**Lab Signal Generation Periodic and Pseudo Random Tutorial**

Today lab is all about generating a signal with the DSP to put out the D/A converter. We divide the generated signals into 2 major categories. Periodic like a sine or triangle wave and random which is essentially noise that is not correlated to itself.

**The periodic signal**

The periodic signal: is a little easier to examine.

**In the analog world,**

where is the amplitude of the sine wave, is the frequency of the sine wave in radians per second, and is the offset angle. The continuous time variable, , allows the argument to be any value. The sine wave is periodic over 2π radians or 360°. can be any value.

**In the DSP world,**

We have a major constraint. The sine wave can never be the sample frequency divided by 2 or higher, which means We also no longer have a continuous time variable. The sample frequency dictates the smallest time increment:

In some ways this seems limiting. In other ways it seems the choices we have are finite.

**1. Chapter 5.1 in the Textbook**

Read this section to understand the phase accumulator and how to use it to make a sine wave of any arbitrary frequency between 0 and .

**2. Chapter 5.2 in the Textbook**

It describes the WinDSK8 code for generating signals and of course you can generate some signals.

**3. Chapter 5.2 in the Textbook**

It has the MATLAB implementation of using a phase accumulator to generate a sine wave. Since you will be doing exactly that review and understand how this works.

**4. First Assignment: Problem 3 in Chapter 5.6: Follow-On Challenges**

The table lookup technique actually only needs to define of the table since the sine and cosine functions are symmetric functions (only of the 0 to 2π table is unique). Design and implement a real-time program to take advantage of this symmetry.

IMPORTANT NOTE: Read “Generating a Sine Wave using a Lookup Table” from the website now because it tells you how to do this the easiest way.

**(1) Create a new CCS project for Table Lookup**

**a. Notice while you are creating a new CCS Project:**

Variant: Generic C674x Device

Output format: eabi(EFL)

Linker command file: Book3rdEdition/code/common code/link6748.cmd

Project templates and examples: Empty Project

Add the common files that are needed.

Add the configuration files to your project.

**b. Go to** Book3rdEdition/code/chapter 05/ccs/sigGenTable, add **StartUp.c** and **tableBasedSinGenerator\_ISRs.c** to your project.

**(2) Modify the tableBasedSinGenerator\_ISRs.c**

Number of Entries: 17

Desired Frequency: 1kHz

Put the 16th entry at

SineTable[i] = sinf( i\* (float)(6.283185307 /4\*NumTableEntries));

On the left channel repeat the code, and add linear interpolation when the angle argument falls between integer values to your right channel. This means if the angle argument is 4.632 then you get the sine value for 4 and add to it the linear interpolation of the amount between the sine for 4 and 5. The value to add to sine(4) is [sine(5) - sine(4) ] \* 0.632.

**(3) Compare how these 2 sine waves sound.**

a. Compare them spectrally especially the distortion. Use the oscilloscope FFT function to analyze the harmonic distortion.

<https://www.youtube.com/watch?v=oRf-IpG6XAw>

<https://www.youtube.com/watch?v=6-zEHn-MrO8>

b. You can use RTSPECT if you have your own computer. This is software the PC can run. There a link for downloading the RTspect:

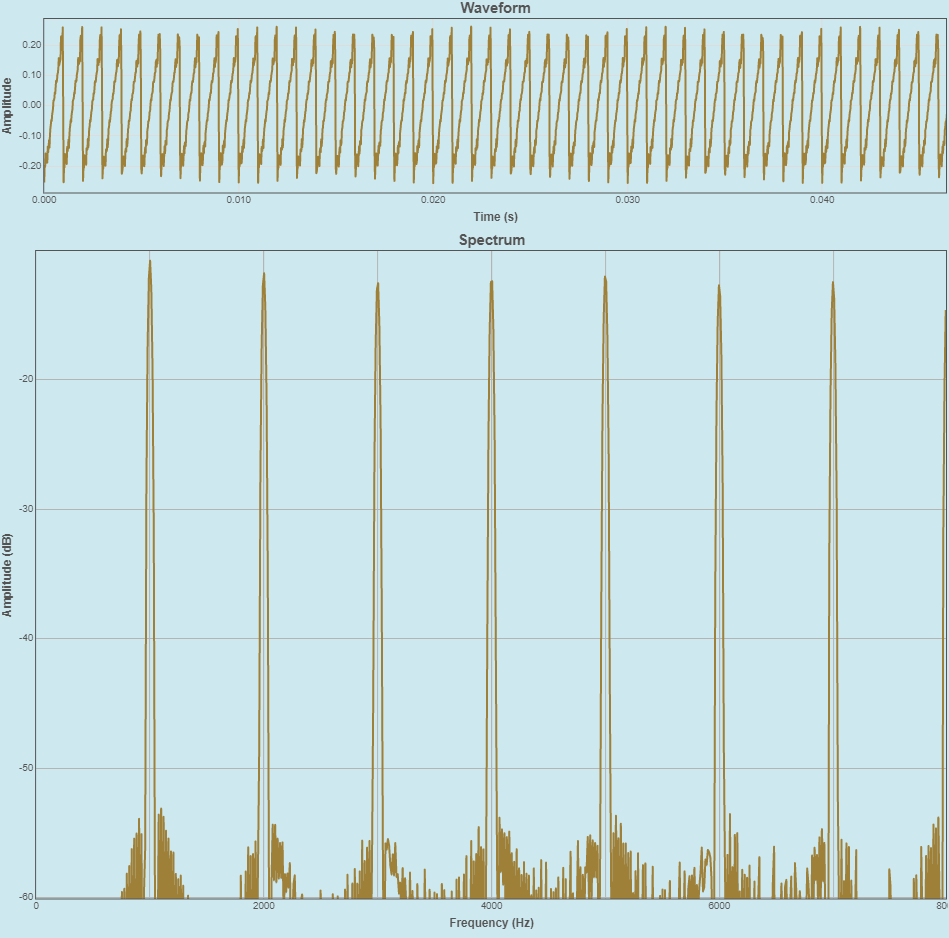
<https://www.phon.ucl.ac.uk/resource/sfs/rtspect/>

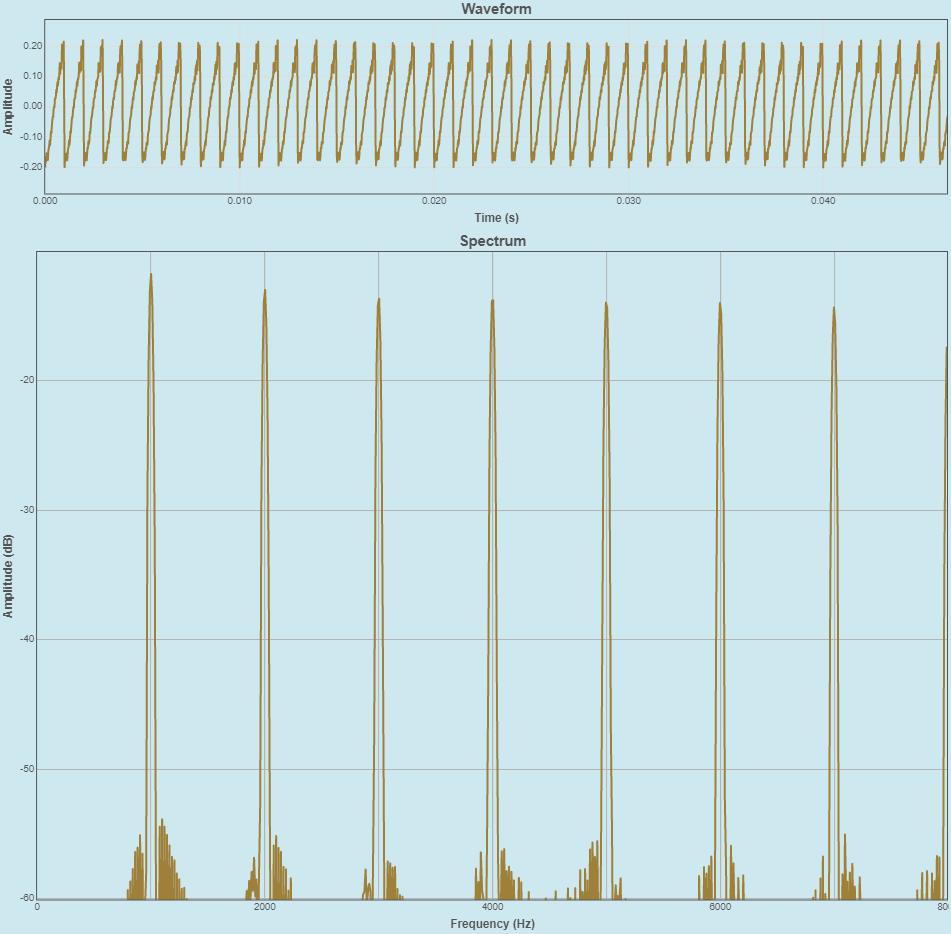
And by the link below you may use the RTspect online:

<https://www.speechandhearing.net/laboratory/rtspect/>

Connect the output of the DSP to the line in of the PC (Blue Jack) and RTSPECT will display the spectrum of the 2 signals. You are welcome to find some other spectrum analysis software if you want. How much distortion does each sine wave have?

You are supposed to submit the whole project for the 1st assignment.





The second figure shows the result of implemented filter. The second one has less noise than the first one while all other remain the same.

**5. Second Assignment: Pseudo random noise generator**

**(1) Please read section 5.5 of the book**. It will shed a lot of light on the subject. Notice that I refer to the generator as pseudo random noise generator. Whether it sounds like noise or more periodic depends on how long the sequence is that generates the sequence. Let it be understood that all of the sequences repeat. You can hear them repeat if they are ‘short’ compared to your listening interval. They sound random if the listening interval is ‘long’.

The pseudo random sequence generator you will code is a binary shift register. Each Ts the values in the register are shifted 1-bit position. It doesn’t matter if they are shifted to the left or the right. You must fill in the empty position created by the shift with a bit value that is calculated by an XOR (mod 2) value of the sum of some of the bits from within the sequence. The bits that get added together are very specific. If you use the correct bits then the sequence won’t repeat until all combinations of the shift register values have been realized. This excludes all 0’s. That one is impossible to get any value other than 0 with the XOR summation. The correct summation results in what is commonly referred to as a maximum length sequence, MLS. The Wikipedia page on Linear Feedback Shift Registers presents the correct bits to XOR various length shift registers to make them maximum length.

**(2) Create a new CCS project for Pseudo random noise generator**

a. Notice while you are creating a new CCS Project:

Variant: Generic C674x Device

Output format: eabi(EFL)

Linker command file: Book3rdEdition/code/common code/link6748.cmd

Project templates and examples: Empty Project

Add the common codes that are needed.

Add the configuration files to your project.

b. Go to Book3rdEdition/code/chapter 05/ccs/PN, add **StartUp.c** and **ISRs\_LFSR.c** to your project.

**(3) Modify the ISRs\_LFSR.c**

The code given for CH5 has a length 16 shift register pseudo random generator. Implement this code with an addition. The code generates a sequence of bits. We will use the generated bit each Ts to output a value to the D/A’s that we can hear. 1 lsb is not really audible. So your added task is to put out a constant positive and negative value based on a 1 or 0 bit value.

Make the constant value between INT (0x0001 and 0x7FFF) for the bit = 1 and between INT (0xFFFF and 0x8000) for the bit = 0. Remember 0x0001 is too small to hear and 0x7FFF will be way to loud. Be careful.

**Hint:** The range of negative values for a 16-bit two's-complement number is -32,768 (0x8000) to -1 (0xffff). Zero is 0x0000. The range of positive values is one (0x0001) to 32,767 (0x7fff).

When this works you need to generate a different sequence for the right channel. Make this sequence a 7 bit long linear shift register.

The equation for a 7 bit MLS is given on Wikipedia.

Hint: look up ‘Table of Linear Feedback Shift Registers’on google.com

With both sequences running listen to them and see if you can hear a difference. Does 1 sound more periodic than the other?

You are supposed to submit the whole project and a word/txt/pdf for the 2nd assignment.

**For this week, turn in the projects for the first and second assignment.**

**6. A new pseudo random value generator:**

You can also generate the new pseudo random value with a subtraction. This just gives a different look up table for the amplitude. -1, 0 and 1 instead of 0, 1 and 2.

