Assignment 2

ES22E Software Development

0013679

School of Engineering, University of Warwick

04/12/01

Specification

- The program will determine the following responses of a recursive digital filter over a range of frequencies:

  Magnitude response: $M(\omega) = |G(z)|_{z=\exp(j\omega T)}$ and attenuation response,

  $-20\log_{10} M(\omega)$

  Phase Response: $\Phi(\omega) = \arg G(z)_{z=\exp(j\omega T)}$

  Delay Response: $\tau(\omega) = -\dfrac{\partial\Phi}{\partial\omega}$

- The transfer function of a digital recursive filter is:

$$G(z) = \frac{\sum_{n=0}^{M} b_n z^{-n}}{1 + \sum_{m=1}^{N} a_m z^{-m}} \qquad z = e^{j\omega T}$$

- The program will allow input of filter coefficients.
- The sampling frequency and the frequency range will be user specified.
- The calculated responses will be tabulated.
- A plot of the magnitude response will be made using standard asci characters.
- The program will allow output of data for graphing in an external program (e.g. MATLAB).

Design and Implementation

The formula contains complex variables which are not directly supported in c. The code to handle the complex numbers will need to be written directly. The complex numbers can be split into real and imaginary part and then these calculated separately. This can be done using DeMovires theorem. Two for loops can be used to calculate the summations independently. The top and bottom halves of the equation do not need to be combined using a complex conjugate method. The following formulas can be used:

$$z = \frac{a+bi}{c+di} \qquad |z| = \sqrt{\frac{a^2+b^2}{c^2+d^2}} \qquad \arg(z) = \tan^{-1}\frac{b}{a} - \tan^{-1}\frac{d}{c}$$

Th coefficients will need to be stored in an array for use by the for loops. The most flexible solution would be to read the coefficients from a formatted data file. This would mean the user would not have to type them in on the command line. A space separated two-column format will be used. E.g. with given coefficients:

Data.dat

```
1.0000 0.0040
-3.8082 -0.0004
6.2425 0.0043
-5.4277 0.0043
2.4913 -0.0004
-0.4821 0.0040
```

The user will then need to specify the file name of the input file at run time.

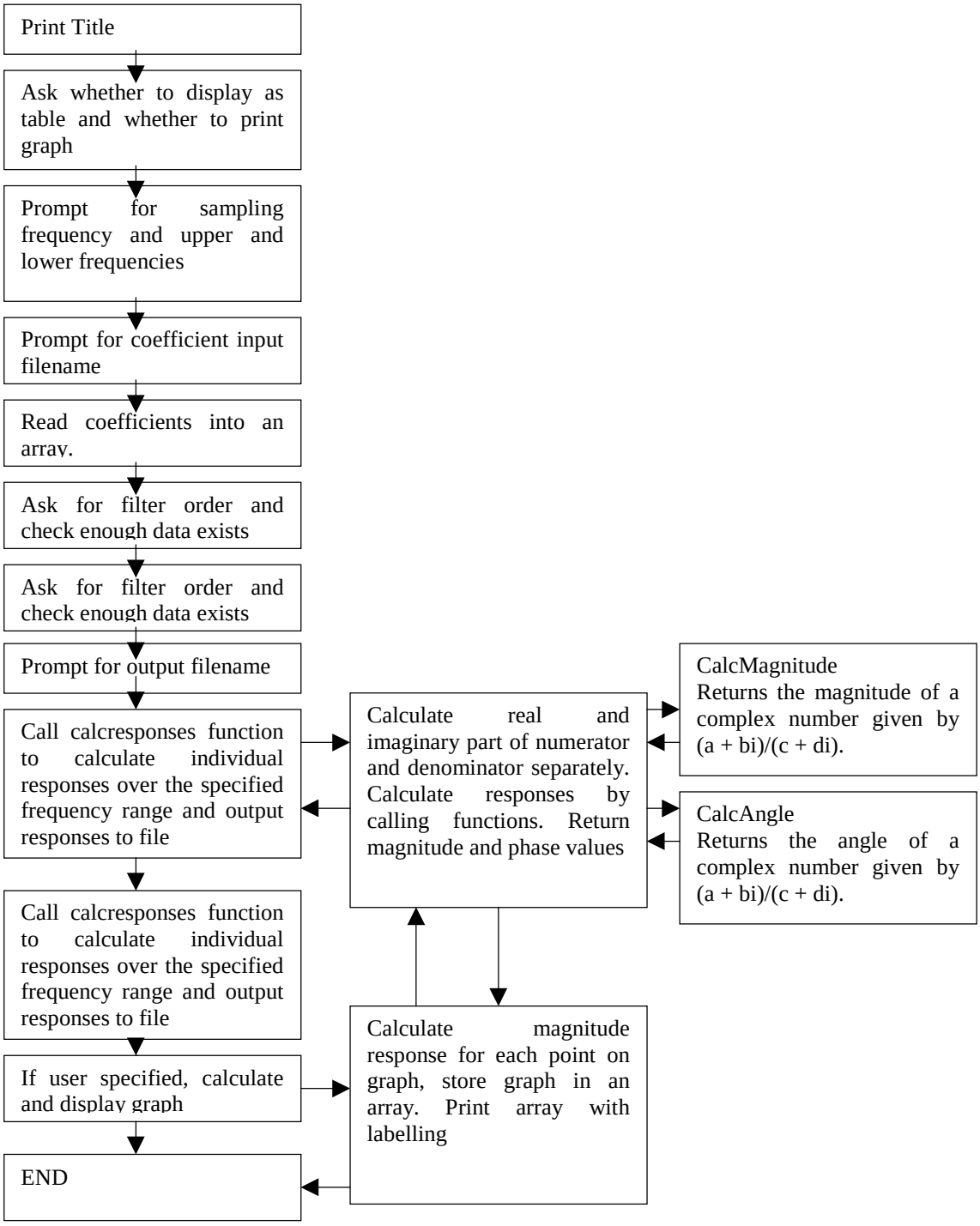
The program will need to calculate the values of the responses over a user specified frequency range.
A for loop will be used for this. The responses (magnitude and argument) can be calculated using a separate function. This will help with testing and make the program tidier. Since two variables need to be returned a structure will be used, the definition for this is shown below:

```
typedef struct { double Mag,Arg; } responses;
```

The attenuation response and the delay response can be calculated external to the function. The delay response can be approximated by taking (difference in phase)/(difference in omega) between each step.
The program will need to ask the user to input the sampling frequency, and the start, end and step of the frequency range.
The program will need to output the data generated to a file in a formatted table or as a file of data only (user specified). 'Data only' should be suitable for importing into a program such as Matlab for graphing etc.
The following will need to be included in the file: Magnitude response, Attenuation Response, Phase Response and Delay Response.

The asci graph should be displayed using a separate function since it may not be needed. A two dimensional character array can be used to store each character on the graph. The array can be populated with spaces and then a '*' character inserted at the correct height for each step. The function will need to calculate the magnitude responses at each step by calling the function to perform this. The array can then be printed out with relevant axes and labels.

Program Block Diagram:



Program Listing

```
/*
** Assignment2.cpp **

0013679

Assignment 2 - Digital Filter Response

Version 1.3b

Copyright 2001

*/


#include "stdlib.h"
#include "stdio.h"
#include "math.h"
#include "conio.h"
#define PI 3.1415927

typedef struct { double Mag,Arg; } responses;

// Function declarations

responses GetResponse(double omega,double T,int length);
double CalcMagnitude(double a,double b,double c,double d);
double CalcAngle(double a,double b,double c,double d);
double CalcAtten(double mag);
void DoGraph(double start,double end,double T,int length);
void char_line(int n,char c);

//Global Array for coefficients
double coef[50][2];

void main(void)
{

   // Variables
   double lower_bound,upper_bound,interval;  // Frequency range and interval
   int num_responses; // Stores number of frequency steps
   double sample_freq; // Sampling Frequency
   int N;              // Filter order
   double omega,T;     // omega , sampling interval
   double a, b;        // Used to read filter coefficients
   double freq;        // Frequency to calculate responses at
   double atten,delay; // Tempory variables for attenuation and delay
   int length = 0;     // Length of coefficient array (later filter order)
   responses response; // Tempory structure for reciving calculated responses
   double old_phase = 0; // Used to calculate delay response
   double old_omega = 0; // Used to calculate delay response
   char file_name[100];  // Used for user filename input
   typedef FILE* stream; // Defines stream as file pointer
   stream file_in,file_out; // File input/output streams
   char do_table,do_graph; // Question response variables

   // Prints Title
   printf("\nAssignment 2 - Digital Filter Response\n");
   printf("----------------------------------------\n\n");

   //Ask about table form
   printf("Write output in table form (y/n)?");
   do_table = _getch(); //Get keypress
   if(tolower(do_table) == 'y') {
      do_table = 1;
      printf("\nTable output selected\n");
   }
   else {
      printf("\nTable output not selected\n");
      do_table = 0;
   }

   //Ask whether to show graph
```

```
printf("Print Graph (y/n)?");
do_graph = _getch(); //Get keypress
if(tolower(do_graph) == 'y') {
   do_graph = 1;
   printf("\nGraph will be displayed\n");
}
else {
   printf("\nNo Graph will be displayed\n");
   do_graph = 0;
}

//Get sample frequency
printf("\nPlease enter sampling frequency (Hz):");
scanf("%lg", &sample_freq);
T = 1/sample_freq;

//Get upper frequency lower frequency and interval
printf("-Frequency Range-\nPlease enter lower frequency bound (Hz):");
scanf("%lg", &lower_bound);
printf("Please enter upper frequency bound (Hz):");
scanf("%lg", &upper_bound);
if (sample_freq/2<upper_bound) {
   printf("!Sampling frequency lower than 2*max frequency!\n");
}
printf("Please enter frequency interval (Hz)   :");
scanf("%lg", &interval);


num_responses = 1 + (int)((upper_bound - lower_bound)/interval);
printf("Number of points = %i\n",num_responses);

//Reads coefficients from two column space separated file
//Storing them in an global array.
printf("Please enter coefficient input filename:");
scanf("%s", &file_name);
file_in = fopen(file_name, "r");
if((feof(file_in)) | (file_in == NULL)) {
   printf("Error: File not found!\n");
   exit(1);
}
printf("Reading coefficients...");
for(;;) {
   fscanf(file_in, " %lg %lg", &a, &b);
   if(feof( file_in )) break;
   coef[length][0] = a;
   coef[length][1] = b;
   if(length==49) {
     printf("Maximum 50 coefficient pairs - Read 50");
     break;
   }
   length++;
}
fclose(file_in);
printf(" Done\n");

//Asks user to input filter order, checks enough data is present
printf("Please enter filter order (max %i):",length-1);
scanf("%i", &N);
if ( N>=length) {
   printf("Not enough data. Defaulting to order %i\n",length-1);
}
else {
   length = N + 1;
}

//Get output filename and open file for writing
printf("Please enter output filename:");
scanf("%s", &file_name);
 file_out = fopen(file_name, "w");
if(file_out == NULL) {
   printf("Error: Cannot open file for writing!\n");
   exit(1);
}

//Print headings if formatted output
if(do_table) {
   printf("Output in table form\n");
```

```
      fprintf(file_out,"-------------------------------------------------------------
---------\n");
      fprintf(file_out,"| Frequency   | Magnitude   | Attenuation | Phase       |
Delay (sec) |\n");
      fprintf(file_out,"|             | Response    | Response(dB)| Response    |
Response    |\n");
      fprintf(file_out,"-------------------------------------------------------------
---------\n");
   }


   //Calculate responses
   freq = lower_bound;
   printf("Calculating Responses...");
   while(freq<upper_bound+1){
      omega = 2 * PI * freq;
      response = GetResponse(omega,T,length);
      atten = CalcAtten(response.Mag);
      delay = (response.Arg-old_phase)/(omega-old_omega);
      if (do_table) {
      fprintf(file_out,"| %-11lf | %-11lf | %-11lf | %-11lf | %-11lf
|\n",freq,response.Mag,atten,response.Arg,delay);
      }
      else {

   fprintf(file_out,"%lg\t%lg\t%lg\t%lg\t%lg\n",freq,response.Mag,atten,response.Arg,d
elay);
      }
      old_omega = omega;
      old_phase = response.Arg;
      freq += interval;
   }
   if(do_table) {
      fprintf(file_out,"-------------------------------------------------------------
---------\n");
   }

   fclose(file_out);
   printf(" Done\n");

   //If user specified calculate and display graph
   if(do_graph) DoGraph(lower_bound,upper_bound,T,length);

   printf("\nNormal Program Termination\n");
}


responses GetResponse(double omega,double T,int length) {
   //Variables
   responses result; // Result structure to return
   double a, b;      // coefficient variables
   double real_numerator, img_numerator;
   double real_denominator, img_denominator;
    int m,n;          // Loop indexes
   //Initialise variables
   real_denominator = 1;
   img_denominator = 0;
   real_numerator = 0;
   img_numerator = 0;
   //Calculate summations for real and imaginary parts
   for(n=0;n<length;n++) {
      b = coef[n][1];
      real_numerator += b * cos(n * omega * T);
      img_numerator += b * sin(-n * omega * T);
   }
   for(m=1;m<length;m++) {
      a = coef[m][0];
      real_denominator += a * cos(m * omega * T);
      img_denominator += a * sin(-m * omega * T);
   }
   //Call functions to calculate Magnitude and angle
   result.Mag =
CalcMagnitude(real_numerator,img_numerator,real_denominator,img_denominator);
    result.Arg =
CalcAngle(real_numerator,img_numerator,real_denominator,img_denominator);
   return result;
}
```

```
double CalcMagnitude(double a,double b,double c,double d) {
    // z = (a + bi)/(c + di)
   // return |z|
   return sqrt((a*a+b*b)/(c*c+d*d));
}

double CalcAngle(double a,double b,double c,double d) {
   // z = (a + bi)/(c + di)
   // return arg(z)
   return (atan2(b,a)-atan2(d,c));
}

double CalcAtten(double a) {
   // return -20log10(a)
   return (-20 * log10(a));
}

void DoGraph(double start,double end,double T,int length) {
   //Prints asci graph of magnitude response
   int i,j;
   responses response;
   int bar_height;
   int GRAPH_WIDTH = 70;
   int GRAPH_HEIGHT = 20;
   double step = (end-start) / GRAPH_WIDTH; // set step
   double freq = start;
   double omega;
   char temp; // Used for 'press any key to continue'
   char graph[70][20]; // 2D character array for graph

   //Store space character in entire 2D array
   for (i=0;i<GRAPH_WIDTH;i++) {
      for (j=0;j<GRAPH_HEIGHT;j++) {
        graph[i][j] = 32;
      }
   }

   //Call GetResponse to calculate position of point for each bar
   for (i=0;i<GRAPH_WIDTH;i++) {
      omega = 2 * PI * freq;
      response = GetResponse(omega,T,length);
      bar_height = (int)(response.Mag*GRAPH_HEIGHT); //Calculate position (max=1
min=0)
      graph[i][bar_height] = 42; // stores '*' char in calculated position
      freq += step; // Calculates next step
   }
   printf("\nPress a key for graph...");
   temp = _getch();  //Waits for keypress
   //Display graph by printing array with axes and labels
   printf("\n\n");
   char label[20] = "     Magnitude     "; // Vertical axis label
   //Print title
   char_line(GRAPH_WIDTH/2-6,' ');
   printf("Magnitude Response\n  1|");
   //Print graph character array
   for (i=GRAPH_HEIGHT-1;i>=0;i--) {
      for (j=0;j<GRAPH_WIDTH;j++) {
        printf("%c",graph[j][i]);
      }
      printf("\n  %c|",label[GRAPH_HEIGHT-i]); // y-axes and labels
   }
   //Print x-axis and labels
   printf("\r  0+");
   char_line(GRAPH_WIDTH,'-');
   printf("\n   %-4g",start);
   char_line(GRAPH_WIDTH/2-3-7,' ');
   printf("Frequency (Hz)");
   char_line(GRAPH_WIDTH/2-4-7,' ');
   printf("   %4g",end);
   //Wait to preserve graph layout
   printf("\nPress a key to continue...");
   temp = _getch();
}

void char_line(int n,char c) {
   // Function for writing a line of a single character
```

```
   int i;
   for(i=0;i<n;i++) printf("%c",c);
}
```

Testing

Each function was tested by using dummy input data and printing results at each step. This was especially important for the DoGraph function which was tested in a separate program, without calculating the spot heights so the layout could be perfected. The CalcResponses function was also tested with single values instead of the for loop. The results obtained were compared to results calculated by hand or in Matlab. Two of the Matlab programs used are listed below. These were also modified to output data to the screen at different points in the calculation. The sample coefficients were used and all input was written directly into the programs:

```
%testfil.m
N = 512
b = [0.0040, -0.0004, 0.0043,  0.0043,-0.0004,  0.0040];
a = [1.0000, -3.8082, 6.2425, -5.4277, 2.4913, -0.4821];
Mag = zeros(N,1);
Atten = zeros(N,1);
Phase = zeros(N,1);
X = zeros(N,1);
%b = 0.0040;
%a = 1.00;
T = 1 / 1000;
maxomega = 2*pi*500;
step = maxomega/N;
for i=1:N,
   numerator = 0;
   denominator = 1;
   omega = i * step;
   X(i) = omega/(2*pi);
   for n=0:5,
      numerator = numerator + b(n+1)*exp(j*omega*T*-n);
   end
   for m=1:5,
     denominator = denominator + a(m+1)*exp(j*omega*T*-m);
   end

   G = numerator/denominator;
   Mag(i) = abs(G);
   if Mag(i)==0
      Atten(i) = 0;
   else
      Atten(i) = +1*20*log10(Mag(i));
   end
   Phase(i) = angle(G);
end
plot(X,Mag)
Title('Magnitude')
pause
plot(X,Atten)
Title('Attenuation')
pause
plot(X,Phase)
Title('Phase')
%fprintf(1,'Mag = %d Atten = %g Phase = %g\n',Mag,Atten,Phase)
```

```
%irr.m
b = [0.0040, -0.0004, 0.0043,  0.0043,-0.0004,  0.0040];
a = [1.0000, -3.8082, 6.2425, -5.4277, 2.4913, -0.4821];
 (b,a,512,1000);
Title('Elliptic Lowpass Filter'); grid;
```

The irr.m program was used to check that testfil.m and the c program were giving the correct answers. Freqz is an inbuilt Matlab function and can be used to calculate the response of a digital filter.

Analysis of Results

Data.dat

```
----------
1.0000 0.0040
-3.8082 -0.0004
6.2425 0.0043
-5.4277 0.0043
2.4913 -0.0004
-0.4821 0.0040
```

The program input parameters and output are shown below (test input used):

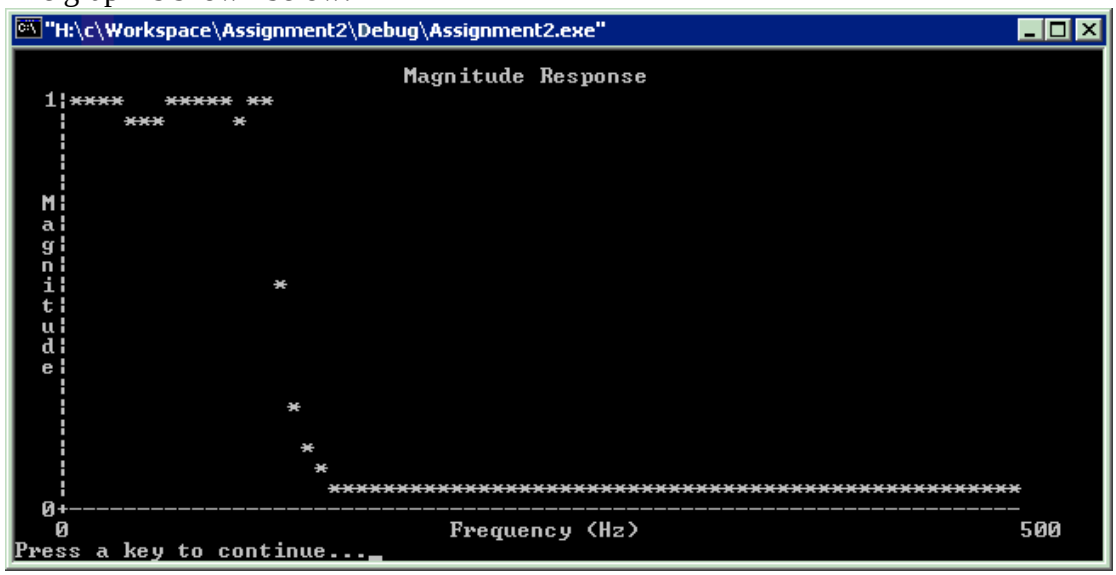
```
Assignment 2 - Digital Filter Response
--------------------------------------

Write output in table form (y/n)?
Table output not selected
Print Graph (y/n)?
Graph will be displayed

Please enter sampling frequency (Hz):1000
-Frequency Range-
Please enter lower frequency bound (Hz):0
Please enter upper frequency bound (Hz):500
Please enter frequency interval (Hz)   :1
Number of points = 501
Please enter coefficient input filename:data.dat
Reading coefficients... Done
Please enter filter order (max 5):5
Please enter output filename:out.txt
Calculating Responses... Done

Press a key for graph...
```

The graph is shown below:



The following program was used to import and graph the output in MATLAB. The unwrap function is used to unwrap the phase response, removing the jumps from ±infinity.

```
%graphdata.m
load out.txt;
Freq = out(:,1);
Mag = out(:,2);
Atten = out(:,3);
Phase = out(:,4);
Delay = out(:,5);

subplot(2,2,1), plot(Freq,Mag);
Title('Magnitude Response');
xlabel('Frequency Hz');
ylabel('Magnitude Response');
subplot(2,2,2), plot(Freq,Atten);
Title('Attenuation Response');
xlabel('Frequency Hz');
ylabel('Attenuation Response (dB)');
subplot(2,2,3), plot(Freq,Phase);
Title('Phase Response');
xlabel('Frequency Hz');
ylabel('Phase Response (rad)');
Phase = unwrap(Phase);
subplot(2,2,4), plot(Freq,Phase);
Title('Unwrapped Phase Response');
xlabel('Frequency Hz');
ylabel('Phase Response (rad)');

plot(Freq,Delay);
Title('Delay Response');
xlabel('Frequency Hz');
ylabel('Delay Response');
```

The graphs produced are shown on the next two pages. Also shown is the output from the irr.m program with the given coefficients.

The results shown agree with the results from freqz. The response clearly shows a low-pass filter.

The following program is used to generate a new set of coefficients for a different filter:

```
%irr2.m
%Set varibles
Wp = .1  % pass freq
Ws = .2  % stop freq
Rp = 1;
Rs = 70;
%calculate coefficients
[N,Wn] = ellipord(Wp,Ws,Rp,Rs)
[b,a] = ellip(N,Rp,Rs,Wn);
%Print graph
freqz(b,a,512,1000)
Title('Elliptic Lowpass Filter'); grid;
%output coefficients
FID = fopen('coeff.dat','wt');
for i=1:N+1,
   fprintf(FID,'%g %g\n',a(i),b(i));
end
fprintf(FID,'\n');
fclose(FID);
```

Coefficients generated (coeff.dat):

```
1 0.000561052
-4.58482 -0.000975781
8.53324 0.000608935
-8.05224 0.000608935
3.85033 -0.000975781
-0.746128 0.000561052
```

This is then inputted into the program as shown below (program output):

```
Assignment 2 - Digital Filter Response
--------------------------------------

Write output in table form (y/n)?
Table output not selected
Print Graph (y/n)?
Graph will be displayed

Please enter sampling frequency (Hz):1000
-Frequency Range-
Please enter lower frequency bound (Hz):0
Please enter uppe
r frequency bound (Hz):500
Please enter frequency interval (Hz)   :1
Number of points = 501
Please enter coefficient input filename:coeff.dat
Reading coefficients... Done
Please enter filter order (max 5):5
Please enter output filename:\out.txt
Calculating Responses... Done

Press a key for graph...
```

The output from running programs irr2.m and graphdata.m is shown on the next two pages:

Comparison of the graphs shows that the data generated by the c program is the same as generated by freqz. A number of other sets of coefficients were tested and the results also conformed.

A sample of a tabulated data is shown below with the input used to produce it:

```
Assignment 2 - Digital Filter Response
--------------------------------------

Write output in table form (y/n)?
Table output selected
Print Graph (y/n)?
No Graph will be displayed

Please enter sampling frequency (Hz):1000
-Frequency Range-
Please enter lower frequency bound (Hz):0
Please enter upper frequency bound (Hz):500
Please enter frequency interval (Hz)   :100
Number of points = 6
Please enter coefficient input filename:data.dat
Reading coefficients... Done
Please enter filter order (max 5):5
Please enter output filename:table.txt
Output in table form
Calculating Responses... Done

Normal Program Termination
```

Table.txt:

```
-------------------------------------------------------------------------
| Frequency   | Magnitude   | Attenuation | Phase       | Delay (sec) |
|             | Response    | Response(dB)| Response    | Response    |
-------------------------------------------------------------------------
| 0.000000    | 1.000000    | 0.000000    | 0.000000    | -1.#IND00   |
| 100.000000  | 0.956536    | 0.385974    | 1.463076    | 0.002329    |
| 200.000000  | 0.000774    | 62.221161   | 2.133927    | 0.001068    |
| 300.000000  | 0.000929    | 60.635071   | -1.290426   | -0.005450   |
| 400.000000  | 0.000758    | 62.407293   | -1.447449   | -0.000250   |
| 500.000000  | 0.000000    | 204.316122  | 1.570796    | 0.004804    |
-------------------------------------------------------------------------
```

The program can be used to obtain the responses of an Nth order recursive digital filter. The ability to output the data as a file for importing into Matlab for use with the program specified makes this a powerful and flexible tool.

Bibliography

Lecture Notes.
A. Fischer, D. Eggert, S. Ross, *Applied C: An introduction and more, McGraw-Hill*, 2001.
