An Implementation of Wave Propagation Simulation Through Finite-Difference Schemes

Luwei Chen

May 12, 2024

1 Introduction

Mathematical model for wave propagation are often entangled in implicit nonlinear partial differential equations with complex spatial and temporal derivatives. Often, the complexity of these models increases with that of the analyzed surfaces. However, the propagation in a stiff vibrating steel string presents a uniquely ideal case. This string can be effectively described using up to second-order time variables and well-defined boundary conditions Bensa (2003). This essay explores the implementation of finite-difference schemes to this problem, benefiting from the linear nature of the proposed model. Theoretical background and rationale for this particular methodology will be first introduced. Then, an implementation is presented. Finally, a discussion on outcomes and limitations are provided.

2 Background

The literature proposed the following family of partial differential equations:

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2} - \kappa^2 \frac{\partial^4 y}{\partial x^4} - 2b_1 \frac{\partial y}{\partial t} + 2b_2 \frac{\partial^3 y}{\partial x^2 \partial t}.$$

One should notice that the linearity of this partial differential equation gives a clear, anatomical view of what wave propagation consists of. Considering the starting terms on both sides of the equation, a typical hyperbolic partial differential equation is described with a constant c:

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

The second term at the right hand side describes the frequency-dependent wave velocity parameterized by a constant κ that relates to the stiff string. The remaining terms represent loss in vibration.

3 Methodology

As suggested (Bensa, 2003), finite difference schemes are considered as an outstanding way in approximating a numerical solution for the equation for several reasons.
First, the linearity of the damping and dispersion terms in the model, generated by an explicit frequency domain analysis of a second order (in time) system, ensures that the

discrete approximation maintains the behavior of this continuous system. Second, the transient dynamic of wave equation can be modeled by finite-difference scheme that step the solution through time iteratively. Lastly, since the wave propagation in our modeled are described in time and space, it makes an ease in using several points on a discretized plane to approximate the derivatives.

4 Implementation

In this case, the program mainly encapsulates two functionalities to implement the simulation process: `wave_prop` is used for generating simulation of wave displacement while `wave_gen` helps creating a WAV sound file for reviewing the simulated results. Notably, the generation of wave propagation with finite-difference schemes broadly contains the following steps.

1. Model Specification
2. Stability Check & Discretization
3. Initial & Boundary Conditions
4. Finite-Difference Iteration

4.1 Model Specification

According to the literature, parameters helping to specify wave propagation model of specific notes out from a family are provided as follows. A series of parameters are adapted from the literature as to make numerical simulations of note C2.

Parameters	C2	Units
L	1.23	m
c	160.9	$m \cdot s^{-1}$
κ	0.58	$m^2 \cdot s^{-1}$
b_1	0.25	s^{-1}
b_2	7.5×10^{-5}	$m^2 \cdot s^{-1}$
F_s	16 000	s^{-1}

Table 1: Adapted from (Bensa, 2003)

```
T = 1/16000 # timestep
fs = 1 / T
L = 1.23 # string length
X = L / 100 # space step
k = 0.58
c = 160.9
b1 = 0.25
b2 = 7.5e-5
lamda = (c * T) / X
mu = (k * T) / X**2
```

4.2 Discretization & Stability

Let t notes a time subinterval and x denotes a sptial subinterval. Then, the temporal domain $[0, T]$ is represented by mesh points:

$$0 = t_0 < t_1 < t_2 ... < t_{M-1} < t_M = T$$

Similarly, the spatial domain $[0, L]$ is represented by points:

$$0 = x_0 < x_1 < x_2 ... < x_{N-1} < x_N = L$$

To capture the wave dynamics with higher accuracy, we chose the time intervals by taking the reciprocals of the given frequency F_s, $t_1 = 1/16000$, for simulated notes C2. Meanwhile, the conditions of stability are given through the following inequality Bensa (2003). It is derived from the fact that the amplification factors' magnitude, typically in finite-difference schemes, does not exceed unity to ensure a stable solution by iterations over-time.

$$t \leq x^2 \frac{-4b_2 + \sqrt{16b_2^2 + 4(c^2x^2 + 4\kappa^2)}}{2(c^2x^2 + 4\kappa^2)}$$

We employed the bisection method to approximate the solutions of this equation, resulting in acceptable minimum of spatial interval $x_1 \approx 0.0118$ and $x_2 \approx 0.0121$ for two notes, respectively. Hence, while satisfying the resulted minimum of space interval x and considering simplicity in algebraic calculation, we set number of subintervals in space as $N = 101$ and that in time, M, to be the product of summation of time (in seconds) and frequency F_s.

```
def stable(X, T=1/32000, b=2.7e-4, c=329.6, k=1.25):
    sqrt_component = sqrt(16 * b**2 + 4 * (c**2 * X**2 + 4 * k**2))
    denominator = 2 * (c**2 * X**2 + 4 * k**2)
    return X**2 * ((-4 * b + sqrt_component) / denominator) - T

def bisection(f, a, b, tol=1e-12):
    if a > b:
        c = b
        b = a
        a = c
    if a == b:
        raise ValueError
    if f(a) * f(b) >= 0:
        raise ValueError
    while fabs((b-a)/fabs(a)) > tol:
        c = (a+b)/2
        sfc = sign(f(c))
        if sfc == sign(f(a)):
            a = c
        elif sfc == sign(f(b)):
            b = c
        elif sfc == 0:
            print("Here is an exact zero")
            return c
        else:
```

```
            raise ValueError("Something went horribly bad: sign(f(midpoint))={}".format(sfc))
    return c
```

The function `stable` helps generating the magnitude of a spatial subinterval that fulfills the stability conditions, given a specified magnitude of time subinterval. Meanwhile, function `bisection` is presented for approximating numerical solutions for the stability inequation.

4.3 Boundary Conditions

In order to generate a uniquely defined numerical solution, initial conditions shall be given. In this model, the spatial domain is restricted to $x \in [0, L]$, where L represents the string length. We supply the initial conditions as following,

$$y|_{x=0} = y|_{x=L} = \frac{\partial^2 y}{\partial x^2}|_{x=0} = \frac{\partial^2 y}{\partial x^2}|_{x=L} = 0$$

The displacement y is set to equal zero at both $x = 0$ and $x = L$ as to simulate a string clamped on both ends. Meanwhile, the zero second derivatives of y (displacement) with regard to x (space) is explained by the stiffness of the steel string, whose displacement, initiated by an exogenous force at both ends, approximates to zero while vibrating.

4.4 Initial Conditions

$$f(x) = \begin{cases} \frac{h}{p}x & \text{for } 0 \leq x < p, \\ -\frac{h}{L-p}(x-p) + h & \text{for } p \leq x \leq L \end{cases}$$

Considering the second order system given, two initial conditions shall be supplied, the initial displacement $f(x)$ and the initial velocity $g(x)$. $f(x)$ prefigures a piecewise function that simulates a plucked string about to be released. The point $(0.7, 0.002)$ specifies f(x) in this case, where $p = 0.7$ represents the initialized point on the string and $h = 0.002$ denotes the vertical altitude of the plucking. It is worth noticing that the discontinuity at the peak of the piecewise function $f(x)$ shall not result in a rather deviated result compared to a counterpart with continuity, the exponential functions. The nature of discretization will compensate such discontinuity. Namely, when dividing the x-axis into pieces of subintervals, the interpolated curve would compensate such discontinuity, which is further reduced through iterations. Meanwhile, we define $y_t(x, 0) = g(x)$ as the initial velocity function, which is by default set to zero.

```
def f(x):
    if 0 < x <= 0.7:
        return (0.002/0.7)*x
    elif 0.7 < x <= 1.23:
        return (-0.002/(1.23-0.7))*(x-0.7)+0.002
    else:
        return 0
```

4.5 Finite Difference Iteration

The function `wave_c2` implements the finite-difference iteration for solving the wave equation. The formulae for calculating the finite differences are supplied by the literature. After prefiguring the finite differences, it assigns the initial conditions to the output matrix U as preparations for further recurrence steps.
To calculate the first step of the finite-difference iteration, we take the generic formula for $n = 0$.

$$y_m^1 = a_{10}y_m^0 + a_{11}(y_{m+1}^0 + y_{m-1}^0) + a_{12}(y_{m+2}^0 + y_{m-2}^0) + a_{20}y_m^{-1} + a_{21}(y_{m+1}^{-1} + y_{m-1}^{-1})$$

To derive a more explicit formula for the first time step by reducing undefined terms y^{-1}, we employ the results from the initial conditions that $g(x) = 0$ for $x \in [0, L]$, giving a symmetry

$$y_m^{-1} \approx y_m^1 - 2Tg(m \cdot X) = y_m^1$$

which is further simplified by assuming $y_m^1 \approx y_m^0$ for the zero velocity. Thus, inserting the simplification in the formula, we have the explicit form for the first step iteration.

$$y_m^1 = (a_{10} + a_{20})y_m^0 + (a_{11} + a_{21})(y_{m+1}^0 + y_{m-1}^0) + a_{12}(y_{m+2}^0 + y_{m-2}^0)$$

```
def wave_c2(sum_of_time):
    # Meshgridding
    N = int(100)
    M = int(sum_of_time / T)
    # Finite Differences
    a10 = (2 - 2*(lamda**2) - 6*(mu**2) - (4*b2*mu)/k) / (1 + b1*T)
    a11 = ((lamda**2) + 4*(mu**2) + (2*b2*mu)/k) / (1 + b1*T)
    a12 = (-(mu**2)) /(1 + b1*T)
    a20 = (-1 + ((4*b2*mu)/k) + b1*T) / (1 + b1*T)
    a21 = (-2*b2*mu/k) / (1 + b1*T)
    # Initializing a matrix
    U = zeros([N+2, M + 1])
    # Setting up boundary conditions
    for i in arange(0, M+1):
        U[0,i] = 0
        U[N,i] = 0
    # Setting up initial conditions
    for j in arange(0, N+1):
        U[j,0] = f1(j*X)
        U[j,1] = (a10+a20)*f1(j*X) + (a11+a21)*(f1((j+1)*X) + f1((j-1)*X)) + a12*(f1((j+2)*X)+f1
    ((j-2)*X))
    # Setting up finite-difference recurrence
    for j in arange(1, M-1):
        for i in arange(1, N):
            U[i,j+1] = (a10*U[i,j] + a11*(U[i+1,j]+U[i-1,j]) +
                        a12*(U[i+2,j]+U[i-2,j]) + a20*U[i,j-1] +
                        a21*(U[i+1,j-1]+U[i-1,j-1]))
    U = U.T
    U = U[:,:101]
    return U
```

4.6 Sound Generation

Function `wave_gen` is responsible for generating a sound file to review the numerical solution of the wave equation. The inputs contains a matrix of wave displacement U, the spatial location loc, represented by the index of the matrix, for sound data sampling, and sampling rate fs. In particular, it normalizes the displacement data into 16-bit PCM format. This ensures a maximum possible dynamic range while prepares a compatibility with audio hardware and software on computer.

```
def wave_gen(U, loc, fs):
    data = U[:loc]
    # Normalization to 16-bit PCM range
    audio_data = np.int16(data / np.max(np.abs(data)) * 32767)

    # Parameters
    nchannels = 1
    sampwidth = 2
    nframes = len(audio_data)

    # Initialize WAV file
    wav_file = wave.open('wavefile.wav', 'w')
    wav_file.setparams((nchannels, sampwidth, fs, nframes, 'NONE', 'not compressed'))

    # Load the data
    wav_file.writeframes(audio_data.tobytes())
    wav_file.close()

    print("WAV file created successfully.")
```

5 Results & Limitations

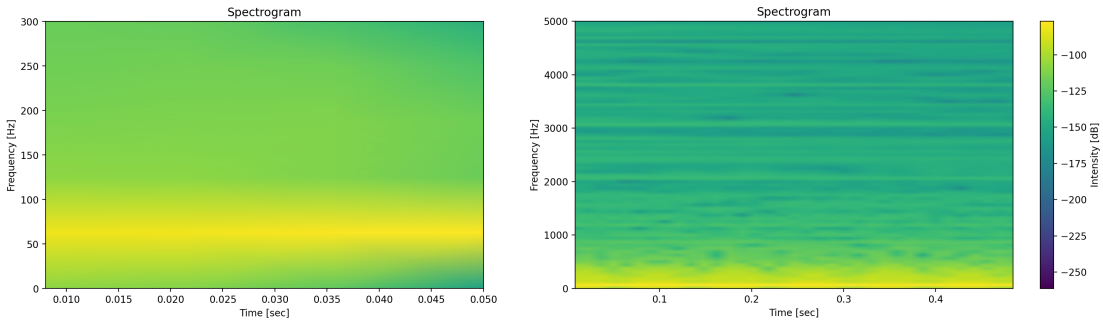

Figure 1: Spectrograms of the finite difference scheme for the note C2 in short term (left) and long term vibration (right).

The particular finite-difference schemes in this essay clearly present a solid, stable representation of the intended note C2. The note C2 has a fundamental frequency of approximately 65.41 Hz. According to the spectrograms, the prominent bright lines with high intensity suggest the presence of the targeted note (Fig 1). Meanwhile, in Figure 1,

the equally spaced lines in a consistent manner above the critical frequency showed clear shapes of the corresponding harmonics in integer multiples, of the note.

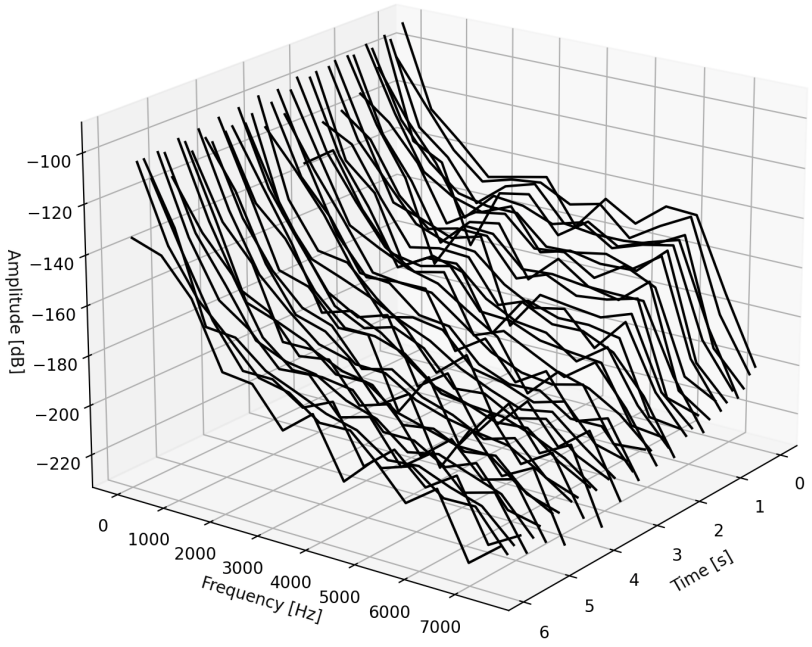

Figure 2: Cumulative Spectral Decay for the simulated note C2.

Apart from the outstanding accuracy in its short term performance, its stability after numerous finite-difference iterations is also guaranteed. Taking a summation of time for simulation up to six seconds, the body of the intended propagation is maintained (Fig 2). In the low frequency domain, where the simulated note is mainly characterized, the loss through time is stable in a predictable manner. At the same time, it is also the case for the decay in medium frequency domain, as the distinct slopes in decay within each frequency illustrated in the graph aligned with the theoretical expectations about the model: a frequency-dependent loss.

However, despite a robust, stable results is presented in the numerical implementation, certain limitations shall be discussed. First of all, compared to its numerical counterparts, the amount of time spent on computational calculations is relatively greater, which can be explained by the complexity in its finite-difference stencil and the generic formula. At the same time, its compatibility with simulating notes in higher fundamental frequency, like C7 or above, is limited, as one may find it difficult to comply with the related stability conditions in such case. While simulating higher notes requires a more refined time steps in order to better capture the waveform, the corresponding resizing in spatial step might be difficult because of the shrinking string length in experimental sampling at the same time.

6 Conclusions

The original model largely contributes to numerical implementations in acoustic simulation by offering an ideal diagram. It is due to the fact that it transforms the complex wave propagation modeling into a well-posed boundary value problem, leading to possible numerical solutions. On top of that, considering its accuracy in capturing waveform, the sustained stability in a proper practical scale, and capacity for a numerical approximation in an entire string vibrations instead of a single point, this theoretical framework successfully offers multi-dimensional understandings of the wave behaviors on an instrumental string while serving as a robust theoretical material for sound synthesizing engineering.

References

Bensa, B. S. K.-M. R. . S. J. O. r., J. (2003). The simulation of piano string vibration: from physical models to finite difference schemes and digital waveguides. *The Journal of the Acoustical Society of America*, *114*(2), 1095–1107. doi: 10.1121/1.1587146