ECE 437 Project Report

Brandon Dotson

12/8/17

Purpose

The purpose of this project was to code an implementation of a Direct I filter by using MATLAB. Additionally, students were tasked with calculating the values of y(n) using some given MATLAB functions and a few specifications.

Implementation

The different methods for implementing my own functions are described in the following sections:

Writing the iir() function

I started by defining all the arrays required for the filter. These were "a" for the a array, "b" for the b array, x_array for the calculated b_k * x(n-k) values, and the y_array for the a_k * y(n-k) calculations. It should be noted that x_array and y_array were initialized to zero. The next step was looping through the x(n-k) loop to obtain values for the x_array vector. The looping for y_array was next, and since all the initial values for y(n) were zero, I summed over an empty vector during the first iteration. I added these two sums to generate y(0). From here, the loop repeated n times, depending on my value for n.

Calculating y(n) method 1

Unlike the previous function, writing the program that relied solely on the impz function was much more straightforward. Before anything could be written, I needed to do some algebraic manipulation to obtain the X(z) H(z) form. This is shown below in Equation 1.

$$\frac{1}{1-a z^{-1}} \quad \frac{1-r\cos(\omega) z^{-1}}{1-2rcos(\omega) z^{-1}+r^2 z^{-2}} = \frac{1-r\cos(\omega) z^{-1}}{1-2rcos(\omega) z^{-1}+r^2 z^{-2}-az^{-1}+az^{-2}2rcos(\omega)-r^2 az^{-3}}$$

Equation 1: X(z) H(z)

Once this was complete, the coefficients of the numerator were assigned to a vector called “b” and the denominator was assigned to a vector named “a”. The impz( ) function was then used to obtain y(n).

Calculating y(n) method 2

In this case, x(n) and h(n) were needed before anything could be done. The coefficients for both X(z) and H(z) were recorded into two sets of “a” and “b” variables. The impz( ) function was used on both sets of data to obtain the values in the time domain. Once this was complete, the convolution was taken using the conv ( ) function.

Calculating y(n) method 3

This was the most involved of the three calculation problems. Like the previous iteration, I started by making two sets of “a” and “b” variables that represented the coefficients of X(z) and H(z). Following this, I used the impz ( ) function to obtain approximations of x(n) and h(n). The next step was to use the Discrete Fourier Transform on both of these. Lastly, I multiplied the two results together and took the inverse Discrete Fourier Transform to finally obtain y(n).

Results

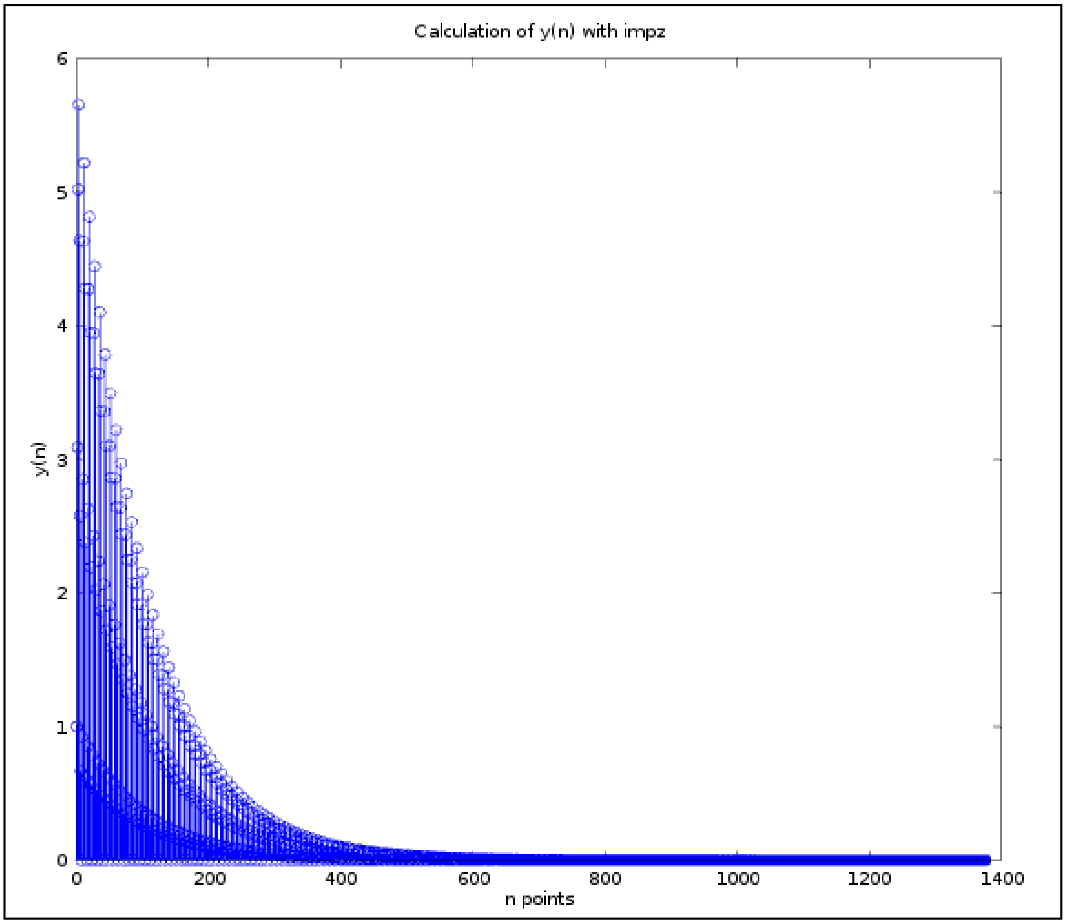

Figure 1: Calculation of y(n) with impz

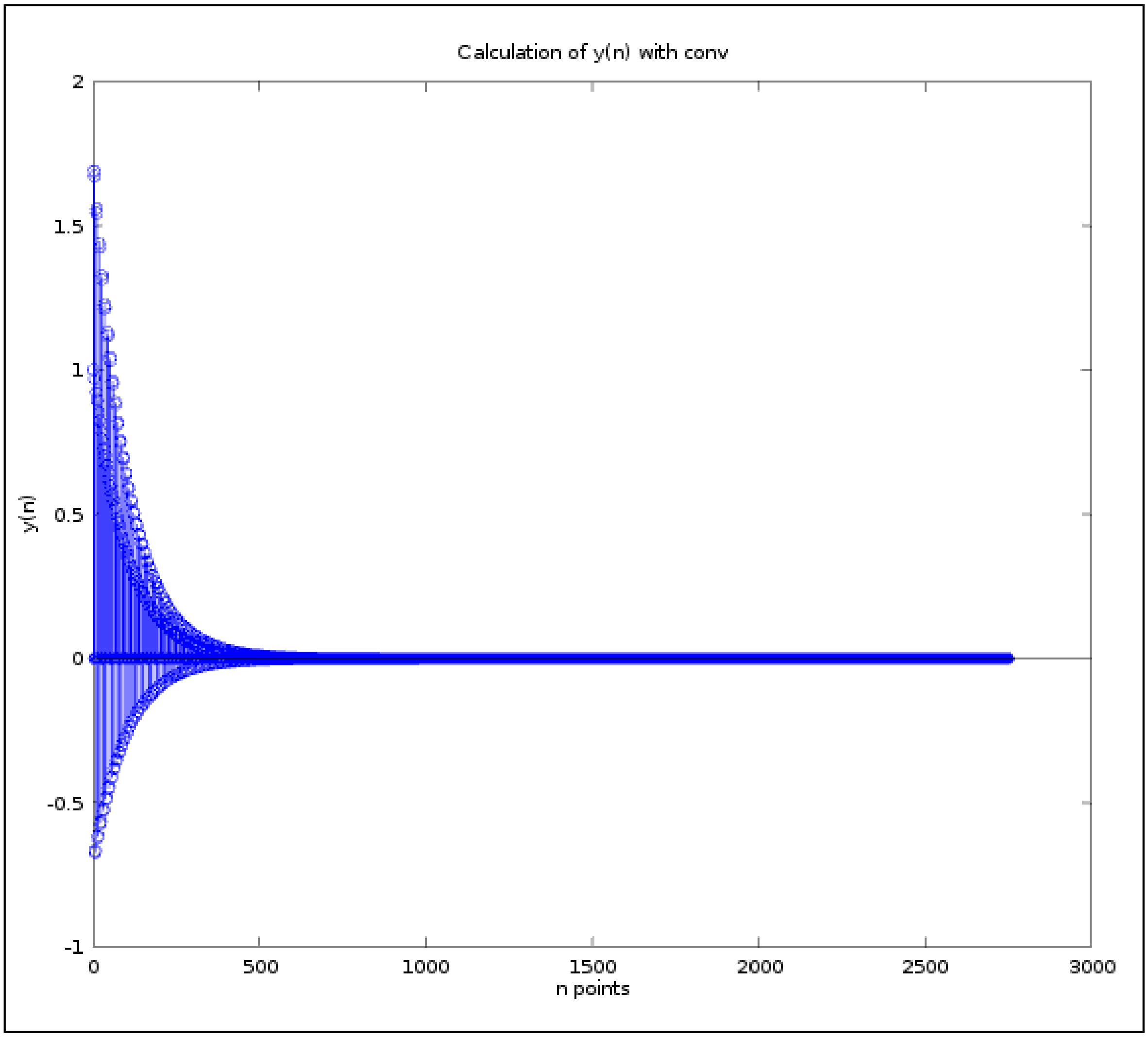

Figure 2: Calculation of y(n) with conv

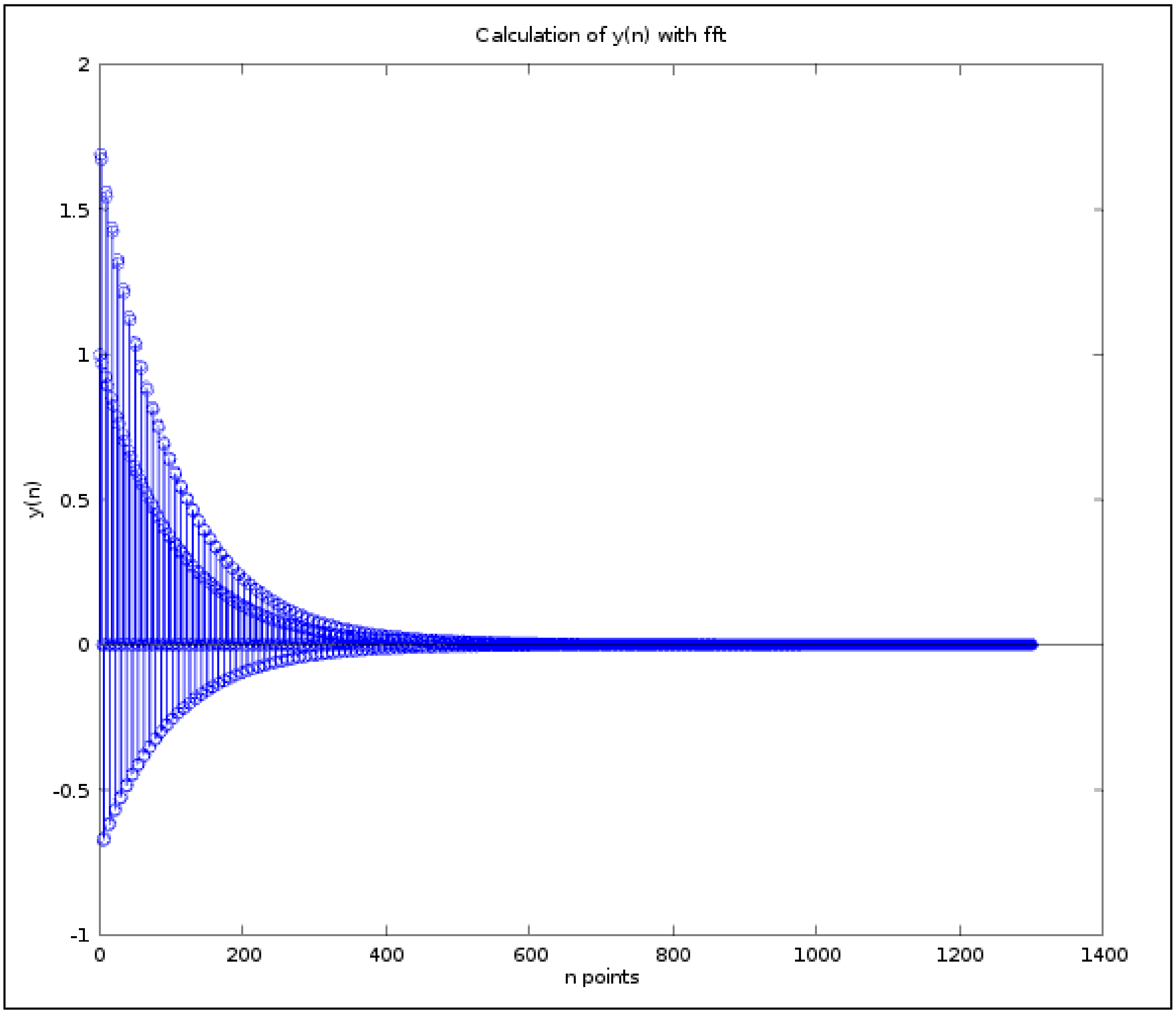

Figure 3: Calculation of y(n) with fft

Discussion

In this computer exercise observe and comment how changes in the length of calculated x(n) and h(n) (e.g. experiment with N=8, 16, 32, 64, 128 and 256 samples), changes the calculated output y(n) .

By increasing N samples by the listed intervals, the output y(n) followed suit by increasing in resolution. That is, the number of points received as output directly corresponded to the numbers used for N.

Justify difference between results obtained by impz, vs. conv vs. fft vs. your own iir() function, i.e.. for which samples in the result y(n) will the discrepancies/errors be the highest and why?

The highest number of discrepancies came from the fft approach, as there were complex values appearing in the results. So many approximations and domain changes negatively affected the results. This also happened in the conv( ) method but not as severely. I attribute this to the taking the approximations with impz( ) while also transforming them again.

Which method(s) produce the result closest to the actual y(n)?

The methods that produce the results closest to the actual y(n) is the impz( ) (Method 1).

Is there a K where both conv and fft provide the same result? Why?

The conv( ) and ff( ) functions will provide the same result if K is set to be 2750. This is because that is the number of points that were automatically generated when the conv( ) method was used.

Conclusion

For this project, I wrote several different functions to implement a Direct I filter and various approaches to calculating y(n). My actual iir( ) function did not perform as intended due to difficulties with the indexing, but the other three functions worked as intended.

Appendix

Source Code

```
function [y_n] = iir (x_n)
  M = 2; #Adjusted to be one less than normal because of non-zero indexing of array
  N = 3;
  n = 0;

  #Calculating x(n)
  #a1 = [1, -.99];
  #b1 = 1;
  # [h1, t1] = impz(b1, a1);
  # x = h1;


  #Used for H(z) coefficients
  a = [1, -2*.99*cos(pi/4), .99^2];
  b = [1, -.99*cos(pi/4)];


  # x
  x_array = zeros(1,15);
  # y
  y_array = zeros(1,15);


  for n = 0:9 #Calculate terms from n = 0 to n = 10.

    x_sum = 0;
    y_sum = 0;

    ## Computing b * x(n-k) terms
    for  k = 0:M  # Adjusted loop iteration value
      x_array(k+1) = b(k+1) * x_n(n - (k-(2k+1))); #Must add 1 to adjust the indices (arrays not zero-
indexed)
      x_sum = x_sum + x_array(k);
    end

    ## Computing a * y(n-k) terms
    for  k = 1:N
      if n-k >= 0 #Check if next y(n-k) will be initial condition or not (aka: is n-k positive?)
        y_sum =  a(k) * y_array(k); #Calculates bi * x(n-M). Required index adjustments
      end
    end

    y_array(n+1) = x_sum - y_sum;
    out = ['y(n) =  ', num2str(y_array(n+1)), ", n = ", num2str(n)] ;
```

```
    disp(out);
    disp(" ");

   end


endfunction
```

Program Output

The program does not output properly. I could not figure out the indexing after generating x(n) for the function. See below:

```
iir(x)
error: iir: subscript indices must be either positive integers less than 2^31 or logicals
error: called from
    iir at line 33 column 18
```