RILEY POOLE
ME 477 – LAB REPORT 06
WINTER 2019

I. DESCRIPTION

The purpose of this program is to create a general-purpose discrete filter. It will accept an analog input from a function generator connected to the ADC chip on the MyRio and output a response from the DAC. It will also be able to take measurements of both input and output and export them to a .mat file for processing. The program begins taking data as soon as it is launched and stops when the backspace button is pressed. The timing for the discrete filter is done with a clock interrupt rather than the simple wait() function used in previous labs.

Data processing includes comparing the step response of the filter to theoretical calculations as well as the magnitude and phase differences.

THE MAJOR TASKS PERFORMED BY THIS PROGRAM WERE

- begin a loop terminated by the user's input of a backspace key
- measure the input at the ADC
- calculate the next output from the biquad cascade and store the result
- output the cascade() to the DAC
- when the backspace button on the keypad is entered, end the program and save the results to a .mat file

THE LIMITATIONS OF THIS PROGRAMS CAPABILITY ARE

- the numerical accuracy of the floating-point values is likely higher than the accuracy of the measurements due to quantization errors
- there is no low pass filter on the input which could be adding high frequency noise to our input signal
- no way to save multiple trials without restarting the program
- user must set the name of the .mat file before it is compiled
- precision is only as good as the ADC and DAC which are both much slower than the CPU
- filter parameters must be calculated outside of the program

II. EXPLAIN ANY ALGORITHMS

The filter is implemented as a biquad cascade to minimize rounding errors. Biquads are stored as structures of 11 components. Six filter parameters, the last three inputs, and the last two outputs. Timing is accomplished with a timer interrupt. There are two threads. One to set up the interrupts and wait for the user to terminate the program with input to the keypad, and another to handle the analog input/output and calculate the biquad cascade.

Cascade() implements the input/output for an arbitrary order transfer function. To be used another program is used to break the transfer function into second order biquad sections. Then these sections and using their coefficients to calculate intermediate outputs which are then fed into subsequent sections. This allows a chain of outputs and is referred to as a "serial-input biquad filter." Cascade() is a simple function that essentially loops through the biquad sections calculating the outputs and updating the member values. Cascade is also implemented with pointers to make the execution faster without the second dereferencing caused with using an array "[]" notation.

The filter calculations and the keyboard interrupts are handled in separate threads to prevent delays and interference. The program also uses incremented buffers for the storage of data for output to Matlab.

HIERARCHICAL STRUCTURE

- **main**
  - MyRio_Open
  - MyRio_IsNotSuccess
  - printf_lcd
  - Irq_RegisterTimerIrq
  - pthread_create
  - getkey
  - Irq_UnregisterTimerIrq
  - MyRio_Close
- **Timer_Irq_Thread**
  - AIO_initialize
  - Aio_Write
  - Aio_Read
  - Irq_Wait
  - NiFpga_WriteU32
  - NiFpga_WriteBool
  - cascade
  - Irq_Acknowledge
  - pthread_exit
- **cascade**
  - Aio_Read
  - SATURATE -> macro

III. TESTING

1. Test that the program can act as a discrete filter
    a. Set the function generator connected to the ADC to a 60 Hz sinewave with an amplitude of 5 V.
    b. The correct frequency response of our filter should be a magnitude of -10.93 dB.
        i. Check that the output voltage is roughly 1.4 V to meet the magnitude requirements
    c. The correct phase shift of the filter should be -194°.
        i. Check that the time difference between the two signals is roughly 8.6 ms to meet the phase requirements
2. Test that the program can output a .mat file
    a. Change the title of the matlab file in openmatfile() to “Test.mat”
    b. Run the program and press <backspace> on the keypad to end the program and save the results
    c. Open Test.mat to see that the input and output have been recorded
    d. This process can also be done with the 60 Hz sample signal above and compared against the theoretical response with lsim().

IV. RESULTS

HOW SUCCESSFULLY THE PROGRAM RUNS AND UNSOLVED PROBLEMS

There are not significant issues with the program. It can use multiple biquads to calculate the response of an arbitrary transfer function and take in data quickly through the ADC. An inherent problem in the design of our system is that it does not provide protection against under-sampling and aliasing. There is no hardware low pass filter to help with this.

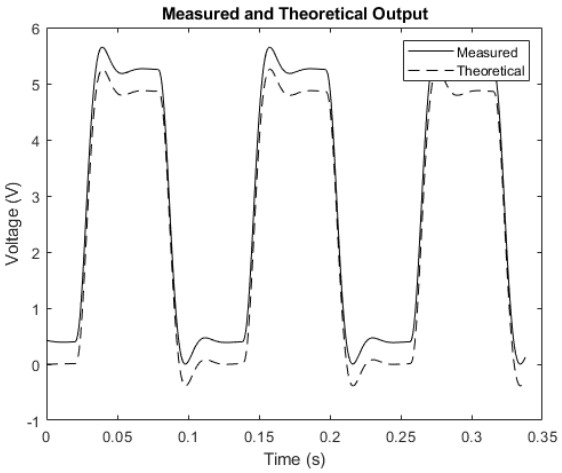
SPECIFIC QUESTIONS IN THE ASSIGNMENT

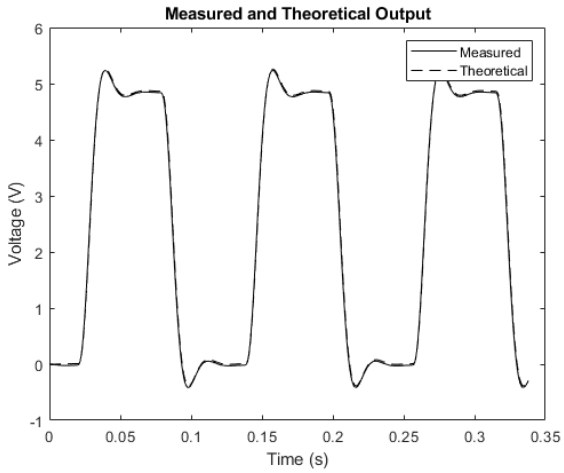
Record the Amplitude and Phase

Frequency (Hz)	Measured Mag (dB)	Theoretical Mag (dB)	% Error
5	0.00	0.00	-
10	0.00	0.00	-100.00
20	-0.28	-0.07	309.00
40	-4.08	-3.01	35.62
60	-12.04	-10.93	10.16
100	-22.50	-23.89	-5.84
140	-32.96	-32.65	0.95
200	-45.00	-41.94	7.30

Frequency (Hz)	Delta t (sec)	Measured Phase (°)	Theoretical Phase (°)	%Error
5	7.6E-03	-14	-14	-4.7
10	9.6E-03	-35	-29	19.3
20	9.0E-03	-65	-60	7.5
40	1.0E-02	-148	-135	9.9
60	9.0E-03	-194	-186	4.4
100	6.3E-03	-226	-223	1.5
140	4.9E-03	-246	-237	3.9
200	3.6E-03	-259	-247	5.0

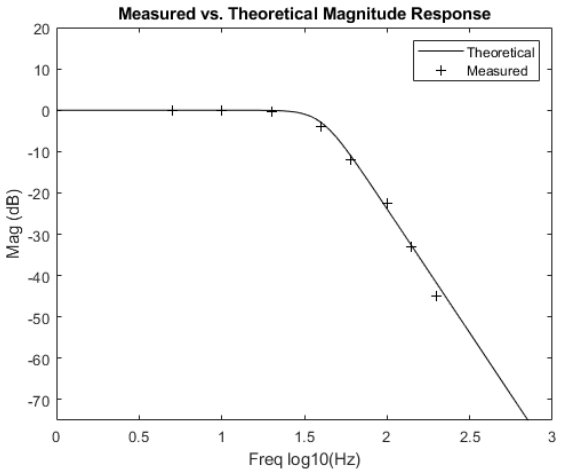
Step Response





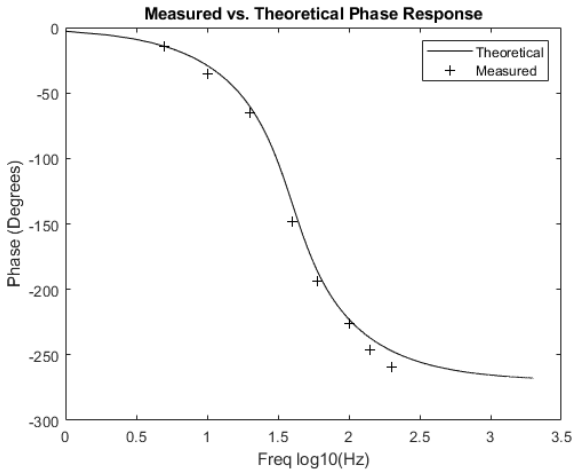
The first figure is the raw measured response versus the theoretical. Their shapes are nearly identical indicating good performance of the discrete filter, but there is an initial offset error in the response due to the initial conditions of the Matlab functions. When eliminated brings the theoretical and the measured into close agreement as shown in the second figure. Averaging the errors between the two graphs indicates an error of less than 0.7%.

Magnitude Response



The magnitude response matches well with the theoretical calculations. Beyond 40 Hz the errors are less than 10%. It is also surprising that it can carry the correct roll-off rate for more than a decade. This indicates that the filter is accurate at least for these frequencies. It also shows an excellent precision as well. The biquad filter algorithm can calculate and output magnitudes correctly. Below 40 Hz there are significant amplitude errors near 100% but these are drowned out in in the log scale. So, while apparently accurate on a log scale there are still some significant amplitude issues in the low frequency spectrum.

Phase Response



The phase response of the filter matches well with errors within 10%. The deviation is greatest in the high frequency spectrum. It appears that the filter may not be able to reach the limiting value of -270° rapidly enough as indicated from the deviation from the theoretical line. Phase shifts at higher frequencies require very rapid switching as the period only lasts for a few milliseconds. The DAC probably has an output frequency on the order kilohertz and controlling the phase shifts at an order of microseconds may become an issue, and this explains the higher deviation from the theoretical values at higher frequencies.

POSSIBLE IMPROVEMENTS AND EXTENSIONS

With the addition of a few more interrupts the program could measure the frequency and phase shift and then save this data to the .mat file. Another improvement would be to allow the user to set the filter type [low, high, pass, reject] and enter the cutoff frequency with the keypad. Then have the program calculate the correct coefficients. It would also be nice to have the LCD tell the user when the Matlab data has finished being collected.

V. ALGORITHMS AND PSEUDOCODE

```
main()
      open the myRio session
      set up the IRQ channel
      register the timer interrupt
      create the new thread
      enter a loop ending only when a backspace is reieved
            enable the timer IRQ
      signal the timer thread to terminate using the IRQ threadRdy flag and
wait for it to terminate
      unregister the timer thread
      close the myRio session

timer_irq_thread()
      define the biquad structure
      cast thread resource back from void
      initialize the analog input/output and set the analog output to 0V
      while main has not indicated to stop
            get ready for the next interrupt
                  wait for the IRQ to assert
                  write to the timer set time the register
            read the analog input to obtain the current input value
            call cascade to calculate the current value of the output
            send the output value to the analog output
            acknowledge the interrupt
      save response to matlab file

cascasde()
      loop for the number of biquad sections
            calculate the current biquad
            increment the current biquad
            increment to the next biquad
            chain the outputs
            saturate the value if needed
```

VI. FUNCTIONS OF THE PROGRAM

- **main()**
  - Prototype:
    - int main(void)
  - Purpose:
    - Sets up threads and interrupts; Procedural section for testing the filter.
  - Rationale for creation:
    - Need a procedural section to layout testing environment and handle threads and interrupts.
  - Inputs and parameters:
    - No inputs or parameters used
  - Return:
    - “Running” on LCD indicates filter is running
    - “End” on LCD indicates filter has stopped
    - status if there is an error
- **Timer_Irq_Thread()**
  - Prototype:
    - void* Timer_Irq_Thread(void* resource)
  - Purpose:
    - Code to run during the timer interrupt
  - Rationale for creation:
    - Need a procedural section to perform the biquad algorithm, output the values to the DAC, and save the responses to a matlab file
  - Inputs and parameters:
    - void* resource - a pointer to void that is actually a pointer to a ThreadResource structure
  - Return:
    - Sets the output on the DAC based on the previous values of cascade()
    - saves the responses to a matlab file with pointers
    - void* - No returns
- **cascade()**
  - Prototype:
    - double cascade(double xin,struct biquad* fa,int ns,double ymin,double ymax)
  - Purpose
    - Calculate the next output
  - Rationale for creation:
    - Need a procedural section to layout testing environment and handle threads and interrupts.
  - Inputs and parameters:
    - No inputs or parameters used
  - Return:
    - “Running” on LCD indicates filter is running
    - “End” on LCD indicates filter has stopped
    - status if there is an error