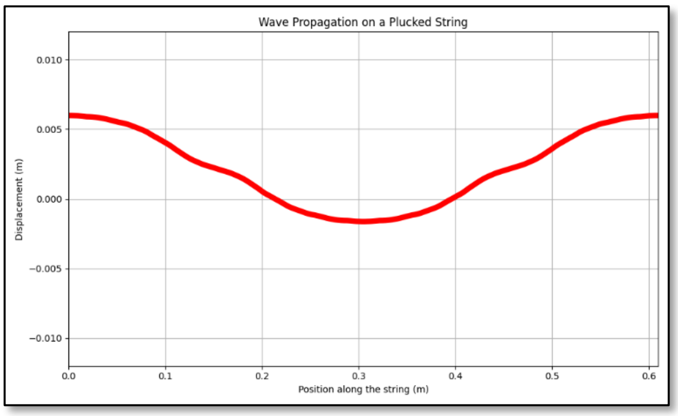


Fig 10: Symmetric Displacement of a Plucked String*: The waveform displays a symmetric amplitude about the zero-displacement axis, characteristic of a standing wave pattern formed due to energy conservation and the boundary conditions of the string.*

- The wave amplitude appears to go below zero and grow symmetrically. This could be due to the conservation of energy within the system and the symmetric nature of the initial displacement. In a real-world scenario, similar waveforms would experience as said before damping due to air resistance and internal material friction.

Part 3: Behaviour of a whip (T and μ can vary along the string)

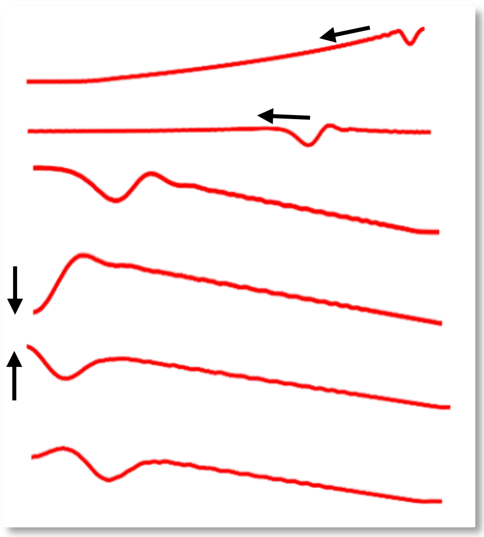


Fig 11: Wave patterns of a string with a varying tension and μ: This series of snapshots illustrates the progressive wave propagation on a string with varying tension and mass distribution.

- In the simulation, it is observed that the wave propagation patterns that reflect a non-uniform tension and mass distribution along the string. A noticeable energy concentration on one side indicates a build-up of amplitude where tension decreases, slowing down wave velocity. This dynamic aligns with the wave equation which relates wave speed to the square root of tension over mass per unit length. As the wave moves towards lower tension areas, it experiences a decrease in frequency and amplitude, like the energy transfer in a whip from the high-tension handle to the low-tension tip, culminating in a sharp amplitude peak.
- This result confirms the theoretical premise that a varying tension and mass distribution along a string can led to complex wave dynamics. These findings have implications for understanding not only physical whips but also other systems where energy transfer results in rapid dynamic events, such as in certain types of musical instruments.

Part 4: Fourier analysis of a Uniform and Realistic string

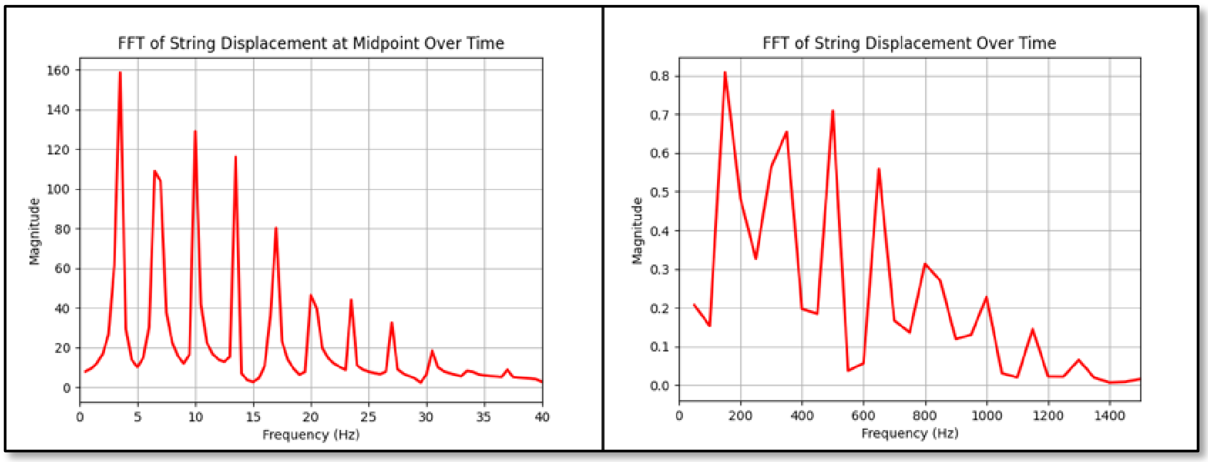


Fig 12/13: Comparative Fourier Spectral Analysis of String Displacements: The left graph illustrates the frequency spectrum resulting from the Fast Fourier Transform (FFT) applied to the midpoint displacement of a string (part 1 structured), revealing a series of pronounced harmonic peaks. The right graph depicts the FFT of a string (part 1 unstructured), characterized by a broader frequency distribution with less defined harmonics, indicating a more complex vibrational pattern. Courant number: 0.6666666666666667, the simulation is likely stable (CFL condition is satisfied).

- The comparative analysis of the Fourier spectral data, as depicted in Fig 12/13, provides insightful contrasts between the two string conditions. The left graph, representing a structured string scenario, exhibits a clear and discrete series of peaks. These peaks are indicative of a well-defined harmonic structure, where each peak corresponds to a natural frequency of the string. The tallest peak represents the fundamental frequency, and the subsequent peaks, at integer multiples of this base frequency, signify the higher harmonics. This pattern is characteristic of a string with uniform or systematically varied properties and boundary conditions that promote regular and periodic vibrations.

- In contrast, the right graph portrays the Fourier transform of a string with unstructured or random initial conditions. The resulting frequency spectrum is more complex, with less pronounced peaks and a frequency distribution that lacks the clear harmonic structure seen in the left graph. The broader spread and reduced peak sharpness suggest a string with either a more complex mode of excitation, irregularities in its physical properties, or damping effects that disrupt the formation of clean harmonic resonances.
- These findings are crucial for applications where precise control over vibrational properties is required.

Conclusion

In conclusion, the investigation into wave propagation on a string has provided valuable insights into different wave properties. The simulations have demonstrated that structured initial conditions yield clear harmonic patterns, while unstructured conditions result in a more intricate frequency spectrum with less defined harmonics. These different wave simulations highlight the significant role initial conditions and physical properties play in wave behaviour. The simulations have successfully illustrated phenomena to those observed in physical systems, such as the energy transfer in whips and the sound production in musical instruments. Furthermore, the simulated scenarios closely align with theoretical predictions, reinforcing the reliability of computational methods in modelling wave mechanics. The implications of these findings extend to practical applications where precise wave control is crucial, offering a deeper understanding of the wave phenomena in various physical and engineering contexts.

Appendix - computer codes used.

All computer codes were written in the Python programming language.
Animations where used: To view animations in Spyder, Go ***to preferences/ IPython console/ Graphics/ Graphics Backend-Change Backend from inline to Qt5***

Structured

Part 1: Uniform string

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parameters
L = 3  # Length of the string
num_elements = 200  # Number of elements or segments
time_steps = 2000  # Number of time steps to simulate
dt = 0.001  # Time step size
c = 10  # Wave speed on the string

def initial_displacement(x):
    return np.exp(-40*(x - L/4)**2) #change to L/2 for pluck in middle

# Define the spatial discretization
dx = L / num_elements

# Define arrays to hold the string displacements
```

```
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# Set the initial displacement
y += initial_displacement(np.linspace(0, L, num_elements))
y_old += y

# Setup the figure and axis for the animation
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(0, L)
ax.set_ylim(-1, 1)
line, = ax.plot(np.linspace(0, L, num_elements), y, 'r', lw=2)

def animate(t):
    global y, y_new, y_old
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_old = y.copy()
    y = y_new.copy()

    line.set_ydata(y)
    return line,

ani = FuncAnimation(fig, animate, frames=time_steps, interval=dt*1000, repeat=False)
plt.xlabel('Position along the string')
plt.ylabel('Displacement')
plt.title('Wave Propagation on a String')
plt.grid(True)
plt.show()

# Calculate the Courant number
courant_number = c * dt / dx
# Print the Courant number
print(f"Courant number: {courant_number}")
# Check if the Courant number satisfies the CFL condition for stability
if courant_number <= 1:
    print("The simulation is likely stable (CFL condition is satisfied).")
else:
    print("Warning: The simulation may be unstable (CFL condition is not satisfied).")
```

Part 2 / 3: **Speed and time steps code**

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
L = 0.5 + 1098/10000  # Length of the string from the given formula
num_elements = 200  # Number of elements or segments
time_steps = 2000  # Number of time steps to simulate
dt = 0.00001  # Time step size
tension = 100  # Tension in the string (N)
mu = 0.01  # Mass per unit length (kg/m)
c = np.sqrt(tension/mu)  # Wave speed on the string

def initial_displacement(x):
    # Gaussian displacement with maximum of 5mm at the center
    return 0.005 * np.exp(-40 * (x - L/2)**2)
```

```
# Spatial discretization
dx = L / num_elements

# Arrays for displacements
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# Set the initial displacement
y += initial_displacement(np.linspace(0, L, num_elements))
y_old += y

# Store the displacement at different times
times_to_plot = [0, 2, 4, 8, 12]  # in ms
displacements = {t: None for t in times_to_plot}

for t in range(time_steps):
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_old = y.copy()
    y = y_new.copy()

    # Store the y values at specified times
    current_time_ms = t * dt * 1000  # convert to ms
    if current_time_ms in times_to_plot:
        displacements[current_time_ms] = y.copy()

# Setup the figure and axis for the combined plot
plt.figure(figsize=(10, 6))

# Plot all times on the same graph
for t, disp in displacements.items():
    plt.plot(np.linspace(0, L, num_elements), disp, lw=2, label=f'{t} ms')

plt.xlabel('Position along the string (m)')
plt.ylabel('Displacement (m)')
plt.title('Wave Propagation on a String at Different Times')
plt.ylim(-0.003, 0.0055)  # Adjust the y-limits to fit the 5mm displacement
plt.legend(loc='upper right')
plt.grid(True)
plt.show()

# Print and compare the simulated wave speed with the expected one from Eqn 78
print(f"Simulated wave speed: {c} m/s")
c_expected = np.sqrt(tension/mu)
print(f"Expected wave speed from Eqn 2: {c_expected} m/s")

# Calculate the Courant number
courant_number = c * dt / dx
# Print the Courant number
print(f"Courant number: {courant_number}")
# Check if the Courant number satisfies the CFL condition for stability
if courant_number <= 1:
    print("The simulation is likely stable (CFL condition is satisfied).")
else:
    print("Warning: The simulation may be unstable (CFL condition is not satisfied).")
```

Unstructured

Part 1: Realistic Pluck

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# parameters
L = 0.5 + 1098/10000
num_elements = 200
time_steps = 1
dt = 0.00001
tension = 100
mu = 0.01
c = np.sqrt(tension/mu)

# Set up the spatial domain for the simulation
x = np.linspace(0, L, num_elements)
dx = L / num_elements

# Initialize arrays to store the string's displacement
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# initial conditions for the plucked string
d0 = 0.01
d0_loc = 0.1  # Location of the pluck as a fraction of the string length change to 0.5 for middle pluck

# initial displacement profile based on the pluck location
y[np.where(x <= d0_loc*x[-1])] = d0/(d0_loc*x[-1]) * x[np.where(x <= d0_loc*x[-1])]
y[np.where(x > d0_loc*x[-1])] = -d0/((1.0-d0_loc)*x[-1]) * (x[np.where(x > d0_loc*x[-1])] - x[-1])
y_old = y.copy()

# Set up the plotting environment
fig, ax = plt.subplots(figsize=(10, 6))
line, = ax.plot(x, y, 'r')

# Define the update function for the animation
def update(frame):
    global y, y_old, y_new, c, dt, dx
    # Update the string displacement using the finite difference method
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_old = y.copy()
    y = y_new.copy()
    line.set_ydata(y)
    return line,

# Create an animation
ani = FuncAnimation(fig, update, frames=time_steps, blit=True, interval=1)

# Set up the axes and labels
ax.set_xlim(0, L)
ax.set_ylim(-0.012, 0.012)
ax.set_xlabel('Position along the string (m)')
ax.set_ylabel('Displacement (m)')
ax.set_title('Wave Propagation on a Plucked String')
ax.grid(True)
```

```
plt.tight_layout()
plt.show()

# Calculate the Courant number
courant_number = c * dt / dx
# Print the Courant number
print(f"Courant number: {courant_number}")
# Check if the Courant number satisfies the CFL condition for stability
if courant_number <= 1:
    print("The simulation is likely stable (CFL condition is satisfied).")
else:
    print("Warning: The simulation may be unstable (CFL condition is not satisfied).")
```

Part 2: No boundaries

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parameters
L = 0.5 + 1098/10000
num_elements = 200
time_steps = 2000
dt = 0.00001
tension = 100
mu = 0.01
c = np.sqrt(tension/mu)

# Spatial discretization
x = np.linspace(0, L, num_elements)
dx = L / num_elements

# Arrays for displacements
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# For the middle pluck scenario
d0 = 0.01  # 1 cm displacement
d0_loc = 0.5  # For middle pluck

# Set the initial displacement
y[np.where(x <= d0_loc*x[-1])] = d0/(d0_loc*x[-1]) * x[np.where(x <= d0_loc*x[-1])]
y[np.where(x > d0_loc*x[-1])] = -d0/((1.0-d0_loc)*x[-1]) * (x[np.where(x > d0_loc*x[-1])] - x[-1])

y_old = y.copy()

fig, ax = plt.subplots(figsize=(10, 6))
line, = ax.plot(x ,y, color='red', lw=6)


def update(frame):
    global y, y_old, y_new, c, dt, dx
    for i in range(1, num_elements - 1):  # Adjust the loop bounds
        y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_new[0] = y_new[1]  # Free boundary conditions
    y_new[-1] = y_new[-2]
    y_old = y.copy()
    y = y_new.copy()
    line.set_ydata(y)
    return line,

ani = FuncAnimation(fig, update, frames=time_steps, blit=True, interval=1)

ax.set_xlim(0, L)
ax.set_ylim(-0.012, 0.012)  # Adjust based on maximum displacement
ax.set_xlabel('Position along the string (m)')
ax.set_ylabel('Displacement (m)')
ax.set_title('Wave Propagation on a Plucked String')
ax.grid(True)

plt.tight_layout()
plt.show()

# Calculate the Courant number
courant_number = c * dt / dx
# Print the Courant number
print(f"Courant number: {courant_number}")
# Check if the Courant number satisfies the CFL condition for stability
if courant_number <= 1:
    print("The simulation is likely stable (CFL condition is satisfied).")
else:
    print("Warning: The simulation may be unstable (CFL condition is not satisfied).")
```

Part 3: Whip

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parameters
L = 0.5 + 1098/10000
num_elements = 200
time_steps = 10
dt = 0.00001

# Variable tension and mass per unit length along the string
tension = np.linspace(100, 10, num_elements)  # Tension decreasing from base to tip
mu = np.linspace(0.01, 0.02, num_elements)    # Mass per unit length increasing from base to tip

# Spatial discretization
x = np.linspace(0, L, num_elements)
dx = L / (num_elements-1)

# Arrays for displacements
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# For the middle pluck scenario
d0 = 0.03  # displacement
d0_loc = 0.1  # For middle pluck

# Set the initial displacement
y[np.where(x <= d0_loc*L)] = d0/(d0_loc*L) * x[np.where(x <= d0_loc*L)]
y[np.where(x > d0_loc*L)] = -d0/((1.0-d0_loc)*L) * (x[np.where(x > d0_loc*L)] - L)

y_old = y.copy()
```

```
fig, ax = plt.subplots(figsize=(10, 6))
line, = ax.plot(x, y, 'r', lw=4)

def update(frame):
    global y, y_old, y_new, mu, tension, dt, dx
    c2 = tension / mu  # Speed squared, now an array
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c2[i] * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))

    # Free boundaries condition
    y_new[0] = y_new[1]
    y_new[-1] = y_new[-2]

    # Swap arrays for the next step
    y_old, y, y_new = y, y_new, y_old
    line.set_ydata(y)
    return line,

ani = FuncAnimation(fig, update, frames=time_steps, blit=True, interval=0.1)

ax.set_xlim(0, L)
ax.set_ylim(-0.06, 0.06)  # Adjust based on maximum expected displacement
ax.set_xlabel('Position along the string (m)')
ax.set_ylabel('Displacement (m)')
ax.set_title('Wave Propagation on a Varying Tension and μ String')
ax.grid(True)

plt.tight_layout()
plt.show()
```

Part 4: Both codes: FFT

#Structured:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq

# Parameters
L = 3  # Length of the string
num_elements = 200  # Number of elements or segments
time_steps = 2000  # Number of time steps to simulate
dt = 0.001  # Time step size
c = 10  # Wave speed on the string

def initial_displacement(x):
    return np.exp(-40*(x - L/2)**2)

# Define the spatial discretization
dx = L / num_elements

# Define arrays to hold the string displacements
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# Array to store the midpoint displacement over time
mid_point_disp = np.zeros(time_steps)
```

```
# Set the initial displacement
y += initial_displacement(np.linspace(0, L, num_elements))
y_old += y

# Simulation loop
for t in range(time_steps):
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c**2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_new[0] = y_new[1]  # Neumann boundary condition
    y_new[-1] = y_new[-2]  # Neumann boundary condition
    y_old, y, y_new = y, y_new, y_old

    # Store the displacement of the midpoint
    mid_point_disp[t] = y[num_elements // 2]

# FFT analysis
mid_point_disp -= np.mean(mid_point_disp)  # Remove DC component
frequencies = fftfreq(time_steps, dt)
magnitudes = np.abs(fft(mid_point_disp))
positive_freq_idxs = np.where(frequencies > 0)


# Plotting the FFT results with a red line and increased thickness
plt.figure()
plt.plot(frequencies[positive_freq_idxs], magnitudes[positive_freq_idxs], 'r', lw=2)  # 'r' is for red, lw for line width
plt.title('FFT of String Displacement at Midpoint Over Time')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.xlim(0, 40)  # Adjust the frequency range as needed
plt.grid(True)
plt.show()

# Calculate the Courant number
courant_number = c * dt / dx
# Print the Courant number
print(f"Courant number: {courant_number}")
# Check if the Courant number satisfies the CFL condition for stability
if courant_number <= 1:
    print("The simulation is likely stable (CFL condition is satisfied).")
else:
    print("Warning: The simulation may be unstable (CFL condition is not satisfied).")
```

#Unstructured:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import fft

# Parameters & Discretization
L = 0.5 + 1098/10000
num_elements = 200
time_steps = 2000
dt = 0.00001
tension = 100  # Tension remains constant
mu = 0.01      # Mass per unit length remains constant
x = np.linspace(0, L, num_elements)
dx = L / (num_elements-1)
```

```
# Displacement arrays
y = np.zeros(num_elements)
y_new = np.zeros(num_elements)
y_old = np.zeros(num_elements)

# Middle pluck setup
d0 = 0.01
d0_loc = 0.1
y[np.where(x <= d0_loc*L)] = d0/(d0_loc*L) * x[np.where(x <= d0_loc*L)]
y[np.where(x > d0_loc*L)] = -d0/((1.0-d0_loc)*L) * (x[np.where(x > d0_loc*L)] - L)
y_old = y.copy()

# Storing the displacement of the midpoint over time
mid_point_disp = np.zeros(time_steps)

c2 = tension / mu  # Speed squared, remains constant since tension and mu are constants

for t in range(time_steps):
    for i in range(1, num_elements - 1):
        y_new[i] = (2*y[i] - y_old[i] + c2 * dt**2 / dx**2 * (y[i+1] - 2*y[i] + y[i-1]))
    y_new[0] = y_new[1]
    y_new[-1] = y_new[-2]
    y_old, y, y_new = y, y_new, y_old

    # Store the displacement of the midpoint
    mid_point_disp[t] = y[num_elements // 2]

# Subtract the mean from the signal to remove the DC component
mid_point_disp -= np.mean(mid_point_disp)

# FFT analysis using scipy
frequencies = fft.fftfreq(time_steps, dt)
positive_freq_idxs = np.where(frequencies > 0)  # Only consider positive frequencies
magnitudes = np.abs(fft.fft(mid_point_disp))
plt.plot(frequencies[positive_freq_idxs], magnitudes[positive_freq_idxs],'r', lw=2)
plt.title('FFT of String Displacement Over Time')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.xlim(0, 1500)
plt.grid(True)
plt.show()
```

References:

- Solving the Wave Equation in 1D - Seg Wiki
  https://wiki.seg.org/wiki/Solving_the_wave_equation_in_1D#:~:text=The%20displace
- Waves on a string -Hyper Physics (R Nave)
  http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/string.html
- Fourier and Wave Simulation - Daniel A. Russell, The Pennsylvania State University
  https://www.acs.psu.edu/drussell/Demos/Pluck-Fourier/Pluck-Fourier.html
- Slow Motion Guitar String Pluck – YouTube
  https://www.youtube.com/watch?v=LNNQvG0jWtw
- Courant Condition - CFL (Guilherme Caminha)
  https://www.simscale.com/blog/cfl-condition/
- Dirichlet boundary condition - Wiki
  https://en.wikipedia.org/wiki/Dirichlet_boundary_condition
- Giordano, N. J. (Computational Physics. Purdue University Press.)