



**General Sir John Kotelawala Defense University**  
**Electrical, Electronic & Telecommunication Engineering**  
**ET3142 – Digital Signal Processing**  
**Practical I - Semester V**

U Reg#

Intake#

Marks

**The Discrete Fourier Transform(DFT)**

**1) Theoretical Derivation DFT/DTFT**

Consider the given plot in Figure 1 where there are five pulses ON and five pulses OFF.

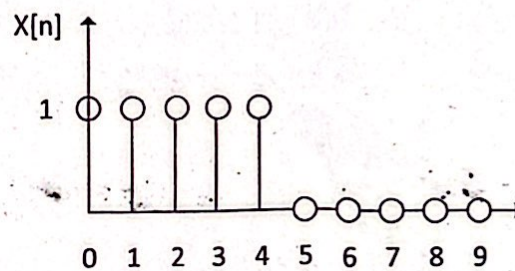


Figure 1

I. Find the DTFT of the plot given in Figure 1.

[2 marks]

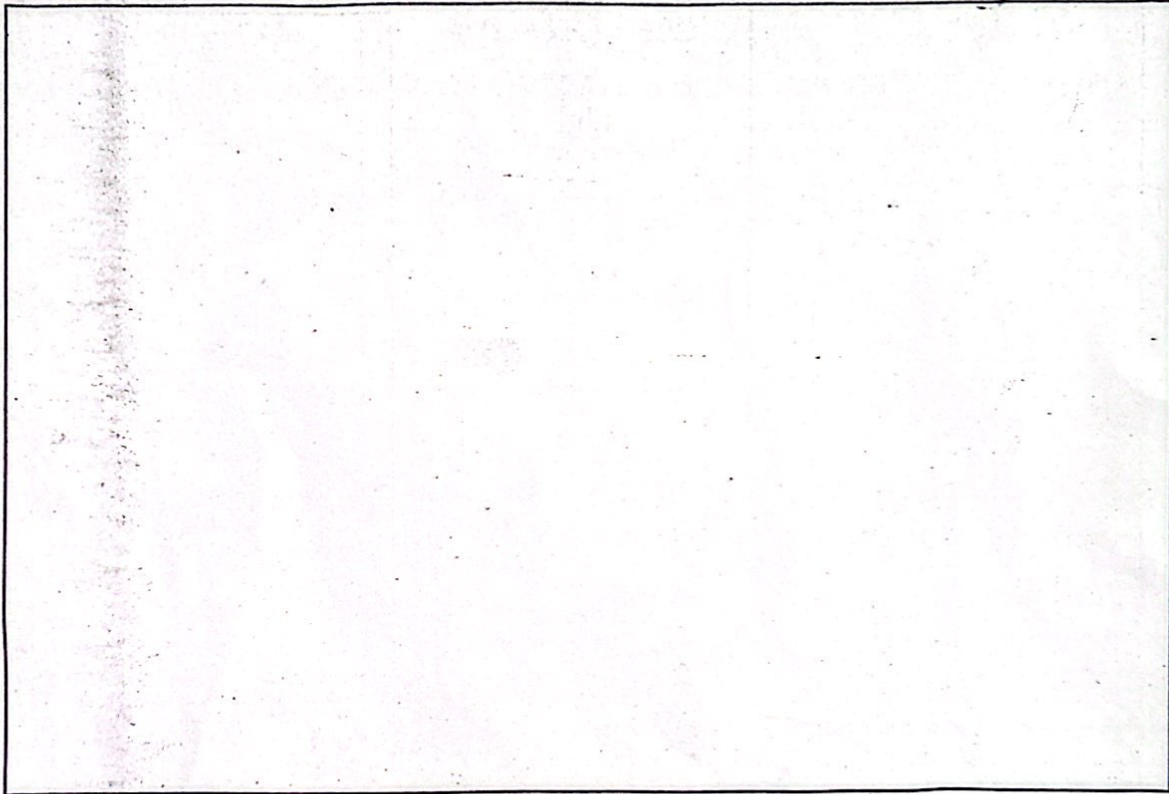
1



**General Sir John Kotelawala Defense University**  
**Electrical, Electronic & Telecommunication Engineering**  
**ET3142 – Digital Signal Processing**  
**Practical I - Semester V**

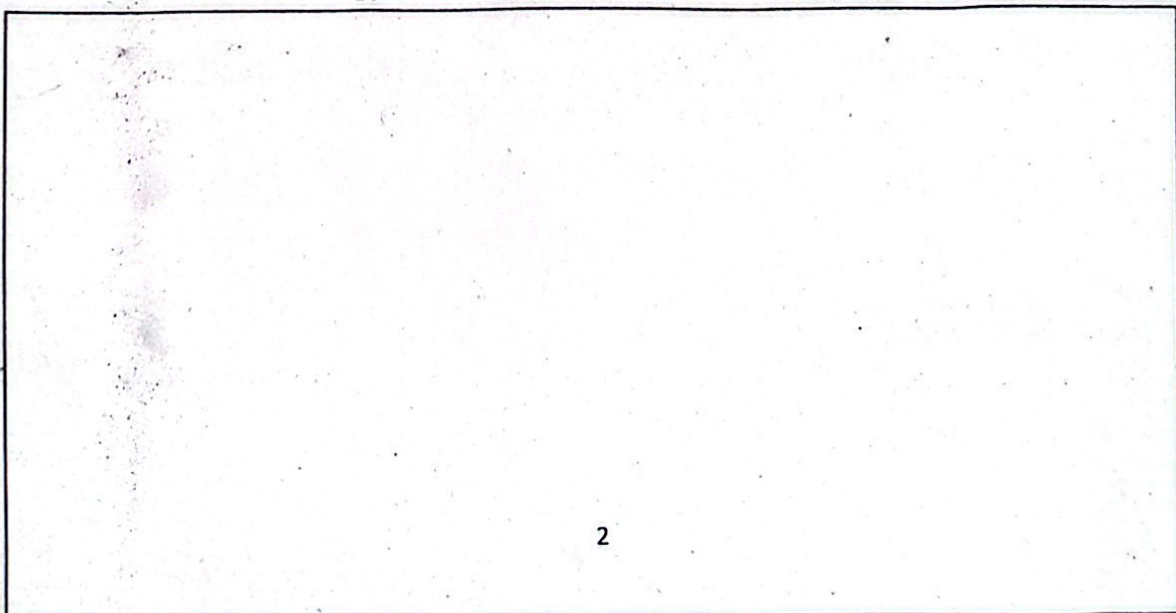
II. Find the DFT of the plot given in Figure 1.

[2 marks]



III. By taking  $\omega$  as  $\omega = \frac{2\pi k}{10}$  derive the answer in Part I

[2 marks]





**General Sir John Kotelawala Defense University**

**Electrical, Electronic & Telecommunication Engineering**

**ET3142 – Digital Signal Processing**

**Practical I - Semester V**

IV. Write your observation by comparing the values in Part I, II and III in Section 1)

[4 marks]

**2) Simulation Using Python**

**#Programme I - Get a resultant DFTF plot for Figure I**

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import math
```

```
from math import e
```

```
w = np.linspace(0, 2*math.pi, 200) #omega
```

```
sine_ratio_dtft = np.sin((5*w)/2)/np.sin(w/2)
```

```
dtft = e ** (2j*w) * sine_ratio_dtft
```



**General Sir John Kotelawala Defense University**  
**Electrical, Electronic & Telecommunication Engineering**  
**ET3142 – Digital Signal Processing**  
**Practical I - Semester V**



```
plt.figure()  
plt.xlabel("<--Amplitude (DTFT)-->")  
plt.ylabel("<--Frequency-->")  
plt.plot(w,abs(dtft))  
plt.grid()  
plt.show()
```

I. Get the resultant DFT Plot on top of DTFT plot using the given Programme I.

Hint : use  $w=(2*\text{math.pi}*k)/10$  for  $k \in \{0,1,2,\dots,9\}$

[5 marks]





**General Sir John Kotelawala Defense University**

**Electrical, Electronic & Telecommunication Engineering**

**ET3142 – Digital Signal Processing**

**Practical I - Semester V**

- II. Write your observation for Part I in Section 2) by considering the theoretical values that you have obtained in Section 1) [5 marks]



**GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY**  
Faculty of Engineering  
Department of Electrical, Electronics and Telecommunication Engineering

**ET5132 – DIGITAL SIGNAL PROCESSING**

**Practical 02 – Design of FIR digital filters using Kaiser Window**

**OBJECTIVE** - To provide experience in the design of FIR digital filters for prescribed specifications using the windowing method in conjunction with the Kaiser window.

The specifications of the digital filter are different from student to student, and are derived using the index numbers of the students. Let us denote the last three digits of the index number of a student as ABC, where A, B and C are integers in the range 0 to 9.

Parameter	Value
Maximum passband ripple, $\tilde{A}_p$	0.2A dB
Minimum stopband attenuation, $\tilde{A}_s$	4B dB
Lower passband edge, $\omega_{p1}$	C00+500 rad/s
Upper passband edge, $\omega_{p2}$	C00+1000 rad/s
Lower stopband edge, $\omega_{s1}$	C00+200 rad/s
Upper stopband edge, $\omega_{s2}$	C00+1200 rad/s
Sampling frequency, $\omega_s$	2(C00+1600) rad/s

Table 1: Filter specifications.

- Using the windowing method in conjunction with the Kaiser window, design an FIR bandpass digital filter that will satisfy the specifications given in Table 1.
  - Plot the impulse response.
  - Plot the magnitude response of the digital filter obtained for the frequency range 0 to  $\omega_s/2$  rad/s.
  - Plot the magnitude response for frequencies in the passband.
- Write a report describing your results. Also, include your MATLAB programs in an appendix. Submit a soft copy of the report.

Useful MATLAB functions:

- **plot:** plots a function
- **kaiserord:** computes the order of an FIR filter and the independent parameter to specify a Kaiser window
- **kaiser:** computes a Kaiser window
- **fir1:** calculates the coefficients of FIR filters designed using the windowing method
- **impz:** computes the impulse response of a digital filter
- **freqs:** computes the frequency, magnitude, and phase response of an analog filter
- **freqz:** computes the frequency, magnitude, and phase response of a digital filter
- **unwrap:** unwraps the phase response of a digital filter
- **phasedelay:** computes the phase delay of a digital filter
- **grpdelay:** computes the group delay of a digital filter
- **fft:** calculates the DFT
- **fftshift:** shifts zero-frequency component to the center of the spectrum
- **fvtool:** this is a very useful and sophisticated tool, which opens a GUI and plots impulse, step, amplitude, and phase responses, delay characteristics, and can also plot many other quantities pertaining to a digital filter.



**GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY**  
Faculty of Engineering  
Department of Electrical, Electronics and Telecommunication Engineering

**ET5132 – DIGITAL SIGNAL PROCESSING**

**Practical 02 – Design of FIR and IIR digital filters**

**Designing Low Pass FIR Filters**

This example shows how to design lowpass FIR filters. FIR filters are widely used due to the powerful design algorithms that exist for them, their inherent stability when implemented in non-recursive form, the ease with which one can attain linear phase, their simple extensibility to multirate cases, and the ample hardware support that exists for them among other reasons. This example showcases functionality in the DSP System Toolbox™ for the design of lowpass FIR filters with a variety of characteristics.

**Lowpass Filter Design in MATLAB**

Design a lowpass FIR filter for data sampled at 48kHz and the passband-edge frequency is 8kHz. The passband ripple is 0.01dB and the stopband attenuation is 80 dB. Constrain the filter order to 120.

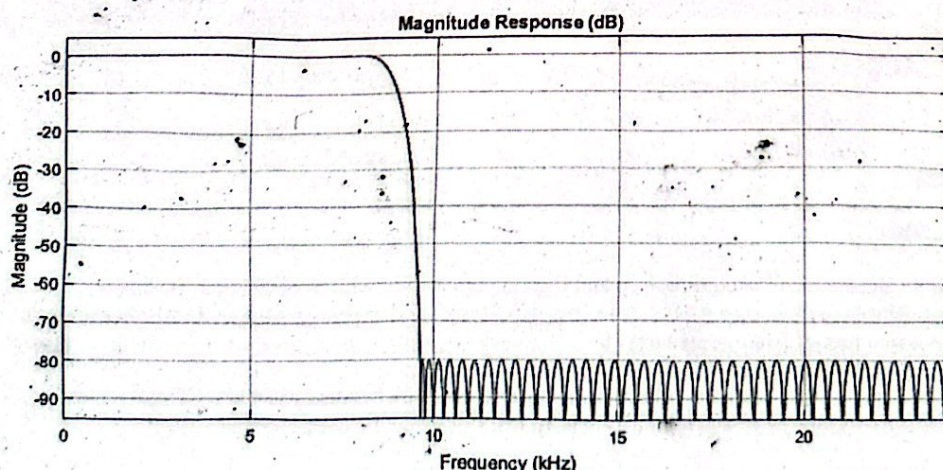
```
N=120;  
Fs=48e3;  
Fp=8e3;  
Ap=0.01;  
Ast =80;
```

Obtain the maximum deviation for the passband and stopband ripples in linear units.

```
Rp=(10^(Ap/20)-1)/(10^(Ap/20)+1);  
Rst=10^(-Ast/20);
```

Design the filter using and view the magnitude frequency response.

```
NUM= firceqrip(N,Fp/(Fs/2),[Rp Rst], 'passedge');  
fvtool(NUM,'Fs',Fs)
```



The resulting stopband-edge frequency is about 9.64 kHz.

## Minimum-Order Designs

Another design function for optimal equiripple filters is `firgr`. `firgr` can design a filter that meets passband/stopband ripple constraints as well as a specified transition width with the smallest possible filter order. For example if the stopband-edge frequency is specified as 10 kHz the resulting filter has an order of 100 rather than the 120th-order filter designed with `fireqrip`. The smaller filter order results from the larger transition band.

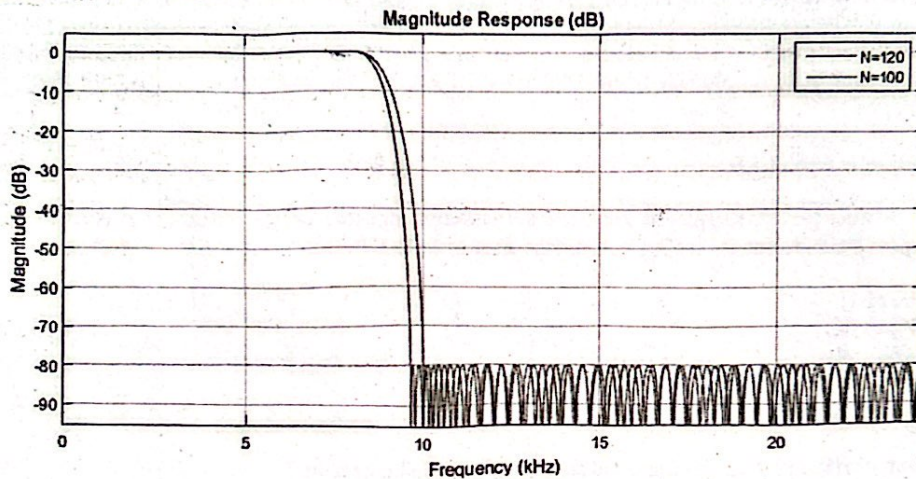
Specify the stopband-edge frequency of 10 kHz. Obtain a minimum-order FIR filter with a passband ripple of 0.01 dB and 80 dB of stopband attenuation.

```
Fst=10e3;
NumMin= firgr('minorder',[0 Fp/(Fs/2) Fst/(Fs/2) 1],[ 1 1 0 0],[Rp,Rst])
```

Plot the magnitude frequency responses for the minimum-order FIR filter obtained with `firgr` and the 120th-order filter designed with `fireqrip`. The minimum-order design results in a filter with order 100.

The transition region of the 120th-order filter is, as expected, narrower than that of the filter with order 100.

```
hvft = fvtool(NUM,1,NumMin,1,'Fs',Fs);
legend(hvft,'N=120','N=100')
```



## Filtering Data

To apply the filter to data, you can use the `filter` command or you can use `dsp.FIRFilter`. `dsp.FIRFilter` has the advantage of managing the state when executed in a loop.

```
LP_FIR = dsp.FIRFilter('Numerator',NUM);
SA = dsp.SpectrumAnalyzer('SampleRate',Fs,'SpectralAverages',5);
tic
while toc<10
    x = randn(256,1);
    y=LP_FIR(x);
    step(SA,y);
end
```

## Using `dsp.LowpassFilter`

`dsp.LowpassFilter` is an alternative to using `fireqrip` and `firgr` in conjunction with `dsp.FIRFilter`. Basically, `dsp.LowpassFilter` condenses the two step process into one. `dsp.LowpassFilter` provides the same advantages that `dsp.FIRFilter` provides in terms of fixed-point support.

Design a lowpass FIR filter for data sampled at 48 kHz. The passband-edge frequency is 8 kHz. The passband ripple is 0.01 dB and the stopband attenuation is 80 dB. Constrain the filter order to 120.

Create a `dsp.FIRFilter` based on your specifications.

```
LP_FIR_minOrd = dsp.Lowpassfilter('SampleRate',Fs,...
    'DesignForMinimumOrder',true,'PassbandFrequency',Fp,...
    'StopbandFrequency',Fst,'PassbandRipple',Ap,'StopbandAttenuation',Ast);
```

The coefficients in LP\_FIR are identical to the coefficients in NUM.

```
NUM_LP = tf(LP_FIR);
```

You can use LP\_FIR to filter data directly, as shown in the preceding example. You can also analyze the filter using FVTool or measure the response using measure.

```
fvtool(LP_FIR, 'Fs', Fs);  
measure(LP_FIR)
```

### Minimum-Order Designs with dsp.LowpassFilter

You can use dsp.LowpassFilter to design minimum-order filters and use measure to verify that the design meets the prescribed specifications. The order of the filter is again 100.

```
LP_FIR_minOrd = dsp.Lowpassfilter('SampleRate', Fs,...  
    'DesignForMinimumOrder', true, 'PassbandFrequency', Fp,...  
    'StopbandFrequency', Fst, 'PassbandRipple', Ap, 'StopbandAttenuation', Ast);  
measure(LP_FIR_minOrd)  
Nlp = order(LP_FIR_minOrd)
```

### Designing IIR Filters

Elliptic filters are the IIR counterpart to optimal equiripple FIR filters. Accordingly, you can use the same specifications to design elliptic filters. The filter order you obtain for an IIR filter is much smaller than the order of the corresponding FIR filter.

Design an elliptic filter with the same sampling frequency, cutoff frequency, passband-ripple constraint, and stopband attenuation as the 120th-order FIR filter. Reduce the filter order for the elliptic filter to 10.

```
N=10;  
LP_IIR = dsp.LowpassFilter('SampleRate', Fs, 'FilterType', 'IIR', ...  
    'DesignForMinimumOrder', false, 'FilterOrder', N, ...  
    'PassbandFrequency', Fp, 'PassbandRipple', Ap, 'StopbandAttenuation', Ast);
```

Compare the FIR and IIR designs. Compute the cost of two implementations.

```
hfvf = fvtool(LP_FIR, LP_IIR, 'Fs', Fs);  
legend(hfvf, 'FIR Equiripple, N = 120', 'IIR Elliptic, N = 10');  
cost_IR = cost(LP_FIR)  
cost_IIR = cost(LP_IIR)
```

### Running the IIR Filters

The IIR filter is designed as a biquad filter. To apply the filter to data, use the same commands as in the FIR case.

Filter 10 seconds of white Gaussian noise with zero mean and unit standard deviation in frames of 256 samples with the 10th-order IIR lowpass filter. View the result on a spectrum analyzer.

```
SA = dsp.SpectrumAnalyzer('SampleRate', Fs, 'SpectralAverages', 5);  
tic  
while toc < 10  
    x = randn(256, 1);  
    y = LP_IIR(x);  
    SA(y);  
end
```