1.linear convolution without conv

```
clc
clear
close all

x=[1,2,3,4,5];
h=[1,3,5,7];
len_x = length(x);
len_h = length(h);

% Calculate the length of the output sequence
len_y = len_x + len_h - 1;

% Initialize the output sequence with zeros
y = zeros(1, len_y);

% Perform the convolution operation manually
for i = 1:len_x
    for j = 1:len_h
        y(i + j - 1) = y(i + j - 1) + x(i) * h(j);
    end
end

% Display the result
disp('Input sequence x:');
disp(x);
disp('Impulse response h:');
disp(h);
disp('Convolved sequence y:');
disp(y);
```

2.linear convolution with conv

```
x = [1, 2, 3];
h = [0, 1, 0.5];
y = conv(x, h);

% Display the result
disp('Input sequence x:');
disp(x);
disp('Impulse response h:');
disp(h);
disp('Convolved sequence y:');
disp(y);
```

3. circular convolution without circshift

```
x = [1, 2, 3, 4];
h = [1, 2];
len_x = length(x);
len_h = length(h);
N = max(len_x, len_h);

% Zero-pad the shorter sequence to match the length of the longer
sequence
x = [x, zeros(1, N - len_x)];
h = [h, zeros(1, N - len_h)];

% Initialize the output sequence with zeros
y = zeros(1, N);

% Perform the circular convolution operation
for n = 1:N
    for m = 1:N
        k = mod(n - m, N) + 1;
        y(n) = y(n) + x(m) * h(k);
    end
end

disp('Input sequence x:');
disp(x);
disp('Impulse response h:');
disp(h);
disp('Circularly convolved sequence y:');
disp(y);
```

4.circular convolution with circshift

```
clc;
clear;
close all;
x = [1, 2, 3, 4];
h = [1, 2];
N1=length(x);
N2=length(h);
N=max(N1,N2);%Length of Circular convolved output sequence
if(N1>N2)
    h=[h,zeros(1,N1-N2)];
else
    x=[x,zeros(1,N2-N1)];
end

x=transpose(x);
```

```
h=transpose(h);
temp=h;

for i=1:N-1
    temp=circshift(temp,1);
    h=horzcat(h,temp);
end

y=h*x;
disp(y)
```

5. DFT

```
x = [1, 2, 3, 4];
N = length(x);
X = zeros(1, N);

for k = 0:N-1
    for n = 0:N-1
        X(k+1) = X(k+1) + x(n+1) * exp(-1i * 2 * pi * k * n / N);
    end
end

disp('Input sequence:');
disp(x);
disp('DFT result:');
disp(X);
```

6. IDFT

```
X = [10,-2+ 2i, -2, -2- 2i];
N = length(X);
x = zeros(1, N);

for n = 0:N-1
    for k = 0:N-1
        x(n+1) = x(n+1) + X(k+1) * exp(1i * 2 * pi * k * n / N);
    end
    x(n+1) = x(n+1) / N;
end

% Display the result
disp('DFT coefficients:');
disp(X);
disp('IDFT result:');
disp(x);
```

7. DIF-FFT

```
x = [1, 2, 3, 4];
N = length(x);
S = log2(N);
X = x;
for s = 1:S
    W = exp(-1i * 2 * pi / 2^s);
    for o = 0:2^(s-1):N-1
        for k = 0:2^(s-1)-1
            e = o + k + 1;
            o = e + 2^(s-1);
            if o <= N
                t = X(e) + W^k * X(o);
                X(o) = X(e) - W^k * X(o);
                X(e) = t;
            end
        end
    end
end
disp('Input sequence:');
disp(x);
disp('DIF-FFT result:');
disp(X);
```

8. FIR filter design

```
Fs = 1000;         % Sampling frequency (Hz)
Fpass = 100;       % Passband frequency (Hz)
Fstop = 200;       % Stopband frequency (Hz)
wn=Fstop / (Fs / 2);

% Filter order estimation
delta_f = Fstop - Fpass;
N = ceil(3.3 * Fs / delta_f);

% Rectangular Window
h_rect = fir1(N,wn, 'low', rectwin(N+1));
freqz(h_rect);
title('Frequency Response of Rectangular Window FIR Filter');


% Hanning Window
h_hanning = fir1(N, wn, 'low', hanning(N+1));
figure;
freqz(h_hanning);
title('Frequency Response of Hanning Window FIR Filter');
```

```
% Hamming Window
h_hamming = fir1(N,wn, 'low', hamming(N+1));
figure;
freqz(h_hamming);
title('Frequency Response of Hamming Window FIR Filter');

% Triangular Window
h_triangular = fir1(N, wn, 'low', triang(N+1));
figure;
freqz(h_triangular);
title('Frequency Response of Triangular Window FIR Filter');

% Blackman Window
h_blackman = fir1(N, wn, 'low', blackman(N+1));
figure;
freqz(h_blackman);
title('Frequency Response of Blackman Window FIR Filter');

% Kaiser Window
beta = 0.5; % Beta parameter for Kaiser window
h_kaiser = fir1(N, wn, 'low', kaiser(N+1, beta));
```

9. Butterworth HPF

```
Fs = 1000;
Fn = Fs/2;
Fc = 200;
Wp = Fc / Fn;
Ws = (Fc - 50) / Fn;
Rp = 1;  % Passband ripple (dB)
Rs = 40;  % Stopband attenuation (dB)

% Calculate the order and cutoff frequency
[N, Wn] = buttord(Wp, Ws, Rp, Rs);

% Design Butterworth high-pass filter
[b, a] = butter(N, Wn, 'high');
[h, f] = freqz(b, a, 1024, Fs);

figure;
plot(f, abs(h));
title('Butterworth High-Pass Filter Frequency Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

10. Butterworth LPF

```
Fc = 200;
Wp = Fc / Fn;
Ws = (Fc + 50) / Fn;
Rp = 1;  % Passband ripple (dB)
Rs = 40;  % Stopband attenuation (dB)

% Calculate the order and cutoff frequency
[N, Wn] = buttord(Wp, Ws, Rp, Rs);

% Design Butterworth low-pass filter
[b, a] = butter(N, Wn, 'low');
[h, f] = freqz(b, a, 1024, Fs);

figure;
plot(f, abs(h));
title('Butterworth Low-Pass Filter Frequency Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

11. Chebyshev BPF

```
Wp = [100 300] / Fn;
Ws = [80 350] / Fn;
Rp = 1;
Rs = 40;

% Calculate the order and cutoff frequency
[N, Wn] = cheb1ord(Wp, Ws, Rp, Rs);

% Design Chebyshev Type I band-pass filter
[b, a] = cheby1(N, Rp, Wn, 'bandpass');
[h, f] = freqz(b, a, 1024, Fs);

figure;
plot(f, abs(h));
title('Chebyshev Type I Band-Pass Filter Frequency Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

12. Chebyshev BSF

```
Wp = [100 300] / Fn;
Ws = [120 280] / Fn;
Rp = 1;  % Passband ripple (dB)
Rs = 40;  % Stopband attenuation (dB)

[N, Wn] = cheb1ord(Wp, Ws, Rp, Rs);
[b, a] = cheby1(N, Rp, Wn, 'stop');
[h, f] = freqz(b, a, 1024, Fs);

figure;
plot(f, abs(h));
title('Chebyshev Type I Band-Stop Filter Frequency Response');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

13. Decimation and interpolation

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
M = 3; %decimation factor
L = 2;

y_decimated = x(1:M:end); %performing decimation

%performing interpolation
y_upsampled = zeros(1, L * length(x));
y_upsampled(1:L:end) = x;

% Define a simple low-pass filter for interpolation
h = ones(1, L) / L;
y_interpolated = conv(y_upsampled, h, 'same');


disp('Original sequence:');
disp(x);
disp('Decimated sequence:');
disp(y_decimated);
disp('Upsampled and interpolated sequence:');
disp(y_interpolated);

figure;
subplot(3,1,1);
```

```
stem(x, 'filled');
title('Original Sequence');
xlabel('Sample');
ylabel('Amplitude');
grid on;

subplot(3,1,2);
stem(y_decimated, 'filled');
title('Decimated Sequence');
xlabel('Sample');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
stem(y_interpolated, 'filled');
title('Upsampled and Interpolated Sequence');
xlabel('Sample');
ylabel('Amplitude');
grid on;
```