ILLINOIS INSTITUTE OF TECHNOLOGY
ECE 437/436 DIGITAL SIGNAL PROCESSING
Computer Project:

Objectives

The major goals of this computer project, using MATLAB (or GNU Octave), are:
Part I

1. Implementation of an IIR filter in Direct Form 1
2. Calculate response of the filter to a signal by calling the function iir()

Part II

1. Calculate system response to a signal using impz
2. Calculate system response by convolution via conv
3. Calculate system response using discrete Fourier transform (DTF) using fft and ifft.
4. Compare and discuss the differences between different MATLAB functions (impz, fft ,conv) and the implementation of iir DF 1 filter

Problems:

Part I

Write a function that implements the Direct Form I of an LTI system given by:

$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \quad (M < N)$$

The code can be in C or Matlab/Octave. **IMPORTANT**: When using MATLAB/Octave, for this part - you may only use for/while loops and addition/multiplication on individual elements of the array. **In Part I - using built-in MATLAB/Octave functions or syntax that operates on the entire array at once** (such as using statements: w=w(2:N), or: y=x .* b) **is not allowed and will count as zero points**! E.g. if you want to copy elements of the array, you must use a "for-loop" or a "while-loop" and access individual elements of the array that you are trying to copy. While MATLAB is a good simulation tool, in most cases, MATLAB code is not used in practice in real-time applications. This exercise is meant to get your code as close to a practical real-time implementation of an iir filter as possible.

Assume the function takes input sample x(n) as its sole input and returns y(n) as its sole output, i.e. it has the following C-style declaration
"double iir(double x_n);" or the MATLAB equivalent

```
function y_n = iir( x_n )
```

Assume that the following arrays/constants are defined as global (note: although M,N are defined below as 1 and 2 respectively, your function should be able to handle any value of M and N (assume M<N). Note also that elements of arrays a[] and b[] below are just examples) :

```
#define M   1               /* e.g. M=1*/
#define N   2               /* e.g. N=2 */
double a[N]={0.223, 0.111};      /* array that contains coefficients a1,a2,…aN */
double b[M+1]={1.0, 0.5};        /* array that contains coefficients b0,b1,b2,…,bM*/
```

or equivalently in MATLAB:

```
M=1;
N=2;
a=[0.223, 0.111]; % just some example values
b=[1.0, 0.5];       % just some example values
global M N a b
```

You will need additional arrays and variables for intermediate results and to store internal states of the filter. You can define those as global. Assume zero-initial conditions.

For debugging purposes, you can verify the correct operation of the function by using some known (pre-calculated) system and it's impulse response and print the values of the output of the function when x(n) = δ(n) for n=0,1,…,9 (i.e. call the function 10 times with appropriate arguments, print the first 10 output values and compare with the pre-calculated values). Assume the system is initially relaxed (zero init-conditions). NOTE: Style of your C/Matlab programming is irrelevant as long as the solution is correct.

In your report, submit the printout of the function + small program that calls the function – i.e. provides x(n) and prints x(n) vs. y(n) for n=0,1,…) in an appendix.

Extra credit:

iir() function can be written so that it avoids copying of elements of arrays/internal states (as this is inefficient in a practical real-time implementation). To avoid copying, consider using a circular buffer to store intermediate states in your implementation of the filter. I.e. when reading/writing values into the array and reaching the end of the array, wrap-around and start reading/writing the array from the beginning (a "modulo" operator (% operator in C) may be useful, or a simple comparison of an index and the size of the array).

Part II)

From class, we know that the output of an LTI can be obtained as:

$$y(n) = Z^{-1}\{H(z)\cdot X(z)\}, |z| \in ROC, \quad (1)$$

Also, using the convolution sum:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k), \quad (2)$$

y(n) can also be calculated using (Inverse) Discrete Fourier transform:

$$y_p(n) = IDFT\{DFT(x(n))\cdot DFT(h(n))\} = \sum_{l=-\infty}^{\infty} y(n-l\cdot K), \quad (3)$$

where DFT is *K*-point discrete Fourier transform (DFT) (and with the assumption of periodicity of the sequences with period K).

For the following input signal (given by its Z-transform):

$X(z) = \frac{1}{1-az^{-1}}$, ROC: |z|>|a|

Where a=0.99

The system is given by its' system function:

$$H(z) = \frac{1 - r\,cos(\omega_0)z^{-1}}{1 - 2\,r\cos(\omega_0)\,z^{-1} + r^2z^{-2}}, \qquad ROC: |z| > r$$

where $\omega_0 = \frac{\pi}{4}$ and r=0.99

Calculate $y(n)$ using different methods impz, conv and fft/ifft built-in MATLAB functions which correspond to Eqs. 1, 2 and 3, respectively. Also, calculate y(n) using your iir() function from part I.

Note that for fft/ifft method and conv you will need $x(n)$ and the system impulse response $h(n)$ in time domain. Approximations of both can be calculated by using the built-in MATLAB function impz. If the signal is infinite, you will have to approximate it by a finite number of samples, i.e. truncate it to a finite number of samples N – which has consequences.

The methods may, therefore, not be producing the same result, but the actual differences/errors may be small (so be careful to capture them). E.g. in the case of convolution we cannot do the infinite summation and in the case of DFT we are implicitly assuming a periodic signal. 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 $y(n)$.

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?

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

Also, explore how zero-padding in case of fft/ifft changes the result! First truncate the signal to N samples, and then zero-pad to create K-sample long signal (part of which is zeros). Use K-point convolution/ DFT where $K>N$ (Use $K=N$, $1.5N$ and $2N$-1). Is there a K where both conv and fft provide the same result? Why?

Ground Rules and Instructions for Submission

You must write your own programs, run your own simulations, and analyze and interpret your results. Submit the report electronically. Document any theoretical/analytical preparation you did (or simply refer to specific material in the textbook/lecture) as part of your design of your computer programs.

Present your results, in a form of a project report with **LESS THAN 10 PAGES**, compactly, clearly, and selectively. A report "stuffed" with redundant material will not qualify for a high grade. In addition to your report, submit all the programs you have written for the project, in one, separate, plain-text file, to course web site on blackboard.iit.edu. Do not include your computer programs with your report (except for part I – where the printout of function iirDF1 should be part of the report).

You may discuss the project with your classmates, but your effort must be essentially independent. Verification of independent effort will be performed, and plagiarism will be treated as academic dishonesty.

Figures (preferred) and tables should be numbered consecutively (Figure 1, Figure 2, etc.). The **text of your report should refer to figures and tables, before they appear in the document,** by number. For example: "The frequency response of the filter is shown in Figure 3. One can see ….".

Any figure or table without reference in the report text will be ignored, and the grade will be determined as if they were not there.

Each figure and table must have a caption. Each figure should include properly labeled axes (both).

Please use meaningful plots. There is no need to include plots for every single case (e.g. N=8, 16, 32, …) in your report. Include the most illustrative plots. In some cases, you may want to plot the difference between the "actual" signal closest to the true value of y(n) and the approximated version of y(n).

Note that in some cases you may have to adjust the MATLAB axes to plot the entire signal (not just zoomed-in portion of it).

ANSWER ALL QUESTIONS I EXPLICITLY ASKED!