KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Sayısal İşaret İşleme
Final Raporu

Fatih BAŞATEMUR
388496

2020-2021 BAHAR DÖNEMİ
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

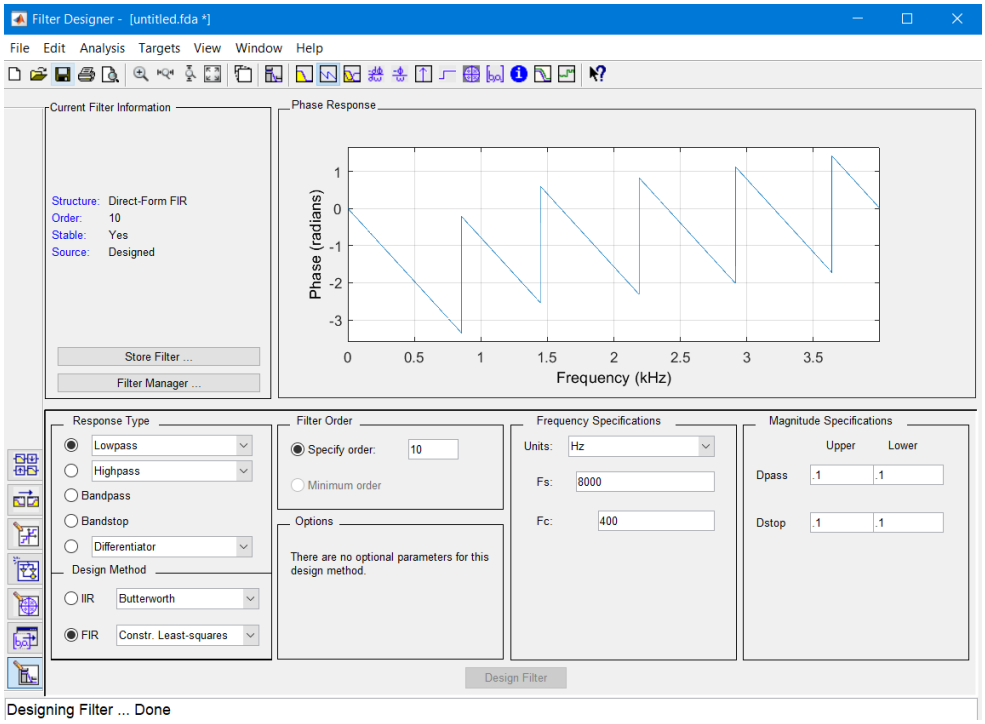
FIR Filtreleme

FIR (Finite Impulse Response) filtreler, birim dürtü tepkisi (Unit Impulse Response) sonlu uzunlukta olan filtrelerdir. N dereceli sayısal bir FIR filtre, M elemanlı x girdisi için N + M – 1 boyutunda çıktı üretir. Ayrıca FIR filtre derecesinin artması, x girdisi için uygulanan kesim frekansının keskinliğini arttırır.

H[n] FIR filtresinin birim tepkisi, çıktı tarafı algoritması (output side algorithm) ile önce filtrenin ters çevrilmesi ve daha sonra konvolüsyon işlemi uygulanması ile yapılır.

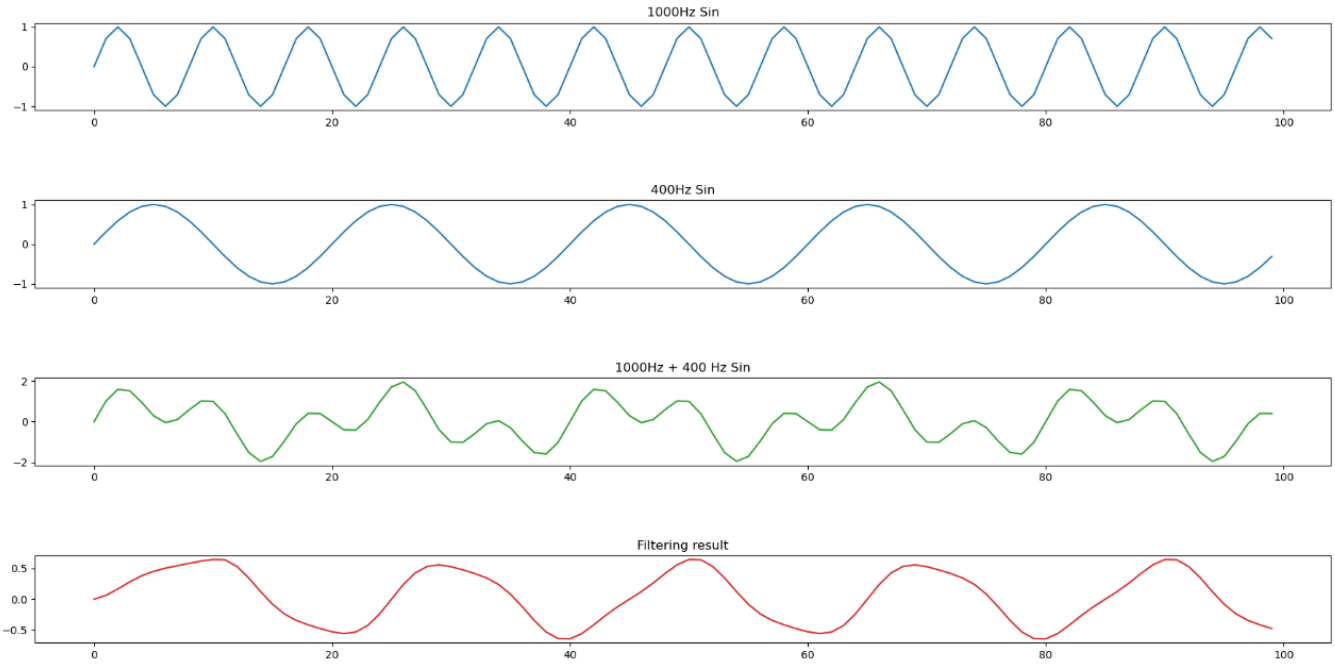
$$(x * h)[n] = \sum x[m].h[n-m]$$

Matlab Filter Designer kullanarak, 8KHz örnekleme frekansı ve 400Hz kesim frekansı için Lowpass filtre katsayıları üretilir.



8KHz örnekleme frekansı ve 400Hz kesim frekansı için Low-pass filtre katsayıları aşağıdadır:

0.061072119396879336050698583449047873728
0.068614601583902409820403534013166790828
0.079336759046606447620675339749141130596
0.092251949094245874749553593119344444122
0.103197321704239342032849435781827196479
0.107515708458737632713386744853778509423
0.103197321704239342032849435781827196479
0.092251949094245874749553593119344444122
0.079336759046606447620675339749141130596
0.068614601583902409820403534013166790828
0.061072119396879336050698583449047873728



Yukarıdaki filtreler 1000Hz ve 400Hz sinüs dalgalarının toplamından oluşan örnek ses dosyası için uygulandığında sonuç değerlerden oluşan grafik şekildeki gibidir.

Çıktı tarafı algoritması, çıktı sinyalindeki bir noktanın, giriş sinyali tarafından nasıl etkilendiğini inceler. Bu işlem filtre değerlerinin, giriş değerleri üzerinde ters çevrilerek katlama işleminin yapılması ile mümkündür.

```
// The Output Side Algorithm
float* Conv(float* input, int inputN, float* filter, int filterN, int& resultN)
{
    resultN = inputN + filterN - 1;
    float* result = new float[resultN]();
    int startk, endk;

    for (int i = 0; i < resultN; i++)
    {
        startk = i >= inputN ? i - inputN + 1 : 0;
        endk = i < filterN ? i : filterN - 1;

        for (int k = startk; k <= endk; k++)
            result[i] += input[i - k] * filter[k];
    }
    return result;
}
```

Parametre olarak verilen filtre değerlerinin, giriş wave dosyası üzerine uygulanması.

```
int main(int argc, char* argv[]) {

    if (argc == 4) {

        WavHeader* header = new WavHeader;
        size_t numSamples;

        float* samples = WavRead(argv[1], header, &numSamples);

        // gelen ses sinyali, 40000 orneklemeye sahip, fs=8KHz 5s
        cout << "Bit rate:        " << header->bps << endl;
        cout << "Sample rate:     " << header->sampleRate << endl;
        cout << "No. of channels: " << header->numChannels << endl;
        cout << "Number of samples: " << numSamples << endl << endl;


        // 400Hz kesim frekansi icin katsayilari al
        int filterN = atoi(argv[2]);
        float* filter = new float[filterN];

        ifstream file(argv[3]);
        string coeff;
        for (size_t i = 0; i < filterN; i++) {
            file >> coeff;
            filter[i] = stof(coeff);
        }
        file.close();

        // 1KHz ve 400Hz den olusan inputa, 400Hz lik kesim frekansli LowPass uygulayalim
        int resultN;
        float* result = Conv(samples, numSamples, filter, filterN, resultN);

        cout << "  -input-  " << "\t" << "  -output-  " << endl;
        for(int i = 0; i< 20; i++)
            cout<< "[" << i << "]: " << samples[i] << "\t -> " << result[i] << endl;

        WavSave("result.wav", header, resultN, result);
    }
    else
        printf("You must enter 4 params \n");

    return 0;
}
```

1000Hz ve 400Hz sinüs sinyallerinin toplamından oluşan giriş ses dosyası, 8000 örneklem frekansına ve 400Hz kesim frekansı özelliklerini sahip filtre katsayıları ile katlandığında elde edilen sonuç değerleri aşağıdadır.

```
  -input-         -output-
[0]: 0    -> 0
[1]: 0.999969     -> 0.0610703
[2]: 0.999969     -> 0.129683
[3]: 0.999969     -> 0.209017
[4]: 0.951019     -> 0.298277
[5]: 0.293701     -> 0.357968
[6]: -0.0477295   -> 0.395644
[7]: 0.102722     -> 0.427934
[8]: 0.587738     -> 0.467349
[9]: 0.999969     -> 0.51248
[10]: 0.998779    -> 0.550706
[11]: 0.397217    -> 0.558873
[12]: -0.587738   -> 0.461849
[13]: -0.999969   -> 0.344464
[14]: -0.999969   -> 0.215491
[15]: -0.999969   -> 0.0653649
[16]: -0.951019   -> -0.0701397
[17]: -0.102722   -> -0.15043
[18]: 0.411041    -> -0.200614
[19]: 0.397217    -> -0.250426
```