1 A Case Study, the Guitar String

1.1 Introduction

In this section we will *derive* a mathematical model from scratch, starting with a basic equation from physics (Newton's law, $F = ma$, i.e. the force F exerted on a body equals the mass of the body times its acceleration). We choose a relatively simple model of vibrations in a guitar string, and will go through all phases in modeling and simulation. We will go through the modeling and solver layer in some detail in order to see an example of how these phases actually work and how the mapping from one layer to another is accomplished. We will also discuss numerical stability of the resulting discretization and the influence of computational errors.

1.2 The model

Imagine a guitar player who wonders how it is possible that strumming the strings of his guitar results in the beautiful guitar sound. Strumming a string results in vibrations in the string, that are coupled to the body of the guitar. The body also starts vibrating and together with the string induces air pressure fluctuations that we recognize as the sound of a guitar. Building a full model of this would be very complicated, and before embarking on such a journey, one usually starts with taking the full system apart and trying to understand the behavior of the parts. In this case, it is clear that the basic phenomenon to be studied is the vibration of the guitar string. Therefore, we decide the study an isolated guitar string of length L, held at a tension force T, and fixed at its end points. We will only consider transverse (i.e. perpendicular to the string) vibrations (see Figure 1).

More specifically, we seek a model for the amplitude $y(x,t)$ of the string as a function of the position x along the string, with $0 < x < L$, and as a function of the time t. We will first derive an equation for the amplitude $y(x,t)$.

In equilibrium the amplitude of the string is zero everywhere and the tension is equal to T everywhere. Next, we assume that as the string vibrates, it has a small amplitude y, such that the tension remains equal to T everywhere in the string. From Figure 2 we will evaluate the tensions on a small part of the string, on point x and $x + dx$, where dx is very small.

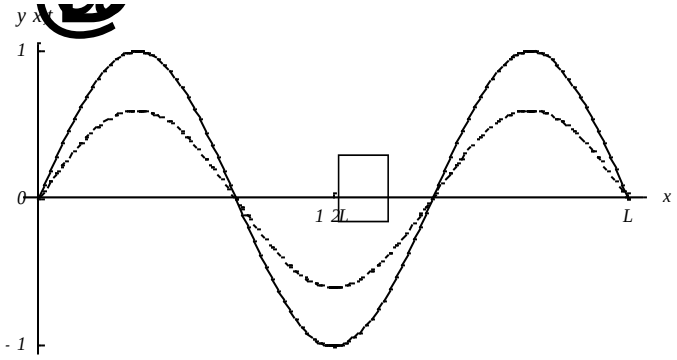

Figure 1: The vibrating string, two solutions are shown on different times t.

Because of the curvature of the string, the tensions on both points are not exactly equal. The difference in the y-components of tension on position x and $x + dx$ yields a resulting force, that can be expressed as

$$F_y = T(\sin(\alpha') - \sin(\alpha)). \qquad [20]$$

Because we assumed that the amplitude y is very small, the angles α and α' are also very small. Therefore, $\sin(\alpha) \approx \tan(\alpha)$. We immediately recognize $\tan(\alpha)$ as the slope the string, i.e. $\tan(\alpha) = \partial y/\partial x$. Using all this, we rewrite Eq. [20] to

$$F_y = T\left(\left.\frac{\partial y}{\partial x}\right|_{x+dx} - \left.\frac{\partial y}{\partial x}\right|_{x} \right), \qquad [21]$$

which, in the limit of very small dx becomes

$$F_y = T\frac{\partial^2 y}{\partial x^2} dx\,. \qquad [22]$$

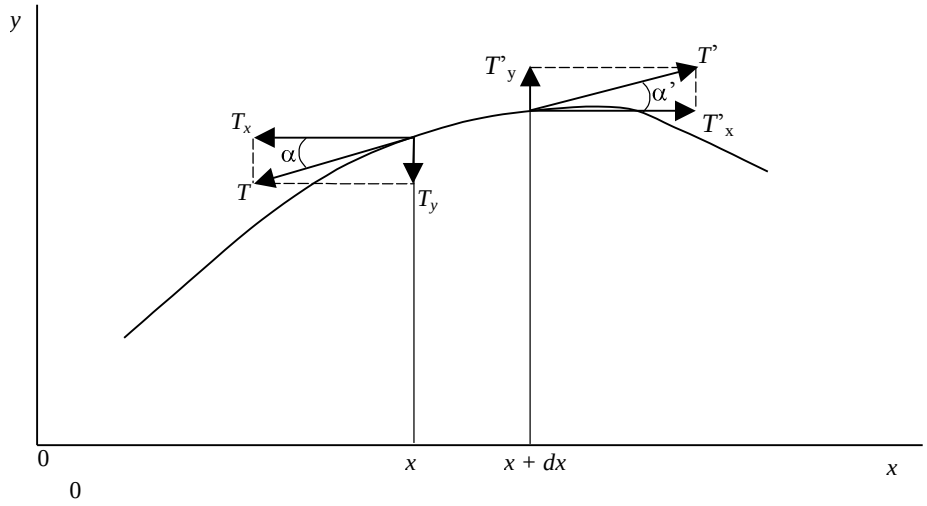

Figure 2: Tensions on a part of the string.

The length of the string between coordinates x and $x + dx$ is ds. In the limit of very small dx, we find that $ds = dx$. If we call μ the mass of the string per length unit, Newton's law results in

$$F_y = \mu dx \frac{\partial^2 y}{\partial t^2}\,. \qquad [23]$$

By combining Eq [22] and [23] we find the one-dimensional wave equation,

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2}\,, \qquad [24]$$

where

$$c = \sqrt{T/\mu} \qquad [25]$$

is the velocity of sound in the string.

Before we proceed, note that in Figure 2 the resulting force in the *x*-direction is zero, which means that the movements of the string are only in the *y*-direction. Prove this yourself.

The one-dimensional wave equation is our model of vibrations in the string. However, before we can solve it, we need to supply boundary – and initial conditions. The boundary conditions have already been mentioned. The string is fixed on both ends, or, mathematically,

$$y(0,t) = y(L,t) = 0\,. \qquad [26]$$

The initial condition gives the amplitude of the string on time $t = 0$, i.e.,

$$y(x,0) = f(x)\,, \qquad [27]$$

where $f(x)$ can be any smooth function that satisfies Eq. [26].

Although this model looks very simple, it is actually used in studies of vibrating guitar and piano strings. If you are interested you could read Ref. [i] as a very nice introduction in this field.

The one-dimensional wave equation [24] with boundary condition, Eq. [26], and initial condition, Eq. [27], can be solved analytically. We will not derive the result, which is easily obtained using Fourier analysis, but just give the final solution,

$$y(x,t) = \sum_{m=1}^{\infty} B_m \sin\left(\frac{m\pi x}{L}\right)\cos\left(\frac{m\pi ct}{L}\right), \qquad [28]$$

where the coefficient B_m is determined by the initial condition as

$$B_m = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{m\pi x}{L}\right)dx\,. \qquad [29]$$

Each term in the series in Eq. [28] must be interpreted as a normal - or eigen mode of the string. The $m = 1$ mode is the ground tone of the string. It has it's maximum in the middle of the string, and vibrates with a frequency

$$\nu = \frac{1}{2L}\sqrt{T/\mu}\,. \qquad [30]$$

Increasing the tension in the string or taking a lighter string results in a higher frequency of the ground tone. If you are a guitar player you know this by experience. The string also supports higher harmonics, whose frequencies are *m* times that of the ground tone. The higher harmonics have so-called knots, places where the amplitude is always zero. For example, the $m = 2$ harmonic has a knot for $x = L/2$. The subtle mix of ground tones and higher harmonics gives each string instrument it's characteristic sound.

1.3 The Solver

We have solved our model, i.e. the one-dimensional wave equation, and could proceed by analyzing in detail what would happen if we take special initial conditions that resemble, e.g., picking a string. However, we proceed in a different way, and seek a numerical method to solve the wave equation. In doing so we enter the field of numerical analysis. We will only touch upon some issues from this field. If you want more information, and like to have it in a

cook book style you are advised to read Numerical Recipes [ii]. For more in-depth discussion Ref. [iii] could be consulted.

We have to cast the wave equation into a form that allows simulation on a computer. We do this by *discretizing* the equation. First, we define a computational grid, see Figure 3. We only seek solutions for $y(x_i,t_j)$, where $x_i = i\Delta x$ and $\Delta x = L/N$, where N is the number of points on the x-axes. Furthermore, $t_j = j\Delta t$, where Δt is a predetermined time step.

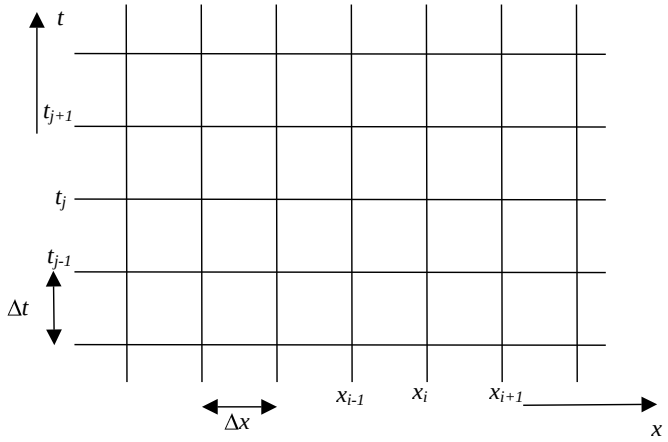

Figure 3: The computational grid.

Now recall that the formal definition of the derivative of a function $g(x)$ is

$$\frac{\partial g(x)}{\partial x} = \lim_{dx\to 0}\left(\frac{g(x+dx)-g(x)}{dx}\right).$$

If the grid spacing Δx is small enough, we may write to a good approximation

$$\frac{\partial g(x)}{\partial x} \approx \frac{g(x+\Delta x)-g(x)}{\Delta x}. \qquad [31]$$

Eq. [31] is called a finite difference representation of the derivative. The approximation in Eq. [31] gets better if Δx becomes smaller. Mathematically we say the error in Eq. 31 is of order Δx^2, or $O(\Delta x^2)$. This is easy to prove by considering the Taylor expansion of $g(x+\Delta x)$ around x. We apply Eq. [31] to discretize the wave equation. First consider the time derivatives in the wave equation:

$$\frac{\partial y(x_i,t_j)}{\partial t} \approx \frac{y(x_i,t_j+\Delta t)-y(x_i,t_j)}{\Delta t}. \qquad [32]$$

Using Eq. [32] we can now derive a finite difference equation for the second time derivative,

$$\frac{\partial^2 y(x_i,t_j)}{\partial t^2} \approx \frac{y(x_i,t_j-\Delta t)-2y(x_i,t_j)+y(x_i,t_j+\Delta t)}{\Delta t^2}. \qquad [33]$$

A comparable result is easily obtained for the second derivative in x,

$$\frac{\partial^2 y(x_i,t_j)}{\partial x^2} \approx \frac{y(x_i-\Delta x,t_j)-2y(x_i,t_j)+y(x_i+\Delta x,t_j)}{\Delta x^2}. \qquad [34]$$

Combining Eqs. [24, 33, 34] results in

$$y(x_i,t_j+\Delta t) = 2y(x_i,t_j) - y(x_i,t_j-\Delta t) + \tau^2 \left(y(x_i-\Delta x,t_j)-2y(x_i,t_j)+y(x_i+\Delta x,t_j) \right) \qquad [35]$$

where

$$\tau = \frac{c\Delta t}{\Delta x}. \qquad [36]$$

Equation [35] is an expression to calculate the amplitude on a next time step $t_j+\Delta t$, using values on earlier times (i.e. on t_j and $t_j-\Delta t$). This is called an explicit finite difference scheme. Many other schemes to discretize the wave equation can be devised, e.g. implicit finite difference equations, but we pick this one because it is relatively easy to parallelize. However, before moving to this subject we do need to say a few more words about the numerical scheme that we just derived. For all details we again refer to books on numerical algorithms, see e.g. Refs [ii, iii].

Explicit versus Implicit

Equation [35] is called explicit because it gives us a formula to calculate explicitly the amplitudes on a next time step, i.e. without the need to solve equations. Alternatively, one could have an implicit finite difference scheme. Here, in order to calculate the amplitudes on the next time step, one must solve a system of equation. This happens when e.g. we would estimate the spatial (in the *x*-direction) derivatives on the next time step, i.e. on $t + \Delta t$, instead of on time *t*. The resulting finite difference equation would now contain the amplitudes on x_i on the next time step, but also on $x_i \pm \Delta x$. Writing down the equations for all values of *i* results in a set of linear equations that need to be solved. The main advantage of implicit schemes is that they are unconditionally stable, and therefore allow much larger time steps then explicit schemes, however at the expense of the need to solve systems of equations.

In general, we judge the quality of a numerical scheme on the basis of the following three criteria:

- consistency,
- accuracy,
- stability.

It would go to far to discuss these items in detail. We will just explain what is meant by these criteria, and shortly state how they apply to our finite difference scheme in Eq. [35].

Consistency is the requirement for any algebraic approximation to a partial differential equation to reproduce the partial differential equation in the limit of an infinitesimal time step and grid spacing. It is easy to prove that our finite difference scheme is consistent with the wave equation.

A natural requirement of the finite difference scheme is that the computed amplitudes resemble, i.e. are very close, to the real solutions of the original wave equation. Accuracy and stability are related to this requirement. Accuracy is concerned with local errors. Local errors arise from two sources. First are the roundoff errors resulting from the finite word length of numbers within a computer, and second are the truncation errors caused by representing continuous variables by discrete sets of values. Generally, roundoff errors are much smaller than truncation errors, and provided the scheme is stable (see below) they can usually be ignored. Truncation errors, which arise from the representation of continuous quantities by discrete sets of values are usually described in terms of the difference between the differential and algebraic equations. A measure of the smallness of truncation errors is given by the order of the difference scheme.

Stability is concerned with the propagation of errors. Even if truncation and roundoff errors are very small, a scheme will be of little value if the effects of the small errors grow rapidly with time. Instability arises from the non-physical solutions of the discretized equations. If the discrete equations have solutions which grow much more rapidly than the correct solution of the differential equations, then even a very small roundoff error is certain eventually to seed that solution and render the numerical results meaningless. A numerical method is stable if a small error at any stage does not lead to a larger cumulative error. Without prove we say that our finite difference scheme is stable if and only if

$$\tau \leq 1 . \qquad [37]$$

This equation is a so-called Courant stability condition. This stability condition in facts limits the time step that we can take to $\Delta t \leq \Delta x/c$. We choose Δx in such a way that we have a high enough resolution to "support" certain wavelengths on the grid. As a rule of thumb, to support a wave with a wavelength λ we should choose $\Delta x < \lambda/10$. This means that in order to support the $m = 1$ node, with $\lambda = 2L$, we should take $N > 5$. In general, to support solutions up to mode m, $N > 5m$. In combination with the sound speed c in the string this determines the maximum time step Δt that we are allowed to take in the numerical scheme. Of course we would like to take an as large as possible time step. However, in the explicit finite difference scheme this is not possible because of the stability constraint. So-called implicit schemes, as discussed in the sidebar, are unconditionally stable, which means that for any Δt stability is guaranteed. This allows to take much larger time steps, however at the expense of the need to solve a set of equations.

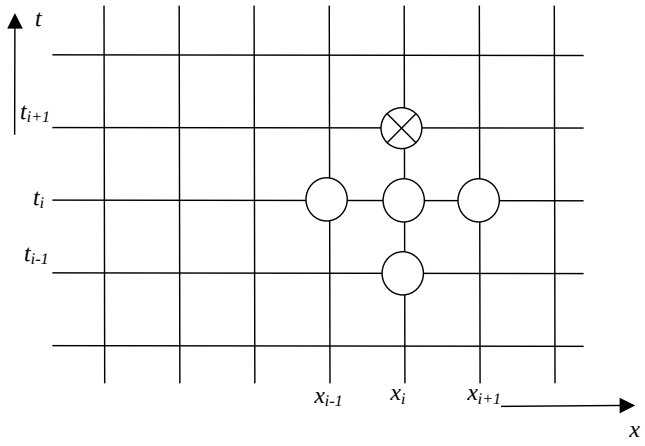

Figure 4: The stencil of the finite difference equation. The circle with the cross inserted is the point that is going to be updated, the empty circles are the points that are needed to perform the upate.

The explicit finite difference formula, Eq. [35] is a so-called stencil operation. In order to calculate the amplitude on a next time step we need a small number of points on previous times to perform the calculation. This set of points is called the stencil, which is drawn in Figure 4. The important property of such stencil operations is that we only need a small and *local* neighborhood to update the amplitude.

---

i Nicholas Giordano, "The physics of vibrating strings", Computers in Physics **12**, 138-145 (1998).

ii W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C, The Art of Scientific Computing* (Cambridge University Press, 1988)

iii J. Stoer and R. Bullirsch, *Introduction to Numerical Analysis* (Springer Verlag, 1980).