Fundamentals of discrete-time signals and systems

442.003 Digital Signal Processing, Laboratory
Winter Term 2023/24
Signal Processing and Speech Communication Laboratory
`www.spsc.tugraz.at`

Last updated: November 3, 2023

Required equipment:

- PC with netbeans & MATLAB installed
- Raspberry Pi
- Oscilloscope Agilent 54622D
- Signal generator Agilent 33120A
- Headset
- Cables

1 Practical part

Practical part of the work consists of several tasks that should be performed on Raspberry Pi and some MATLAB simulations. It includes:

1. Frequency response of the embedded digital system.
2. Build-in ADC(DAC).
3. Aliasing and quantization effects.

Experiment 1:
The DSP output noise measurement

1. Start netbeans and load the `Unit1/Exp1` project. Build the program and run it. Ensure that target host is set to the Raspberry Pi!
2. Connect the output of the DSP board to the input of the oscilloscope.
3. On the oscilloscope press $\boxed{\textbf{Edge}}$ button. Trigger on Ext or the channel corresponding to the output signal from the signal generator (which is not connected yet). Compress the horizontal axes to 200ms per division.

4. Press $\boxed{\textbf{Quick meas}}$ and set ***Source*** to the channel that corresponds to the output from the DSP board. By pressing corresponding soft key measure the RMS value of the DSP output signal. If the obtained value still significantly varies, try to obtain better estimate by compressing the horizontal axis a bit more. Under the assumption that the additive noise is a zero-mean process, the obtained value equals the estimated standard deviation σ_n of the additive noise present at the output of the DSP board and induced by the connecting cables.

5. Assume the maximum allowed output amplitude to be 1Vpp. Based on the DSP noise level calculate how many bits out of 16 are then masked by the noise.

Experiment 2:
Measuring the DSP antialiasing filter frequency response.

1. In netbeans, open and load project file `Unit1/Exp2`. Rebuild it. Make sure that the program was loaded into DSP's memory (put attention to the target host!)

2. Connect the output of the signal generator to the input channel of the DSP board.

3. On signal generator set the waveform to a sinusoid by pressing $\boxed{\sim}$ key. On the signal generator, set the input frequency to 100Hz and amplitude to 1Vpp (assuming the signal generator was set up for the input with the high impedance). Press $\boxed{\textbf{Freq}}$ again.

4. In netbeans choose ***Debug*** →***Run*** (or press F5; see further information in).

5. Make sure that the oscilloscope uses the external trigger. To check it, press $\boxed{\textbf{Edge}}$ button in the trigger section of oscilloscope controls. On the oscilloscope display the check mark should be at ***Ext***.

6. Switch off the oscilloscope input channel that corresponds to the signal generator. Press $\boxed{\textbf{Math}}$ button and choose ***FFT***, then ***Settings***. Set ***Source*** to the channel that corresponds to the input for the DSP board, ***Span***=20kHz, and ***Center***=10kHz. You should see one peak at 100Hz. To make sure that the peak appears exactly at 100Hz use cursors. Press $\boxed{\textbf{Cursors}}$ and set ***Source***="Math". Press ***X1*** or ***X2*** soft key and turn ↻ knob to adjust the cursors. Press ***X Y*** soft key to change cursors to Y–axis and set up ***Y1*** to the noise level around the peak.

7. By increasing the frequency of the sinusoid from 100Hz, find the frequency at which the amplitude of the peak disappears below the noise floor. This value stands for the highest harmonic that passes through the DSP.

8. Set up the signal generator to produce a sweep between 50Hz and the frequency you estimated in the previous task + some extra bandwidth (just try several different values (on the order of kHz) to obtain a good visual result). By changing the vertical position of the waveform on the oscilloscope, place a ground marker at the bottom of the screen to make only half of the waveform visible.

9. Sketch the "end-to-end" frequency response from the analog input to the analog output of the DSP.

10. Determine the following values: width of the passband region W_{pass} [Hz] (measured as part of the filter's frequency response that lies within 3dB from the maximum response), maximum passband gain G_{max}^{pass} [dB], and roll-off rate R_{roff} [db/Hz]. Note, that the frequency response you see on the oscilloscope's screen is not in dB scale and the time axis also has to be converted into frequency scale (according to the start and stop sweep frequencies set on the signal generator). Make sure you re-label the axes properly before computing the values.

Experiment 3:
Transfer characteristic of the DSP's ADC and study of the quantization noise.

1. In netbeans, stop a program and close all open projects, if any.

2. On the signal generator press the 'Ramp-shape' button, set frequency to 350Hz and amplitude to 1Vpp (assuming the signal generator is set up for the input with the high impedance).

3. Load `Unit1/Exp3` project following the known procedure.

4. In netbeans, unwrap the project list and open `main.cpp`.

5. By default, the loaded program *emulates* a 1 bit quantizer. As the result, the oscilloscope will display the original ramp signal from the signal generator and the corresponding 1-bit quantized waveform at the output of the DSP. You can remove some noise by enabling averaging in oscilloscope. For that just press **Acquire** and then select the ***Averaging*** option in the menu.

6. Take a look in the callback function `static int paCallback` contained in `main.cpp`. This function is evaluated once the input frame is filled with `FRAMELENGTH=256` sampling points. Go through the implementation which calculates the total signal's power `enSig` and the arising quantization noise power `enErr`. What is their unit (Volt^2, Volt$^2/f_s$ with sampling frequency f_s, Watt, ...)?

7. Using these plots, determine the amplitude of the input signal that triggers the ADC to change its state. Plot the appropriate ADC transfer characteristic assuming a two's complement code.

8. Similarly, plot the ADC transfer characteristic for 2-bit quantization. To change the quantization level, open the `main.cpp` file, find the `short MASK=...` line, and modify it according to the remark lines provided in the same file.

9. Estimate the Signal-to-Quantization-Noise Ratio (SQNR) that corresponds to quantization levels of 1−, 2−, 3−, and 4-bits. Use the values forwarded in `enSig` and `enErr` variables.

10. Plot the SQNR as a function of the number of bits. Predict how the SQNR will change for 5,6 and more bits. **Hint:** To do that, plot the SQNR values obtained in the previous experiments and fit a straight line to these points (linear regression). By extrapolating the line beyond the 4-bit value we can predict the SQNR behavior for higher number of bits.

11. Disconnect the signal generator and the oscilloscope from the Raspberry Pi. Connect a CD player or computer's sound card line-out to the 3.5 input on RPi. Connect the headset to the corresponding plug on the RPi. Alternatively, you can use any other sound source available to you.

12. Using the headset listen to the output sound that corresponds to the $1-$, $2-$, $3-$, and 4-bit quantization levels. Beware of the high volume of the output signals!

A Credits

This document was authored and/or adapted by Dmitriy Shutin and Josef Kulmer.