Karplus-Strong Plucked String Modeling in MATLAB

Andrew O'Neil-Smith
Frost School of Music
Music Engineering Technology
University of Miami
Coral Gables, FL 33146

Introduction

This project is based on the Karplus-Strong plucked string-modeling technique of digital audio synthesis. MATLAB was used to code both the graphical user interface and the code structure behind it. It demonstrates the different types of sounds that can be created using the Karplus-Strong technique.

Initial Need for a Plug-In

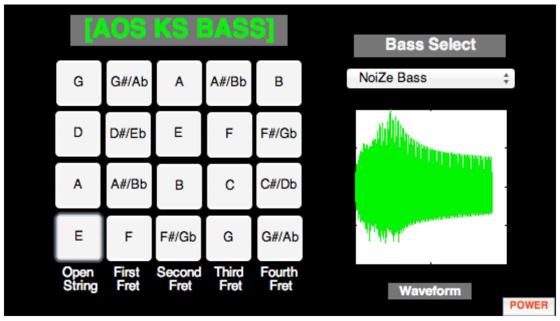
The main goal that I wanted to achieve with this lab was to build a cool user interface that took input from the QWERTY keyboard when the user pressed a key. Any plug-in I have used for a bass or guitar input has the notes mapped to a piano graphical user interface.

The inspiration came from a cool Google Chrome web app called Jam With Chrome (http://www.jamwithchrome.com/select/). You can choose from drums, guitar, keyboard, and bass. Each instrument has an "Easy" and "Pro" mode. The bass and guitar pro mode maps each fret to a row on the QWERTY keyboard. I thought it would be cool to take that idea and apply it to a graphical user interface generated in MATLAB and use some of the different synthesis techniques to create unique bass tones.

Designing my Synthesizer

Almost immediately I ran into problems that would plague idea throughout the rest of the project. The largest obstacle to overcome was learning all about GUIDE, the graphical user interface development environment bundled with MATLAB. I watched many helpful and not-so-helpful videos on YouTube and the

Mathworks website. The MATLAB help files were also very helpful teaching me about how callback functions worked and the different parameters within GUIDE. The elements I used in my GUI were push buttons, static text boxes, a popup menu, and axes for a plot. With just these simple controls, I was able to achieve my initial goal of having a fret board on the QWERTY keyboard with different selectable tones.



The graphical user interface implemented with GUIDE.

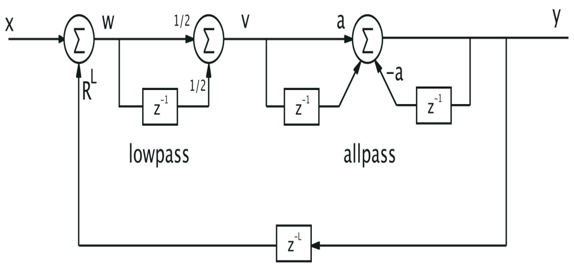
An alternative design I was thinking about using was having four popup menus that would select a note and then one single pushbutton to play the selections from the menus. I decided against this method even though it would theoretically be easier to implement because I wanted to stay true to my original desire to have frets mapped to keys. It would also not sound very pleasing because the frequencies on a bass guitar are too low to sound good played together.

For the design of my graphical user interface, I sketched out a drawing on paper for my initial design. Then I read all about GUIDE in the MATLAB help files. I also watched several videos online explaining the different elements in GUIDE. From that, I was able to decide which features my synthesizer would have and the easiest way to implement them. The color scheme and overall look was inspired by the AMS digital delay outboard gear we have in the studio here.

Coding my Synthesizer

My design for my interface maps the keys [z,x,c,v,b] to the open low E string to the fourth fret on a bass guitar, [a,s,d,f,g] to the open A to the fourth fret, [q,w,e,r,t,y] to the open D to the fourth fret, and finally [1,2,3,4,5] to the open G to the fourth fret.

It took me a while to understand how the callback function linked up and passed parameters with the push buttons on-screen in the GUI. I was able to at least test that the keys were being detected by calling a function I know would give me an error when I pushed it. I called soundsc(y,Fs) and did not pass 'y' to get that line to intentionally fail. This way I was able to verify that MATLAB was at least registering the key push.



Flowchart of a Karplus-Strong filter.

When I first did the Karplus-Strong filter in lab number six, I learned how to write functions in MATLAB. Up until that point, we had been running all of the commands in one script. I wrote a function called KSFilter that would take an input signal and pass it through the Karplus-Strong filter to get a plucked string sound. I chose to do four different excitations; sine wave, square wave, impulse, and noise.

This was the code for the numerator and denominator I solved for to get the Karplus-Strong filter:

```
   N = [a a+1 1 zeros(1,L)];
   D = [2 2*a zeros(1,L-2) -a*R^L
-R^L*(a + 1) -R^L]
```

Improvements on my Design

If I were to come back and work on this project some more, I would have several things I would like to improve. One bug I found was that the low A#/Bb button would only play in sine mode. The bass select menu would jump back to the sine selection no matter what you did to select it. I suspect it had something to do with the switch/case functions that I was using to set the four options.

Speed of My Program

Another thing that is a limitation of this synthesizer is that it runs very slow. This is because it is monophonic; there is no way to play more than one note at a time with it. MATLAB must finish playing the note before it will play the next note.

Another thing that takes slows the performance down is plotting the waveform and displaying it on the synthesizer. There are a few different things I could try to improve this. One would be to have all the plots stored as images and just load them into the desired area on the GUI. I do not have any experience with images in MATLAB so I am unsure if this would be more efficient. A second way to improve would be to only plot the data in a certain range. I believe that I plotted too much data; the more points, the longer it takes to display. I put zoom tools on the toolbar so the user could zoom in on the waveform if they desired. There could also be different ranges for different types of waveforms. The range for the sine wave to look good versus the range for a noise burst to look good is different. This is something I would like to improve in the future.

Another reason it might be slow is that there is close to 1500 lines of code in the project. While a lot of the code was part of the automatically generated GUI code, there are a few things I think I could have done better to reduce the amount of code. What I wanted to do was have a frequency associated with a keyboard button press that would pass it as a variable to the on-screen pressing of the note. What ended up happening was I just copied and pasted the same code from the pushbutton section to the keyboard detection section.

Organization of Code

The code is also somewhat unorganized. Since it was auto-generated by GUI, whenever I added a part to the template, it generated code in the script file. This meant that the generated code had nothing to do with the actual position of the pushbuttons in the GUI. The way the code is numbered and the way the pushbuttons are displayed are not the same. If I had a concrete plan before I designed it in GUIDE, then the code would've been in a better order.

For another iteration, I would like to include some cool effects such as distortion or an auto-wah effect. These types of effects are something that could be attached to a slider somewhere on the GUI. The code behind them is also something that could be done with a little effort.

Conclusion

Overall, I had a lot of fun with this project because I was able to create something cool as a finished project. All of the labs up to that point just put out sound and graphs, but I was able to put those less exciting elements into a creative design that tied them all together. I learned a lot about graphical user interface design. There were several things I think that I could improve on if I came back to this project in the future. I am really looking forward to the next part of this class where we design synthesizers in a dedicated software environment.

References

A Guide to MATLAB. 2006. Cambridge, England: Cambridge University Press.

Jaffe, David A., Smith III. *Extensions of the Karplus-Strong Plucked-String Algorithm.* 1983. Cambridge, MA: MIT Press.

Karjalainen, Valimaki, Tolonen. *Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and Beyond.* 1998. Cambridge, MA: MIT Press.

Smith III, J.O. *Physical Modeling Using Digital Waveguides.* 1992. Cambridge, MA: MIT Press.

Smith III, J.O. *Viewpoints on the History of Digital Synthesis*. 2002. Stanford, CA: Stanford Univeristy.

Steiglitz, K. *A Digital Signal Processing Primer.* 1996. Menlo Park, CA: Addison-Wesley Publishing Co.

Zolzer, Udo. *Digital Audio Effects.* 2011. John Wiley and Sons, Ltd.