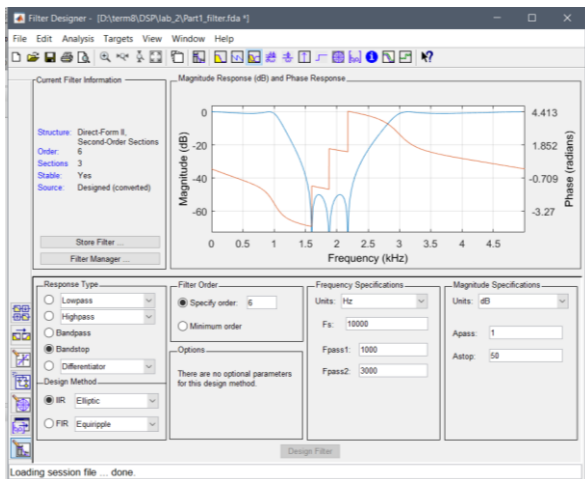


Experiment 2 – DSP Lab

Mohadeseh
Azari,
810196404

I. PART ONE

In this part we implement an IIR filter using fdatool in Matlab, the following is my IIR_Filter in Matlab:

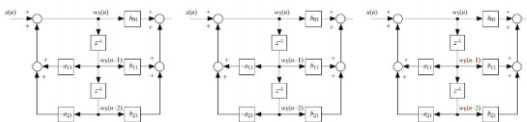


IIR_Filter

This IIR filter have the following properties:

- I. $F_s = 10000\text{Hz}$
- II. $F_{\text{pass1}} = 1000\text{Hz}$
- III. $F_{\text{pass2}} = 3000\text{Hz}$
- IV. $A_{\text{pass}} = 1\text{dB}$
- V. $A_{\text{stop}} = 50\text{dB}$

The aim is to implement this filter using 3 serialized second degree filters.



```
g Fortran wxSmith Tools Tools+ Plugins DowsBlocks Settings Help
X Part1_filter.c X
1 #include <stddef.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5
6 #define NUM_SECTIONS 3
7 float w[3][3];
8 float G[3][1]={0.5292,0.5292,0.8102};
9 float a[3][2]={{-1.4804,0.8386},{0.5241,0.7525},{-0.3142,-0.1773}};
10 float b[3][3]={{{1,-0.4059,1},{1,-1.0725,1},{1,-0.7639,1}}};
11
12 ////////////////////////////////////////////////// Part One ///////////////////////////////////
13 float Part_1(int section,float input)
```

First part of the code concerns with coefficients like a&b and G as a gain.

```
//////////////////////////////////////////////// Part One ///////////////////////////////////
float Part_1(int section,float input)
{
    float yn;
    w[section][2]=input-a[section][0]*w[section][0]-a[section][1]*w[section][1];
    yn=b[section][0]*w[section][2]+b[section][1]*w[section][0]+b[section][2]*w[section][1];
    w[section][1] = w[section][0];
    w[section][0] = w[section][2];
    return yn;
}

float IIR_Filter(float input)
{
    for(int j=0; j<NUM_SECTIONS; j++)
    {
        w[j][0] = 0;
        w[j][1] = 0;
        w[j][2] = 0;
    }

    float Out_Put;
    Out_Put = Part_1(0, input*G[0][1]);
    Out_Put = Part_1(1, Out_Put*G[1][1]);
    Out_Put = Part_1(2, Out_Put*G[2][1]);
}
```

The second part of the code is our IIR filter like the code in manual description with the gain part which has been added.

```

//////////////////////////////////// Part Two //////////////////////////////////////
#define M_PI 3.141592
float FT(float input)
{
    int n=sizeof(input);
    float Out_put;
    for(int k=0;k<n;k++)
    {
        float sumreal=0;
        float sumimag=0;
        for(int t=0;t<n;t++)
        {
            float angle = 2 * M_PI * t * k / n;
            sumreal += creal(input) * cos(angle) + cimag(input) * sin(angle);
            sumimag += -creal(input) * sin(angle) + cimag(input) * cos(angle);
            Out_put = sqrt(pow(sumreal,2)+pow(sumimag,2));
            return Out_put;
        }
    }
}

```

Part two takes the signal as an input and returns the DFT of the input signal as the discrete Fourier transform of the signal.

```

//////////////////////////////////// Part Three //////////////////////////////////////
float noise_generator(int fs, int t, float mu, float sigma) {
    float rand_gen() {
        // return a uniformly distributed random value
        return ( (float) (rand()) + 1. ) / ( (float) (RAND_MAX) + 1. );
    }
    float normalRandom() {
        // return a normally distributed random value
        float v1=rand_gen();
        float v2=rand_gen();
        return cos(2*3.14*v2)*sqrt(-2.*log(v1));
    }

    float x = normalRandom()*sigma+mu;
    return x;
}

//////////////////////////////////// Main Part //////////////////////////////////////

```

Unfortunately, I couldn't plot my output file. All the files are in the code folder.