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

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. Include your computer programs with your report at the end (as an appendix) (program pages are not counted towards the 10 page maximum).

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/or 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). As necessary, consider also plotting multiple signals in one plot etc.

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!

SUBMIT: 1) Project report in PDF format (with code in the appendix) 2) "m" files 3) wave files FilteredSignal_Part1.wav and FilteredSignal_Part2.wav

Objectives

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

1. Implementation of an IIR filter (DF2)
2. Analyze the filter type, frequency response, poles/zeros etc.
3. Calculate response of the filter to a signal by calling the function iirFilt()

Part II

1. Filter a long data sequence using DFT/IDFT (via fft/ifft) on blocks of input data
2. Filter a long data sequence using DFT/IDFT overlap-add method
3. Compare and discuss the problems/differences/advantages

Problems:

Part I

Write a function that implements an LTI filter using Direct Form II :

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{j=1}^{N} a_j y(n-j)$$

The code can be 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 is not used in practice in real-time applications or devices. This exercise is meant to get your code as close to a practical real-time implementation of an fir filter as possible.

Given is a "signal_plus_feedback.wav" file (represents an audio signal sampled at 48000 Hz corrupted by a feedback signal at a specific frequency). A MATLAB program "Project_Part1.m"is also given. **You must not modify "Project_Part1.m" and your function must work with Project_Part1.m as is.**

Your task is to write the function "iirFilt.m" that is called by the "Project_Part1" program which defines the filter coefficients of the filter that will eliminate the feedback, opens the audio file, and calls your function on a sample-by-sample basis. Notice that the function takes ONE input sample x(n) as its sole input and returns ONE output sample y(n) as its sole output, i.e. that it has the following MATLAB definition:

```
function y_n = iirFilt(x_n)
global a b M N
persistent w;
if isempty(w)
    w=zeros(1,N+1);
end
```

% FILL IN THE REST OF THE FUNCTION ….

Arrays b = [b0, b1, b2, b3, ..bM] and a=[a0,a1,a2,..,aN] contain the coefficients of the LTI system.

Array "w" represents the internal state of the direct-form-2 implementation of the filter. You may need additional arrays and/or variables. Assume zero-initial conditions.

Although N=M is set to 4 in this particular case, you should write your function so that it works with arbitrary N and M (assume N>=M).

For debugging purposes, you can verify the correct operation of the function by using some known iir-filter coefficients and the (pre-calculated) output of the filter to some specific sequence (perhaps delta(n)). Print the values of the output samples to verify validity. For example, the output of the function when x(n) = δ(n) for n=0,1,… Call the function "iirFilt" a number of times with appropriate arguments, print the output values and compare with the pre-calculated values. Assume the system is initially relaxed (zero init-conditions). NOTE: Style of your Matlab programming is irrelevant as long as the solution is correct.

In your report, submit the printout of the function. Also, submit the "FilteredSignal_Part1.wav" i.e. the audio file that is the result of filtering.

Useful MATLAB functions (use MATLAB help for detailed descriptions):
freqz, impz, zplane, fft, ifft, conv, …

Answer the following questions (provide plots if necessary):

1) Use freqz and zplane MATLAB commands to display the poles and zeros of the filter transfer function as well as the frequency response (vs. frequency) of the filter whose coefficients are given by arrays b and a in "Project_Part1.m". Find roots, zoom-in the figures to get better details …
2) What kind of filter is this?
3) What analog frequency/frequencies will this filter amplify/attenuate/pass thru? Explain / How/Why ?
4) What is the maximum attenuation (in dB) provided by the filter?
5) What kind of attenuation/gain does the filter apply to a discretized sinusoid with analog frequency 10kHz sampled at 48Khz?
6) Observe the maximum value(s) of the internal state "w" of the filter during filtering of the given audio signal "Signal_plus_feedback"? What is the maximum value of "w"? How does it compare to the values of the input and output signal?

Part II)

Due to possible stability issues with the fixed-point IIR filter implementation in Part I, it was decided to use an FIR filter instead. To achieve somewhat similar performance/levels of attenuation, a relatively high order FIR filter has to be used. The filter to be used has been designed as an FIR filter of length 257 samples (see the attached code).

The FIR filter is to be applied to real-time audio signals coming into the system (assume there is hardware/software preprocessing that performs A/D conversion and accumulates a block of input samples).

From class, we know that the output of an LTI y(n) can also be calculated using (Inverse) Discrete Fourier transform:

$$y_p(\mathrm{n}) = \mathrm{IDFT}\{\,\mathrm{DFT}\{\mathrm{x(n)}\}\,\mathrm{DFT}\{\mathrm{h(n)}\}\} = \sum_{l=-\infty}^{\infty} y(n - lN)$$

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

To achieve real-time requirements, the filtering is to be done using DFT/IDFT on blocks of data of length N (e.g. N=2048 or 4096 etc). An attempt at implementation is provided in "Project_Part2.m". Run this program …

- Listen carefully to the output of the filter (saved in the "Filtered_Signal_part1.wav"). Plot relevant parts of the signal (zoom-in if necessary) … What are the problems with this implementation?

- Apply the FIR filter b_fir in time-domain (you can use MATLAB command "filter" or even re-use your iirFilt() function by defining b=b_fir , a=[1] and setting M and N appropriately).

- plot the difference between the time-domain filtered output and the Project_Part2 DFT/IDFT based output (zoom the figure if necessary) … Where are the discrepancies located ? Why do they exist?
- Read chapter 7.3.2 in the textbook (or find other sources that describe block-based DFT/IDFT Overlap-save and Overlap-add method of filtering long sequences).

- Use "Project_Part2.m" as a starting point and modify it to implement overlap-add method of block-based DFT/IDFT filtering using correct block sizes. Verify that the output signal has no artifacts and produces correct output (e.g. by comparing the output to the output of the same FIR filter implemented in time-domain)

Questions:

1) Why would you want to use FIR instead of an IIR filter in general?
2) Analyze the FIR filter "b_fir" (freqz, zplane, …) and compare to the IIR filter in part I
3) Why is the block-based DFT/IDFT implementation considered here instead of FIR filtering in time-domain?
4) Describe the problem(s) with the initial implementation of "Project_Part2.m"? What/where are the artifacts i.e. discrepancies, and why do they appear in the output signal? You may want to analyze the first few blocks …
5) Would it help to simply zero-pad both the filter/input signal?
6) If the length of the FIR filter is L=257, discuss the size of the block N used for DFT/IDFT.
7) What are the benefits and drawbacks of increasing the block size N?
8) Explain the overlap-add method i.e. block-size and the method you are using to do the filtering. What block-size did you end up using?