


For my project I first decided I wanted to do something investigating properties of a guitar. As I looked into the idea further, I decided it would be more feasible to concentrate my investigation on the strings of a guitar. Seeing as I have two guitars, a classical nylon-stringed guitar and a steel-stringed guitar, I thought it would be interesting to compare the two types of strings. I had had a bit of a tough time understanding how to do the fourier transforms earlier this year, so I also decided that I would “face the beast” and use them in my project so that I would learn how to do them.

Given these bases for my project, I decided to write a program to predict the audio time signal generated by a guitar string when plucked, including a comparison of nylon and steel strings. For the project, I decided to use the D string of each guitar, because, one, it was mid-range, and two, in case of breakage, I had extra D strings. Before beginning work on calculations, I first found a software program that could satisfactorily create a real-time plot of a audio time signal from input coming through the sound card. After I had obtained a copy of a program that fulfilled the requirements, I began work on the simulation.


I had attended the first day of Physics 230, so I had a basic overview of Maple, and an idea of its capabilities and syntax. I went to the computer lab and went to work. I first set out to create a simulation of the whole guitar string over time, given an initial position. The hard part seemed to be integrating both time and location into the equations. Though I had a general idea of what was needed, I had problems getting the nodes to stay put. I realized that I should probably use the equation for a standing wave, but when I tried it the simulation looked too simple, so I went back to my old equations. A meeting with my instructor showed me the error



of my way, and I inserted the standing wave equation back into the program. Because of the fixed nodes, I was able to eliminate the cosine from the transform, which simplified matters. When I met with my instructor, he also showed me an equation for adjusting the omega coefficient of the waves due to the imperfect nature of real strings (Appendix III, line 6). I also inserted this equation into the program.

Once the program seemed to be working using arbitrary values for initial data, I began gathering actual data (Appendix II). The lengths and frequency were easy to obtain. To obtain the velocity and the coefficient for the omega-value equation, I solved the equation using the frequencies of the first and second harmonics. With this data, along with an absolute value function to approximate the initial position of the string, I was ready to begin calculations. To predict the time signal, I plotted the location of a point very close (10^{-3} m ~~to~~) to the second node of the string. I created the plots, using a sum of 50 waves, on a time period approximately equal to that captured by the time signal software, beginning .05 seconds from the initial time in order to compensate for a delay in the trigger of the software. The plots created (Appendix I) looked rather reasonable, so I saved images of them and went home to obtain the real things.

Upon comparison of the plots, I found that the captured time signal seemed to be a reversal or mirror image of the predicted signal. I'm guessing that this difference is due to the method of the software. Other than that aspect, the plots seemed to resemble their counterparts to some extent. The method of obtaining the audio signal is a possible reason for the large difference in the smoothness and magnitude of the nylon and steel plots. My steel-string guitar has a piezo-electric pickup in its bridge, which allows for a fairly accurate simulation using the method employed. My nylon-string guitar, however, does not have its own pickup, so I had to use a microphone. This means that the guitar body has a much larger impact on the audio signal,



dampening out the very high frequencies, and amplifying those that coincide with natural resonances of the body. Discrepancies between the predicted and obtained time signals is most likely due to the approximation of the initial position, and lurking variables such as dampening.

Appendix I: Time signal plots

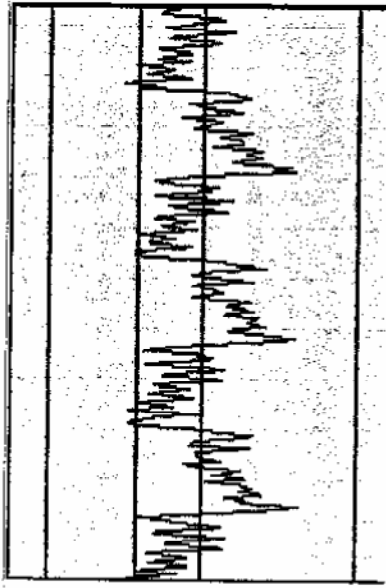


Figure 1 - Steel string time signal

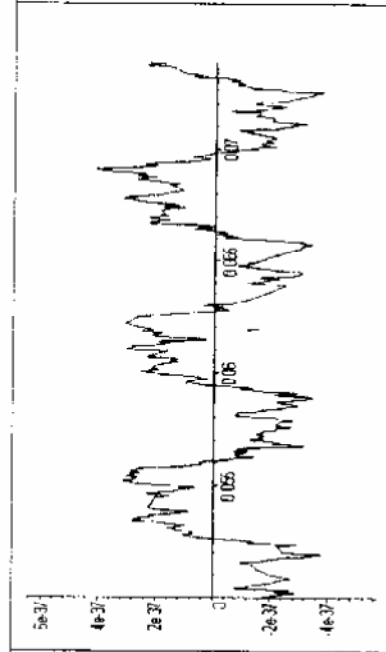


Figure 2 - Steel string simulated time signal with adjusted ω

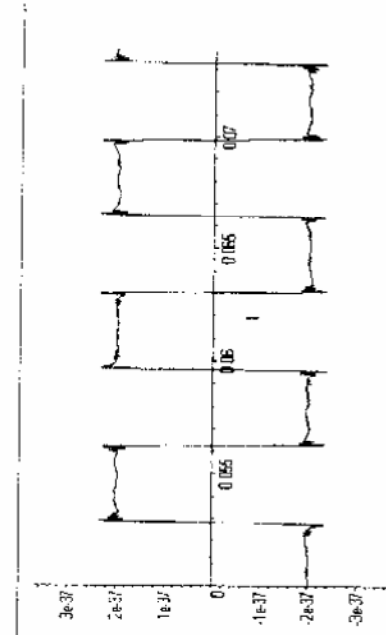


Figure 3 - Steel string simulated time signal with classical ω

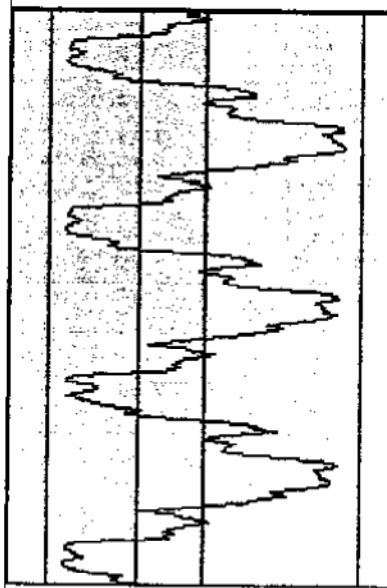


Figure 4 - Nylon string time signal

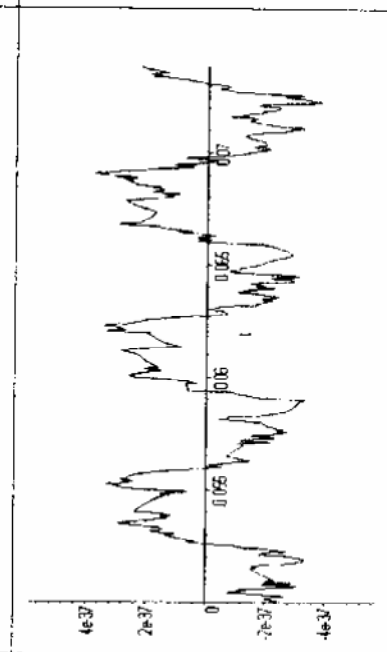


Figure 5 - Nylon string simulated time signal with adjusted ω

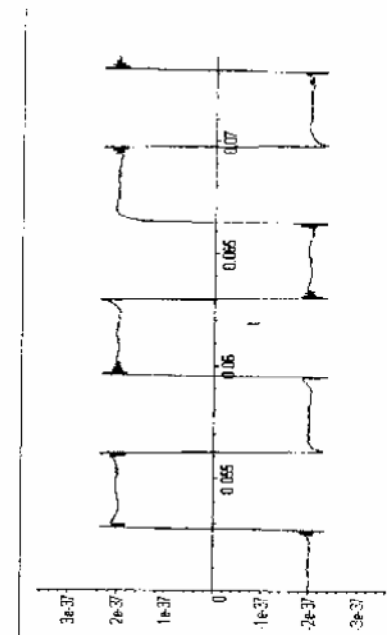


Figure 6 - Steel string simulated time signal with classical ω

Appendix II: Data

Natural Frequency of string (note D)	Nylon	Steel
Length of String	147 Hz	147 Hz
α - alpha coefficient for adjusted ω	.65405 m	.6477 m
Velocity of waves on string	1.8186 m ⁴ /s ²	.7 m ⁴ /s ²
	192.18 m/s	190.59 m/s

Appendix III: Program source code

```

> L := 1.3081;#wavelength = 2*length of string
> k := 2*Pi/(L);
> F(x) :=1-4/L*abs((x-L/4));#description of initial condition of string
> v := 192.1815680;#velocity of wave on string
> w:= k*v;#classical definition of omega
> w2:=sqrt(v^2*m^2*k^2+a*m^4*k^4);#adjusted definition of omega
> a:=.7;#alpha constant for adjusted omega
> t1:=0..10/147;#time period for string simulation
> t2:=(100)..(100)+35/1470;#time period for time signal plot
> x1:=(1-10^(-37))*L/2;#point used for time signal plot
> Am := 4/L*int(F(x)*sin(m*k*x),x=0..L/2);#definition of sine coefficients
> with(plots):animate({F(x),add(Am*sin(m*k*x)*cos(w2*t),m=1..50)},x=0..L/2,t=t1,frames=100);#string
simulation with adjusted omega
> plot(eval(add(Am*sin(m*k*x)*cos(w2*t),m=1..50),x=x1),t=t2);#time signal plot with adjusted omega
> with(plots):animate({F(x),add(Am*sin(m*k*x)*cos(m*w*t),m=1..50)},x=0..L/2,t=t1,frames=100);#string
simulation with classical omega
> plot(eval(add(Am*sin(m*k*x)*cos(m*w*t),m=1..50),x=x1),t=t2);#time signal plot with classical omega

```