

ECE-486 Laboratory 2, Spring 2009

Due Feb. 16

Objectives

- Implement real-time audio signal processing under Linux.
- Develop and test C subroutines for implementing FIR and IIR filters.

Assignment

TASK 1: Learn to write a “plugin” which is loaded and executed by the “dsp486” command. The “dsp486” command continuously captures sampled data from the “line-in” terminal of the sound card, passes the data through your “plugin”, and writes the result back to the sound card to drive the analog “line-out” terminal.

Check out audio-to-BNC cables from the course TA to allow connection of test equipment to the computer line-in and line-out connectors. Each group must sign out one set of cables, and use the cables for the remainder of the semester. Cables should be returned during your last group presentation.

Instructions on how to use the “dsp486” command can be downloaded from the course web site. We’ll go through this document during lab briefings on Feb. 2 and 3.

TASK 2: *FIR Filter Implementation:* Re-write and test your C-code implementations of discrete-time convolution from Lab 1 using a pre-defined function interface. The required function interface is defined in the header comments of the `fir_filter.c` and `fir_filter.h` files provided on the lab web site.

You are **not allowed** to change the format of any of the function calls. Complete the `fir_filter.c` and `fir_filter.h` files, and develop a separate routine to test your implementations.

TASK 3: *IIR Filter Implementation:* Write, compile, and demonstrate a “plugin” which implements an IIR filter in real-time on a Linux computer. Your IIR filter should be implemented using the “*Direct Form II*” structure.

Use a function interface similar to that used for the FIR filters of TASK 2. Use the following function prototypes:

```
IIR_PARAMS *init_iir(double *a, int N_a, double *b, int M_b);
double calc_iir( IIR_PARAMS *s, double x );
void destroy_iir(IIR_PARAMS *s);
```

In the above, $a[i]$ and $b[i]$ should contain the IIR filter coefficients a_i and b_i using the notation adopted in class. N_a and M_b are the *orders* of the denominator and numerator of the transfer function (corresponding to the N and M variables used in class). (Note that the arrays $a[i]$ and $b[i]$ must contain N_a+1 and M_b+1 values respectively.) Complete and comment the above subroutines using file names of `iir_filter.c` and `iir_filter.h`. Write a separate main program to test the routines, and (finally) use the routines in a real-time implementation.

It should be easy to change the filter order and coefficients for your real-time code. Your IIR and FIR functions should be modular, so that (for example) it's easy to implement multiple cascaded filters without duplicating large code segments for each filter.

Your functions will be graded on efficiency as well as whether or not they work. Do not waste instruction cycles! You are running in real-time, so your plugin routine has a limited amount of time to do signal processing. More efficient routines will allow implementation of higher order (and better quality) filters.

What should be handed in:

1. Hand in commented C source code listings, documentation showing how you tested your routines and results showing that they work. Program listings and documentations will be collected at the beginning of class on the due date.
2. Create a TAR archive containing **only** the source code files `fir_filter.c`, `fir_filter.h`, `iir_filter.c`, and `iir_filter.h`. The TAR file should be labeled `lab2.tar` and uploaded on the ECE 486 website. The TAR file must be uploaded before class (11:00 AM) on the due date.
3. Demonstrate your working routines during your scheduled group meeting time in the week of the due date.