4. Sampling theorem verification

AIM

To verify and plot the sampling theorem in MATLAB

THEORY

The sampling theorem can be defined as the conversion of an analog signal into a discrete form by taking the sampling frequency as twice the input analog signal frequency. Input signal frequency denoted by Fm and sampling signal frequency denoted by Fs.

The output sample signal is represented by the samples. These samples are maintained with a gap, these gaps are termed as sample period or sampling interval (Ts). And the reciprocal of the sampling period is known as “sampling frequency” or “sampling rate”. The number of samples is represented in the sampled signal is indicated by the sampling rate.

Sampling frequency **Fs=1/Ts**

Sampling theorem states that “continues form of a time-variant signal can be represented in the discrete form of a signal with help of samples and the sampled (discrete) signal can be recovered to original form when the sampling signal frequency Fs having the greater frequency value than or equal to the input signal frequency Fm.

If the sampling frequency (Fs) equals twice the input signal frequency (Fm), then such a condition is called the Nyquist Criteria for sampling. When sampling frequency equals twice the input signal frequency is known as “Nyquist rate”.

Fs=2Fm

If the sampling frequency (Fs) is less than twice the input signal frequency, such criteria called an Aliasing effect.

Fs<2Fm

So, there are three conditions that are possible from the sampling frequency criteria. They are sampling, Nyquist and aliasing states.

Nyquist sampling theorem states that the sampling signal frequency should be double the input signal’s highest frequency component to get distortion less output signal. As per the scientist’s name, Harry Nyquist this is named as Nyquist sampling theorem.

Fs=2Fm

PROGRAM

```
clc;
clear all;
t=0:00.001:1;
fm=input('Enter the modulating signal frequency :');
x=sin(2*pi*fm*t);
```

```
fs1=input('Enter the sampling frequency < modulating signal
:');
fs2=input('Enter the sampling frequency = modulating signal
:');
fs3=input('Enter the sampling frequency > modulating signal
:');
% sampling at fs < 2fm
n=0:(1/fs1):1;
x1=sin(2*pi*fm*n);

% sampling at fs = 2fm
n=0:(1/fs2):1;
x2=sin(2*pi*fm*n);

% sampling at fs > 2fm
n=0:(1/fs3):1;
x3=sin(2*pi*fm*n);

% Plotting signals
subplot(2,2,1);
    plot(t,x);
xlabel('time');
ylabel('amplitude');
title('MESSAGE SIGNAL');
subplot(2,2,2);
    stem(n,x1);
xlabel('time');
ylabel('amplitude');
title(' fs < 2fm ');
subplot(2,2,3);
    stem(n,x2);
xlabel('time');
ylabel('amplitude');
title(' fs = 2fm ');
subplot(2,2,4);
    stem(n,x3);
xlabel('time');
ylabel('amplitude');
title(' fs > 2fm ');
```

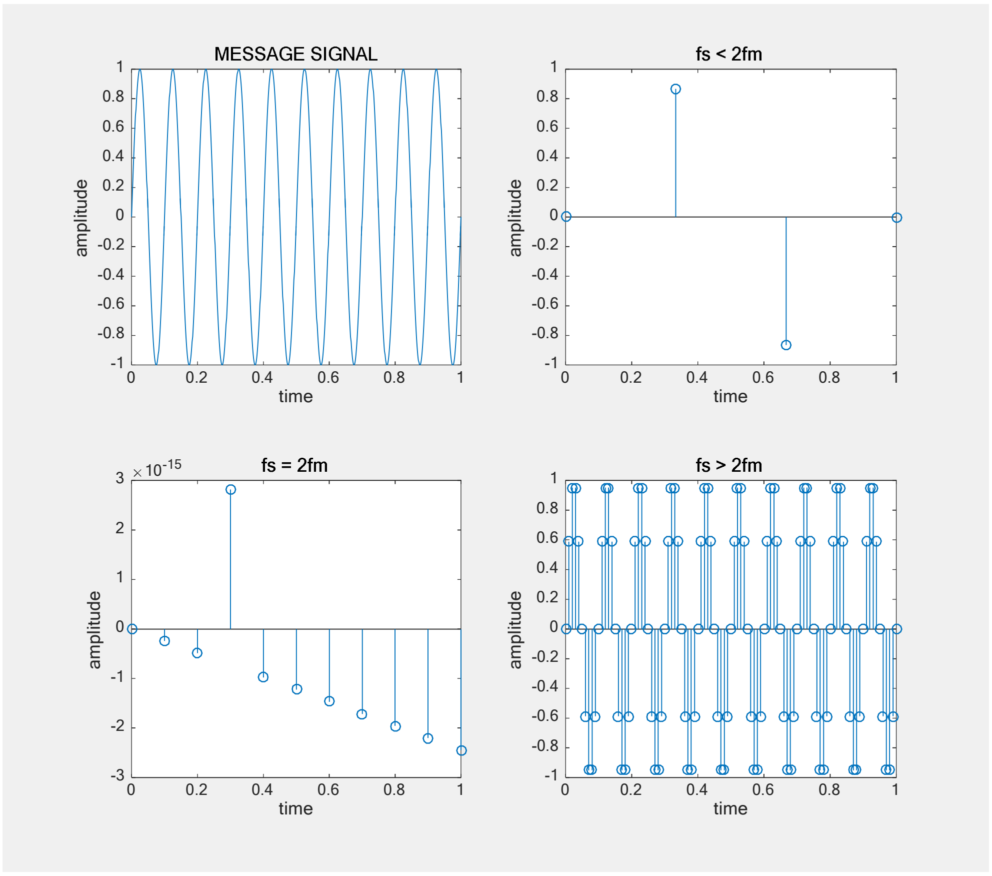
OUTPUT

```
Enter the modulating signal frequency :10

Enter the sampling frequency < modulating signal :3

Enter the sampling frequency = modulating signal :10

Enter the sampling frequency > modulating signal :100
```



Result

5. Verification of the Properties of DFT

AIM

Generate and appreciate a DFT matrix.

1. Write a function that returns the N point DFT matrix $\mathbf{V}_N$ for a given N.
2. Plot its real and imaginary parts of $\mathbf{V}_N$ as images using *imshow* commands for $N = 16$, $N = 64$ and $N = 1024$
3. Compute the DFTs of 16-point, 64-point and 1024-point random sequences using the above matrices.
4. Use some iterations to plot the times of computation against γ. Plot and understand this curve. Plot the times of computation for the *fft* function over this curve and appreciate the computational saving with FFT.

• Circular Convolution.

• Parseval's Theorem

1. For the complex random sequences $x_1[n]$ and $x_2[n]$,

PROGRAM

% Part 1: (For N = 16)

```
clear all
close all
clc
%Part 1 for N=16;
N=16;
x=randi(10,N,1)+i.*randi(10,N,1);
%To generate the twiddle factor
tic %Timer is ON to compute the overall execution time
W_N=exp(-2*1j*pi/N);
%To generate the twiddle factor matrix
for i=1:N
    for j=1:N
        V_N(i,j)=W_N.^((i-1)*(j-1));
    end
end
time(1)=toc;
%2. To pot the real and imaginary parts of Vn
% V_real=real(V_N);
% V_imaginary=imag(V_N);
subplot(2,1,1)
imshow(real(V_N));
title('Real parts of twiddle factor matrix as image')
subplot(2,1,2)
imshow(imag(V_N));
```

```
title('Imaginary parts of twiddle factor matrix as image')
tic
X_direct1=fft(x,N)
time_fft(1)=toc;
```

% Part 2: (For N=64)

```
N=64;
x=randi(10,N,1)+i.*randi(10,N,1);
%To generate the twiddle factor
tic %Timer is ON to compute the overall execution time
W_N=exp(-2*1j*pi/N);
%To generate the twiddle factor matrix
for i=1:N
    for j=1:N
    V_N(i,j)=W_N.^((i-1)*(j-1));
    end
end
time(2)=toc;
%2. To pot the real and imaginary parts of Vn
figure
subplot(2,1,1)
imshow(real(V_N));
title('Real parts of twiddle factor matrix as image')
subplot(2,1,2)
imshow(imag(V_N));
title('Imaginary parts of twiddle factor matrix as image')
tic
X_direct2=fft(x,N);
time_fft(2)=toc;
```

% Part 3: (For N=1024)

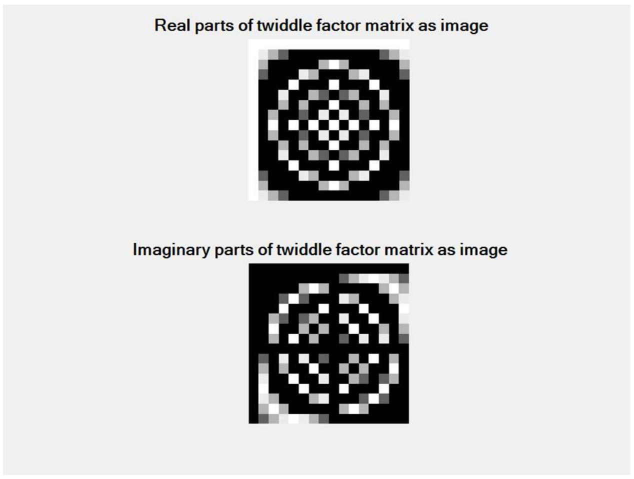
```
N=1024;
x=randi(10,N,1)+i.*randi(10,N,1);
%To generate the twiddle factor
tic %Timer is ON to compute the overall execution time
W_N=exp(-2*1j*pi/N);
%To generate the twiddle factor matrix
for i=1:N
    for j=1:N
    V_N(i,j)=W_N.^((i-1)*(j-1));
    end
end
X=V_N*x;
time(3)=toc;
%2. To pot the real and imaginary parts of Vn
figure
subplot(2,1,1)
imshow(real(V_N));
```

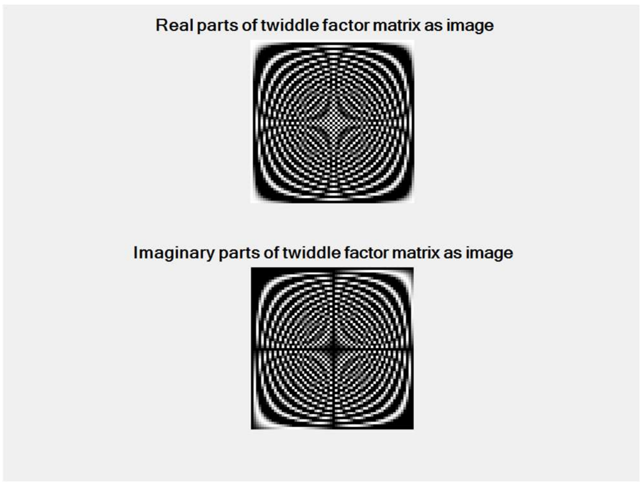
```
title('Real parts of twiddle factor matrix as image')
subplot(2,1,2)
imshow(imag(V_N));
title('Imaginary parts of twiddle factor matrix as image')
tic
X_direct3=fft(x,N);
time_fft(3)=toc;
```

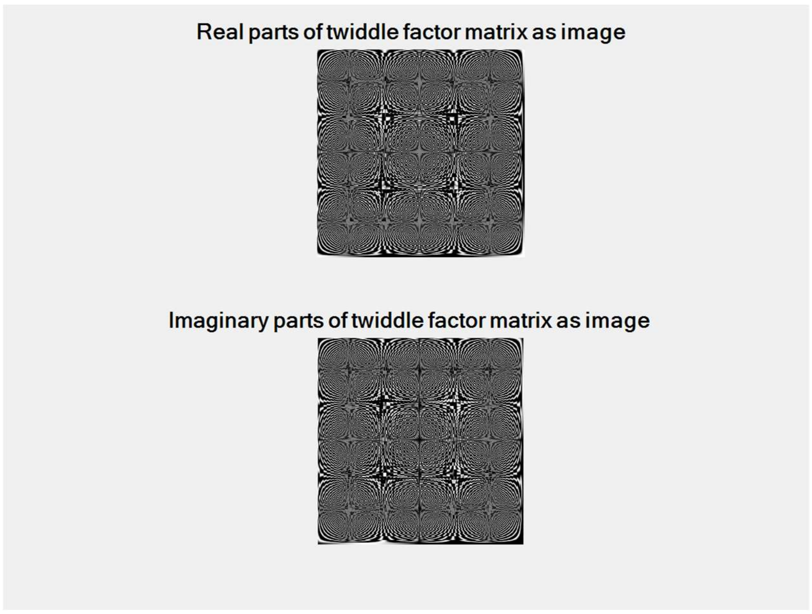
%Part 4: (Plotting time performance curve)

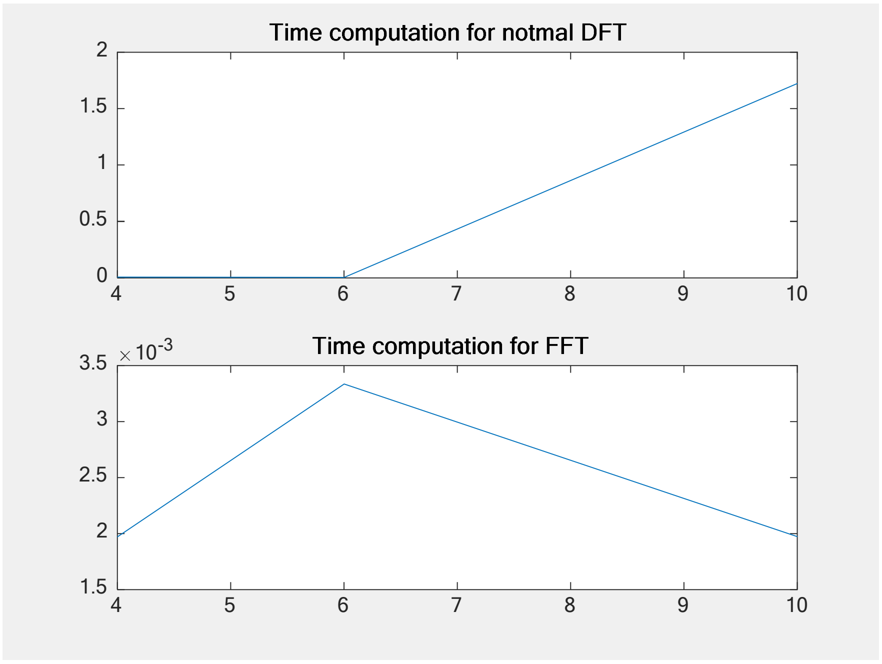
```
figure
gamma=[4,6,10];
subplot(2,1,1)
plot(gamma,time)
title('Time computation for notmal DFT')
subplot(2,1,2)
plot(gamma,time_fft)
title('Time computation for FFT')
```

Output


Real parts of twiddle factor matrix as image
Imaginary parts of twiddle factor matrix as image


Real parts of twiddle factor matrix as image
Imaginary parts of twiddle factor matrix as image





Circular convolution

```
clc
clear all
x1=input('Enter the first sequence x1(n) : ');
x2=input('Enter the second sequence x2(n) : ');
N1=length(x1);
N2=length(x2);
N=max(N1,N2);
disp('circular convuoultion of the result is ')
c=cconv(x1,x2,N)
% Plotting the sequences
subplot(3,1,1);
```

```
stem(x1);
title('first sequence x1(n)');
subplot(3,1,2);
stem(x2);
title('Second sequence x2(n)');
subplot(3,1,3);
stem(c);
title('Circular convolution result');
```

Output

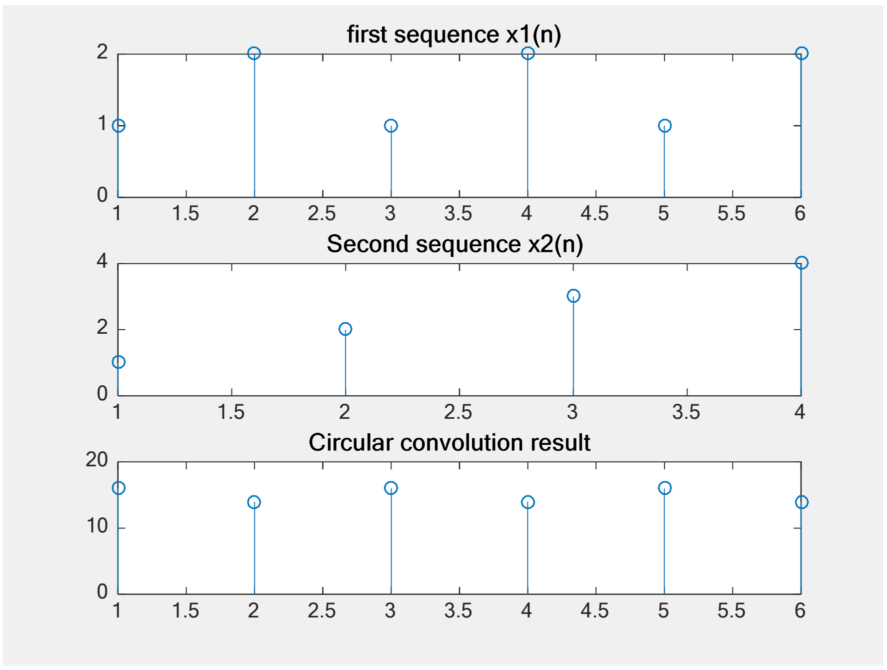
```
Enter the first sequence x1(n) : [1 2 1 2 1 2]

Enter the second sequence x2(n) : [1 2 3 4]

circular convuoultion of the result is

c =

   16    14    16    14    16    14
```



Parseval's theorem :

It is an important theorem used to relate the product or square of functions using their respective Fourier series components. Theorems like Parseval's theorem are helpful in signal processing, studying behaviours of random processes, and relating functions from one domain to another. Parseval's theorem states that the integral of the square of its function is equal to the square of the function's Fourier components.

This theorem is helpful when dealing with signal processing and when observing the behaviour of random processes. When signals are challenging to process with time as their domain, transforming the domain is the best course of action so that the values are easier to work with. This is where Fourier transforms and the Parseval's theorem enters. Taking a look at the equation of Parseval's theorem for continuous functions, a signal's power (or energy) will be much easier to capitalize on and will provide insight on how these quantities behave through a different domain, say frequency.

$$\sum_{i=0}^{n-1} |x_i|^2 = \frac{1}{n} \sum_{k=0}^{n-1} |x_k|^2$$

Program

```
%Program to verify the Parseval's theorem
clear all
close all
clc
N=5000;
x=randi(10,N,1)+1i.*randi(10,N,1);
y=randi(10,N,1)+1i.*randi(10,N,1);
%To find x(n).y*(n)
yc=conj(y);
disp('Left hand side of Parseval's theorem')
lhs=sum(x.*yc)
X=fft(x);
Y=fft(y);
Yc=conj(Y);
%To find (1/N).X(n).Y*(n)
disp('right hand side of Parseval's theorem')
rhs=(1/N).*sum(X.*Yc)
```

Output

```
Left hand side of Parseval's theorem

lhs =

  3.0408e+05 + 2.0650e+03i

right hand side of Parseval's theorem

rhs =

  3.0408e+05 + 2.0650e+03i
```