332:521 – Digital Signals and Filters
Computer Experiment 2 – Due October 5, 2010

Please do the computer experiments described in Problems 4.8–4.12:

4.8 Filtering by convolution.
4.9 Block-by-block processing using the overlap-add method.
4.10 Sample-by-sample processing using a linear delay-line buffer.
4.11 Sample-by-sample processing using a circular delay-line buffer.
4.12 Sample-by-sample implementation of a delay using a circular buffer.

Although you may use MATLAB for all the experiments, the sample-by-sample processing experiments are more appropriately done in C because the C code parallels the hardware implementation of these operations on a modern DSP chip. If you choose to do Problem 4.9 in MATLAB, then write a function with usage:

```
y = blkfilt(h,x,L);
```

where **h** is the filter vector, **x** is an arbitrarily long input vector, L is the block length for partitioning **x** into blocks, and **y** is the overall convolution output vector.

As a starting point for your C programs, the required dynamic allocation of the filter coefficients may be done by the following program segment:

```
#include <stdlib.h>
#include <stdio.h>

void main(int argc, char **argv)
{
   FILE *fph;                                  filter file
   double *h, *w;
   int M, max = 64, dmax = 64;                 allocation for h and increment

   if (argc != 2) {
      fprintf(stderr, "usage: firfilt hfile < x.dat > y.dat \n");
      exit(0);
      }
   if ((fph = fopen(argv[1], "r")) == NULL) {
      fprintf(stderr, "can't open filter file: %s\n", argv[1]);
      exit(0);
      }

   h = (double *) calloc(max + 1, sizeof(double));      preliminary allocation

   for (M=0;; M++) {                            read h
     if (M == max) {                            reallocate h, if necessary
       max += dmax;
       h = (double *) realloc((char *) h, (max + 1) * sizeof(double));
       }
     if (fscanf(fph, "%lf", h + M) == EOF) break;
     }

   M--;                                        M is filter order

   h = (double *) realloc((char *) h, (M + 1) * sizeof(double));   final allocation
   w = (double *) calloc(M + 1, sizeof(double));                   internal states
```