TOBB Ekonomi ve Teknoloji Üniversitesi
ELE 371 Sinyaller ve Sistemler
Proje Raporu

Öğrenci Adı Soyadı: BEREN ÜNVEREN
Öğrenci Numarası: 221101006

Bu proje raporu ELE 371 Sinyaller ve Sistemler dersi proje ödevinin gerekliliklerini yerine getirmek amacıyla hazırlanmıştır. Proje belirtilen kurallar çerçevesinde (tek başına çalışma, akademik dürüstlük, anlaşılır ve yeterince detaylı raporlama, MATLAB kodlarının teslimi ve Turnitin) tamamlanmıştır. Raporun her bir bölümünde istenen açıklamalar, matematiksel analizler ve kodlar sunulmuştur.

I. BİRİNCİ KISIM

Bu bölüm içinde Matlab Academy üzerinden tamamlanması istenen çevrimiçi kurslar için oluşturulan Certificate & Progress Report PDF kurs sitesinden instructor'la paylaşım ile ekazikli@etu.edu.tr'ye yönlendirilmiştir.

II. İKİNCİ KISIM

Bu kısım, verilen periyodik bir sinyalin Fourier serisi analizini içermektedir.

1. **a, b, c, d, e, f Değerlerinin Elde Edilmesi**
   Bu adımda, öğrenci numarası kullanılarak rng fonksiyonu ile rastgele a, b, c, d, e, f tam sayı değerleri üretilmiştir. Bu değerler projenin sonraki adımlarında kullanılan sinyal tanımının parametrelerini oluşturmaktadır.

```
studentId = 221101006;
rng(studentId);

a = randi(4);
b = randi(4);
c = randi(4);
d = randi(4);
e = randi(4);
f = randi(4);

fprintf('a = %d\n', a);
fprintf('b = %d\n', b);
fprintf('c = %d\n', c);
fprintf('d = %d\n', d);
fprintf('e = %d\n', e);
fprintf('f = %d\n', f);
```

```
a = 1
b = 3
c = 2
d = 4
e = 1
f = 2
```

2. x(t) Sinyalinin Matematiksel İfadesinin Çıkarımı

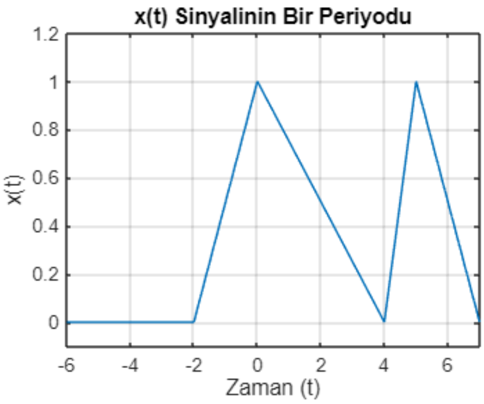
Temel periyodu T = a + b + c + d + e + f olan x(t) sinyalinin bir periyodu grafikte verilmiştir. Bu grafiksel gösterimden yola çıkarak x(t)'nin parçalı matematiksel ifadesi türetilmiştir. Sinyal farklı zaman aralıklarında farklı doğrusal fonksiyonlardan oluşmaktadır:

- -c <= t < 0 aralığı için: x(t) = (1/c)t + 1 (t=-c'de 0, t=0'da 1)
- 0 <= t < d aralığı için: x(t) = (-1/d)t + 1 (t=0'da 1, t=d'de 0)
- d <= t < d+e aralığı için: x(t) = (1/e)(t-d) (t=d'de 0, t=d+e'de 1)
- d+e <= t <= d+e+f aralığı için: x(t) = 1 – (1/f)(t-(d+e)) (t=d+e'de 1, t=d+e+f'de 0)

2) T = a + b + c + d + e + f temel dönemli x(t) sinyali çıkarımı:

```
dt = 0.01;
t = -a-b-c:dt:d+e+f;
xt = zeros(size(t));
idx = 1:length(t);
xt(idx)=(((t(idx)>=-c & t(idx)<0) .* ((1/c)*t(idx)+1)) + ... %x(t
        ((t(idx)>=0 & t(idx)<d) .* ((-1/d)*t(idx)+1)) + ... %x(t)
        ((t(idx)>=d & t(idx)<d+e) .* ((1/e)*(t(idx)-d))) + ... %x
        ((t(idx)>=d+e & t(idx)<=d+e+f) .* (1-(1/f)*(t(idx)-(d+e))
        );

figure;
plot(t, xt);
title('x(t) Sinyalinin Bir Periyodu');
xlabel('Zaman (t)');
ylabel('x(t)');
grid on;
ylim([-0.1 1.2]);
xlim([-a-b-c d+e+f])
```

f = 2



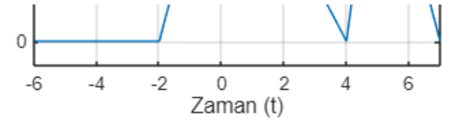
3. x(t) Sinyalinin Fourier Dizisi Katsayıları (ak)

x(t) sinyalinin Fourier dizisi katsayıları ak teorik olarak elde edilmiştir(ak = (1/T) ∫ x(t)e^(-jkω_0t) dt). a0 ve k != 0 durumları için integraller ayrı hesaplanmıştır.
a0 için integral ve k ≠ 0 için karmaşık üstel integrali parçalı x(t) fonksiyonu üzerinden detaylı bir şekilde türetilmiştir.

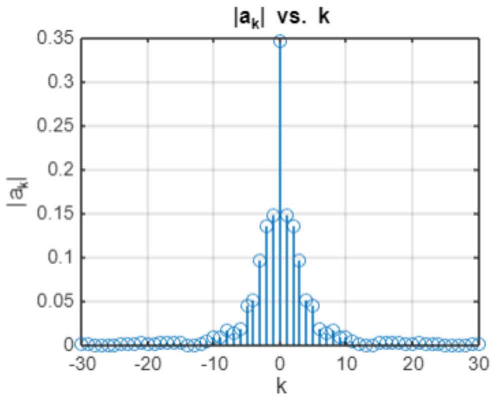
- dc bileşen a0 = (1/T) ∫x(t) dt
- ak = (1/T) [∫(-c)^0 ((1/c)t+1)e^(-jkwot) dt + ∫0^d ((-1/d)t+1)e^(-jkwot) dt + ∫d^(d+e) (1/e)(t-d)e^(-jkwot) dt + ∫(d+e)^(d+e+f) (1 – (1/f)(t-(d+e)))e^(-jkwot) dt]
- her bir integral teriminin detaylı üretimi uzun olduğundan MATLAB’ın anonim fonksiyonlarından faydalanılıp integral fonksiyonu yazılmıştır.
- grafikte |a_k| değerlerinin k=0 etrafında en yüksek olduğu ve k değeri arttıkça azaldığı görülmektedir, bu da sinyalin büyük kısmının düşük frekans bileşenlerinden oluştuğunu gösterir.

3) x(t) Fourier katsayıları: abs(ak) k'ya göre, k=-30,30 için stem komutu ile, a0 ve k!=0 hesabı ayrı. analiz ve grafik.

ak = (1/T) ∫ x(t)e^(-jkω₀t) dt -> burada parcalı integral olacak




```
T = a+b+c+d+e+f;
w0 = 2*pi/T;
integrand = @(t, k)(((t>=-c & t<0) .* ((1/c)*t+1)) + ... %x(t)=(1,
                    ((t>=0 & t<d) .* ((-1/d)*t+1)) + ... %x(t)=(-:
                    ((t>=d & t<d+e) .* ((1/e)*(t-d))) + ... %x(t):
                    ((t>=d+e & t<=d+e+f) .* (1-(1/f)*(t-(d+e))))
                    ).* exp(-1j*k*w0*t);
k_vals = -30:30;
ak = zeros(size(k_vals));
ak(k_vals == 0) = (0.5 * (c+d) + 0.5 * (e+f))/T; %ak=0 durumu
k_nonzero_indices = find(k_vals ~= 0);

for i = 1:length(k_nonzero_indices)
    idx = k_nonzero_indices(i);
    ki = k_vals(idx);
    integral_result = integral(@(t) integrand(t, ki), -c, d+e+f);
    ak(idx) = (1/T) * integral_result;
end

figure;
stem(k_vals, abs(ak));
title('|a_k| vs. k');
xlabel('k');
ylabel('|a_k|');

grid on;
xlim([-30 30]);
```

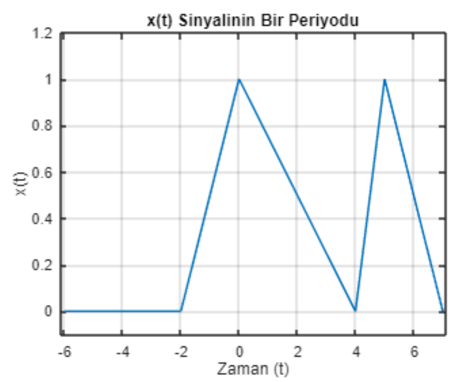


4. x(t) Sinyalinin Bir Döneminin Matlab'da Çizdirilmesi

x(t) sinyalinin bir periyodunun sürekli zamanda temsili ayrık zaman noktaları belirlenerek yapılmıştır. t = -a - b - c'den t = d + e + f'ye kadar 0.01 saniye aralıklarla zaman noktaları tanımlanmış ve x(t)'nin bu noktalardaki değerleri hesaplanmıştır.

4) x(t) sinyalinin 1 dönemi 0.01s aralıklarla t = -a-b-c:dt:d+e+f içinde zaman

```
delta_t = 0.01;
zaman = (-a - b - c) : delta_t : (d + e + f);
xt = zeros(size(zaman));

xt = (((zaman >= -c & zaman < 0) .* ((1/c)*zaman + 1)) + ... %(1/c)t+1
      ((zaman >= 0 & zaman < d) .* ((-1/d)*zaman + 1)) + ... %(-1/d)t+1
      ((zaman >= d & zaman < d + e) .* ((1/e)*(zaman - d))) + ... %(1/e)(t-d)
      ((zaman >= d + e & zaman <= d + e + f) .* (1 - (1/f)*(zaman - (d + e)))) ... %1-(1/f)(t-d-e)
     );

figure;
plot(zaman, xt);
title('x(t) Sinyalinin Bir Periyodu');
xlabel('Zaman (t)');
ylabel('x(t)');
grid on;
ylim([-0.1 1.2]);
xlim([-a - b - c - 0.1, d + e + f + 0.1]);
```

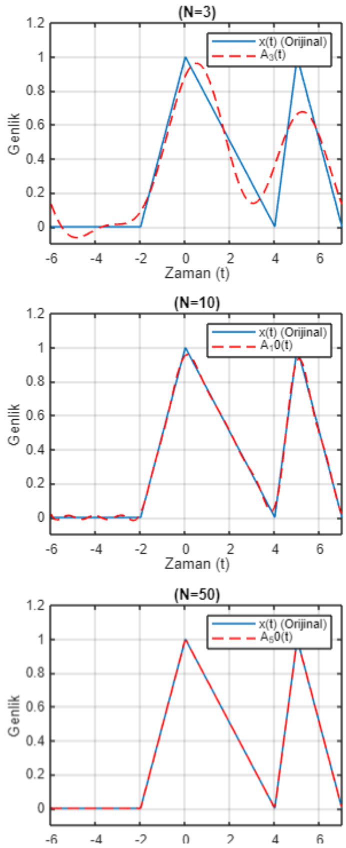


5. Fourier Dizileri ile Yakınsama (x̂N(t))

Periyodik x(t) sinyalinin Fourier dizileri ile x̂N(t) = Σ(k=-N'den N'ye) ak * e^(jk(2π/T)t) şeklinde temsil edilebildiği gösterilmiştir. N = 3, 10, 50 değerleri için x̂N(t) sinyalinin grafikleri orijinal x(t) sinyali ile birlikte çizilerek yakınsama gözlemlenmiştir ve N değeri arttıkça $\hat{x}_N(t)$'nin x(t)'ye daha iyi yaklaştığı ve özellikle x(t)'nin sürekli olan kısımlarında N artışının olumlu etkilerinin bariz görüldüğü söylenebilir. $\hat{x}_N(t)$'nin x(t)'de süreksiz olan noktalarda orijinal sinyal değerini yakalayamaması da Gibbs fenomeninin sonucudur.

5) Fourier ile yakınsama

```
k_vals_new = -50:50;
ak_new = zeros(size(k_vals_new));
ak_new(k_vals_new == 0) = (0.5 * (c+d+e+f)) / T;

k_nonzero_indices_new = find(k_vals_new ~= 0);
for i = 1:length(k_nonzero_indices_new)
    idx = k_nonzero_indices_new(i);
    ki = k_vals_new(idx);
    integral_result = integral(@(t_val) integrand(t_val, ki), -c, d+e+f);
    ak_new(idx) = (1/T) * integral_result;
end

Ns = [3, 10, 50];
for N = Ns
    AN_t = zeros(size(t));
    current_k_indices = (k_vals_new >= -N & k_vals_new <= N);
    current_k_vals = k_vals_new(current_k_indices);
    current_ak = ak_new(current_k_indices);

    for k_idx = 1:length(current_k_vals)
        k_val = current_k_vals(k_idx);
        ak_val = current_ak(k_idx);
        AN_t = AN_t + ak_val * exp(1j * k_val * w0 * t);
    end

    figure;
    plot(t, xt);
    hold on;
    plot(t, real(AN_t), 'r--');
    hold off;
    title(sprintf('(N=%d)', N));
    xlabel('Zaman (t)');
    ylabel('Genlik');
    legend('x(t) (Orijinal)', sprintf('A_%d(t)', N));
    grid on;
    ylim([-0.1 1.2]);
    xlim([-a-b-c d+e+f]);
end
```



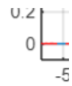
III. ÜÇÜNCÜ KISIM

Bu kısım Sürekli Zaman Fourier Dönüşümü uygulamalarını ve özelliklerini incelemektedir.

1. **a, b, c Değerlerinin Elde Edilmesi**
   Bu adımda öğrenci numarası kullanılarak rng fonksiyonu ile rastgele a, b, c tam sayı değerleri üretilmiştir. Bu değerler sinc fonksiyon tabanlı sinyallerin tanımlanmasında kullanılacaktır.

ÜÇÜNCÜ KISIM

```
studentId = 221101006;
rng(studentId)
a = randi(10);
b = a+randi(10);
c = randi(10);

fprintf('a = %d\n', a);
fprintf('b = %d\n', b);
fprintf('c = %d\n', c);
```



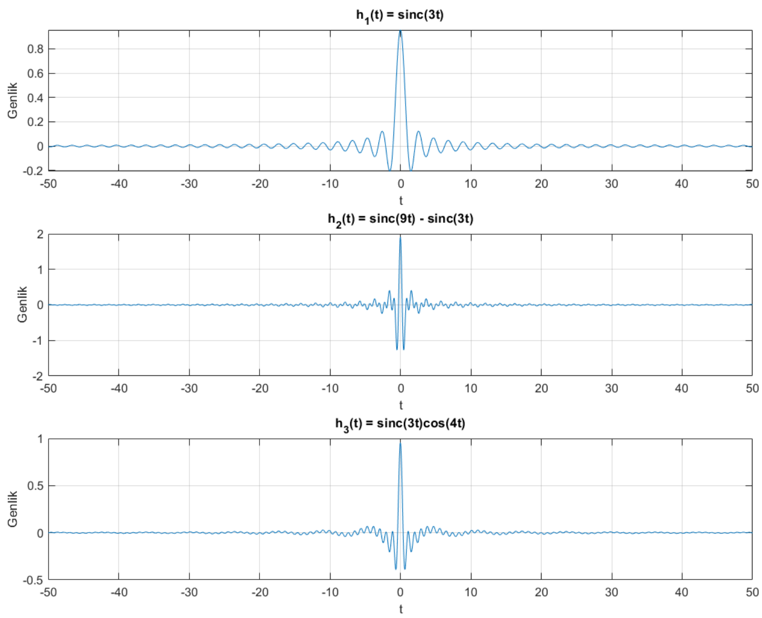
```
a = 3
b = 9
c = 4
```

2. h1(t), h2(t), h3(t) Sinyallerinin Çizdirilmesi

sinc fonksiyonu $\mathrm{sinc}(\alpha t) = \sin(\alpha t) / (\pi t)$ olarak dokümanda tanımlanmıştır ve t=0 noktasındaki özel durumun ele alınmak istenmesi sebebiyle ayrı bir fonksiyon olarak MATLAB’da tekrar yazılmıştır. t=0 noktasındaki özel durum L'Hospital kuralı ile α/π olarak bulunur. h1(t) = sinc(at), h2(t) = sinc(bt) - sinc(at) ve h3(t) = sinc(at)cos(ct) sinyalleri t (-50, 50) aralığında 0.01 saniye aralıklarla örneklenerek çizdirilmiştir.

2) h1(t) = sinc(at), h2(t) = sinc(bt) − sinc(at) ve h3(t) = sinc(at) cos(ct) sinyallerini $t \in (-50, 50)$ zaman aralığında ve zaman ekseninde 0.01 saniye aralıklarla sinyallerden örnekler alarak Matlab’da plot komutunu kullanarak ayrı ayrı grafiklerde çizim:

```
function result = sinc(val_t, val_a)
    result = zeros(size(val_t));
    zero_indices = (val_t == 0);
    result(zero_indices) = val_a / pi;
    non_zero_indices = (val_t ~= 0);
    result(non_zero_indices) = sin(val_a * val_t(non_zero_indices)) ./ (pi * val_t(non_zero_indices));
end

t_sinc = -50:0.01:50;
h1_t = sinc(t_sinc, a);
h2_t = sinc(t_sinc, b) - sinc(t_sinc, a);
h3_t = sinc(t_sinc, a) .* cos(c * t_sinc);

figure;
subplot(3,1,1);
plot(t_sinc, h1_t);
title(sprintf('h_1(t) = sinc(%dt)', a));
xlabel('t'); ylabel('Genlik'); grid on; xlim([-50 50]);

subplot(3,1,2);
plot(t_sinc, h2_t);
title(sprintf('h_2(t) = sinc(%dt) - sinc(%dt)', b, a));
xlabel('t'); ylabel('Genlik'); grid on; xlim([-50 50]);

subplot(3,1,3);
plot(t_sinc, h3_t);
title(sprintf('h_3(t) = sinc(%dt)cos(%dt)', a, c));
xlabel('t'); ylabel('Genlik'); grid on; xlim([-50 50]);
```



3. h1(t), h2(t), h3(t) Sinyallerinin Sürekli Zaman Fourier Dönüşümleri ve Büyüklük Grafikleri

h1(t), h2(t) ve h3(t) sinyallerinin sürekli zaman Fourier dönüşümleri teorik olarak elde edilmiştir. sinc fonksiyonunun Fourier dönüşümünün dikdörtgen fonksiyonu olduğu ve modülasyon gibi özellikler internetten araştırılarak hesaplama kolaylığı adına kullanılmıştır.

- ★ $FT\{sinc(\alpha t)\} = rect(\omega/(2\alpha))$,
- ★ $FT\{h1(t)\} = H1(j\omega) = rect(\omega/(2a))$,
- ★ $FT\{h2(t)\} = H2(j\omega) = rect(\omega/(2b)) - rect(\omega/(2a))$,
- ★ $FT\{h3(t)\} = FT\{sinc(at) * (1/2)(e^{(jct)} + e^{(-jct)})\} = (1/2) * [H1(j(\omega-c)) + H1(j(\omega+c))] = (1/2) * [rect((\omega-c)/(2a)) + rect((\omega+c)/(2a))]$.

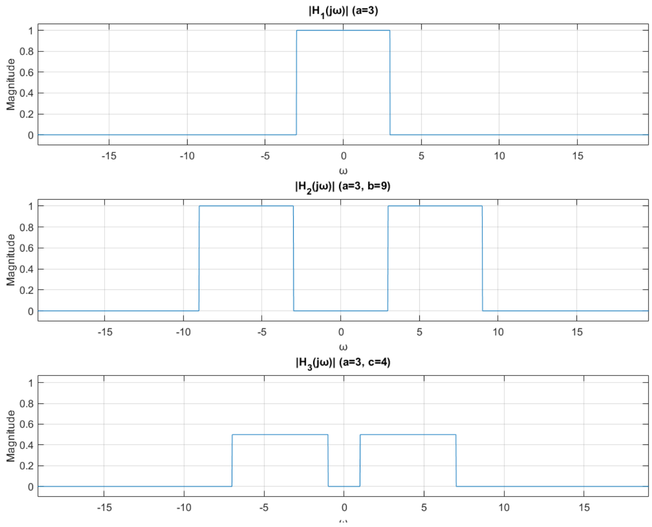
3) h1_t, h2_t, h3_t CTFTleri ve magnitudeları:

FT { sin(A*t) / (pi*t) } = rect(ω / (2A)) h1 icin

h2 FT icin FT(sinc(bt))-FT(sinc(at))

h3 icin normalde iki fonk carpımıysa FTde konva karsılık gelir deriz

```
w_vals = -50:0.01:50;

H1_jw = zeros(size(w_vals));
H1_jw(abs(w_vals) < a) = 1;

H2_jw = zeros(size(w_vals));
H2_jw(abs(w_vals) < b) = 1;
H2_jw(abs(w_vals) == b) = 0.5;
H2_jw(abs(w_vals) < a) = H2_jw(abs(w_vals) < a) - 1;
H2_jw(abs(w_vals) == a) = H2_jw(abs(w_vals) == a) - 0.5;

H3_jw = zeros(size(w_vals));
H3_jw(abs(w_vals - c) < a) = H3_jw(abs(w_vals - c) < a) + 0.5;
H3_jw(abs(w_vals - c) == a) = H3_jw(abs(w_vals - c) == a) + 0.25;
H3_jw(abs(w_vals + c) < a) = H3_jw(abs(w_vals + c) < a) + 0.5;
H3_jw(abs(w_vals + c) == a) = H3_jw(abs(w_vals + c) == a) + 0.25;

figure;
subplot(3,1,1);
plot(w_vals, abs(H1_jw));
title(sprintf('|H_1(jω)| (a=%d)', a));
xlabel('ω'); ylabel('Magnitude'); grid on;
xlim([-50 50]); ylim([-0.1 1.1]);
subplot(3,1,2);
plot(w_vals, abs(H2_jw));
title(sprintf('|H_2(jω)| (a=%d, b=%d)', a, b));
xlabel('ω'); ylabel('Magnitude'); grid on;
xlim([-50 50]); ylim([-0.1 1.1]);
subplot(3,1,3);
plot(w_vals, abs(H3_jw));
title(sprintf('|H_3(jω)| (a=%d, c=%d)', a, c));
xlabel('ω'); ylabel('Magnitude'); grid on;
xlim([-50 50]); ylim([-0.1 1.1]);
```



- ★ h1(t): Bir lowpass filter gibi davranır. Belirli bir kesim frekansına kadar olan frekans bileşenlerini geçirir.
- ★ h2(t): Bir bandpass filter gibi davranır. İki farklı kesim frekansı arasındaki frekansları geçirir.
- ★ h3(t): Bir bandpass filter gibi davranır, ancak sinyali taşıyıcı bir frekans etrafında (c ve -c) kaydırır, bu da modülasyonun bir sonucudur.

4. h4(t) ve h5(t) Sinyallerinin Sürekli Zaman Fourier Dönüşümleri ve Büyüklük Grafikleri

h4(t) = h1(t)h2(t) ve h5(t) = h2(t) ⋆ h3(t) sinyallerinin CTFT'leri Fourier dönüşümü özelliklerinden yararlanılarak teorik olarak elde edilmiştir. Zamanda çarpım frekansta evrişime, zamanda evrişim ise frekansta çarpmaya karşılık gelir.

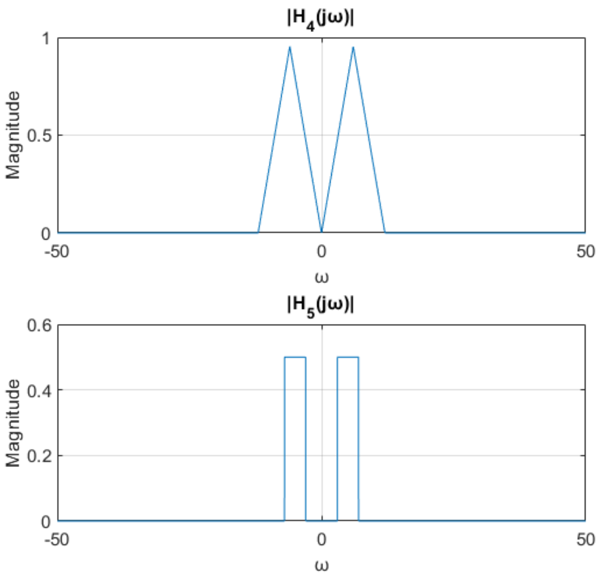
h4(t) = h1(t)h2(t) olduğundan H4(jω) = (1/(2π)) ⋆ (H1(jω) ⋆ H2(jω)) (frekans uzayında conv) h5(t) = h2(t) ⋆ h3(t) olduğundan H5(jω) = H2(jω) ⋆ H3(jω) (frekans uzayında çarpım)

4) h4(t) = h1(t)h2(t) ve h5(t) = h2(t) ∗ h3(t) CTFTleri ve magnitudeları: *frekans ekseninde 0.01 radyan/saniye aralıklarla fonksiyonlardan örnekler alın*

h4 için h1 h2 konv

h5 için h2 ve h3 çarpım

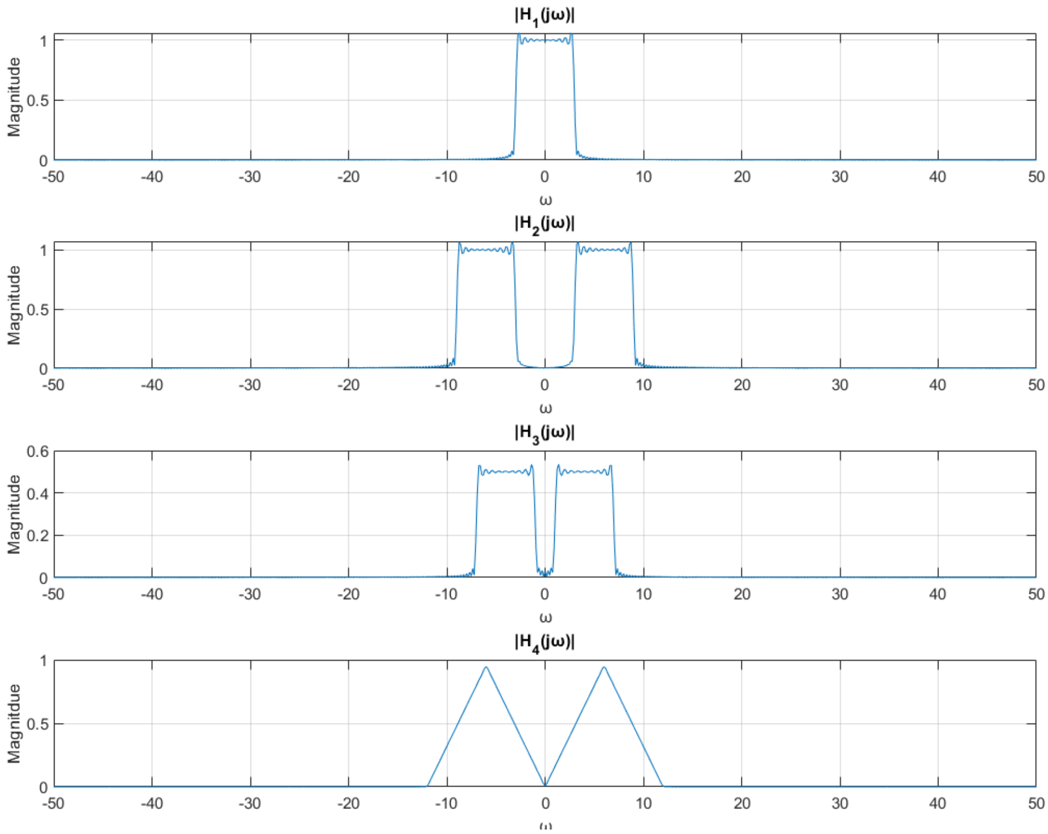
```
H4_jw_conv = (1/(2*pi)) * conv(H1_jw, H2_jw, 'same') * (w_vals(2)-w_vals(1));
H5_jw = H2_jw .* H3_jw;

figure;
subplot(2,1,1);
plot(w_vals, abs(H4_jw_conv));
title('|H_4(jω)|');
xlabel('ω'); ylabel('Magnitude'); grid on; xlim([-50 50]);
subplot(2,1,2);
plot(w_vals, abs(H5_jw));
title('|H_5(jω)|');
xlabel('ω'); ylabel('Magnitude'); grid on; xlim([-50 50]);
```



5. **FFT ve FFTshift Kullanarak CTFT Hesabı ve Karşılaştırma**
Bu bölümde h1(t), h2(t), h3(t), h4(t) sinyallerinin sürekli zaman Fourier dönüşümleri MATLAB'daki fft ve fftshift komutları kullanılarak hesaplanmıştır. t (-10, 10) aralığında Ts = 0.01 saniye aralıklarla örnekler alınmış ve N = 4096 için zero padding uygulanmıştır. Elde edilen H[k] değerleri formül ile(H1[k] = (1/Ts) * H1(jω)|ω=2π(k-N/2-1)/(NTs)) CT frekans eksenine eşlenerek çizdirilmiştir.
zero padding uygulaması giriş sinyalinin sonunu sıfırla doldurarak sinyal uzunluğunu N_fft ye çıkarma işlemidir, DFT'nin daha fazla noktada hesaplanmasını ve spektrumun daha düzgün örneklenmesini sağlar. (hesaplama konusunda kafa karışıklığı sonucu internete başvurulmuştur)

5) fft - fftshift ile CTFT, zero padding

```
Ts = 0.01;
t_fft = -10:Ts:10-Ts;
h1_t_fft = sinc(t_fft, a);
N_fft = 4096;
h1_t_padded = [h1_t_fft, zeros(1, N_fft - length(h1_t_fft))]; % padding
H1_k = fft(h1_t_padded);
H1_k_shifted = fftshift(H1_k);

Fs_fft = 1/Ts;
w_fft = linspace(-pi*Fs_fft, pi*Fs_fft - (2*pi*Fs_fft)/N_fft, N_fft);

figure;
subplot(4,1,1); plot(w_fft, Ts * abs(H1_k_shifted)); title('|H_1(jω)|'); xlabel('ω'); ylabe

h2_t_fft = sinc(t_fft, b) - sinc(t_fft, a);
h3_t_fft = sinc(t_fft, a) .* cos(c * t_fft);
h4_t_fft = h1_t_fft .* h2_t_fft;
H2_k = fft(h2_t_fft, N_fft); H2_k_shifted = fftshift(H2_k);
H3_k = fft(h3_t_fft, N_fft); H3_k_shifted = fftshift(H3_k);
H4_k = fft(h4_t_fft, N_fft); H4_k_shifted = fftshift(H4_k);

subplot(4,1,2); plot(w_fft, Ts * abs(H2_k_shifted)); title('|H_2(jω)|'); xlabel('ω'); ylabe
subplot(4,1,3); plot(w_fft, Ts * abs(H3_k_shifted)); title('|H_3(jω)|'); xlabel('ω'); ylabe
subplot(4,1,4); plot(w_fft, Ts * abs(H4_k_shifted)); title('|H_4(jω)|'); xlabel('ω'); ylabe
```



Elde edilen grafikler fft ve fftshift kullanılarak uygun örnekleme aralığı ve yeterli sıfır doldurmayla CTFT'lerin sayısal olarak doğru ve güvenilir bir şekilde tahmin edilebildiğini göstermektedir.

IV. DÖRDÜNCÜ KISIM

Bu kısım ayrık zamanlı DZD sistemlerin analizini ve farklı yöntemlerle çıkış sinyallerinin hesaplanmasını içermektedir.

1. **a, b, c Değerlerinin Elde Edilmesi**
   Öğrenci numarası kullanılarak rng ile rastgele a, b, c tam sayı değerleri üretilmiştir. Bu değerler ayrık zamanlı sistemin katsayılarında kullanılmıştır.

```
studentId = 221101006;
rng(studentId)
a = 2+randi(8)
b = 2+randi(8)
c = 2+randi(8)
```

```
a = 4
b = 7
c = 6
```

2. **Dürtü Yanıtı h[n]'nin Hesaplanması ve Çizdirilmesi**
   Doğrusal ve zamanla değişmez ayrık zaman sisteminin giriş-çıkış ilişkisi y[n] = -(1/a)y[n-1] + (1/b)y[n-2] + (1/c)x[n] olarak verilmiştir. x[n] = δ[n] (birim dürtü) girişi için çıkış sinyali olan dürtü yanıtı h[n] recursive denklem kullanılarak hesaplanmıştır.

2) DZD ayrık zaman sistemde :

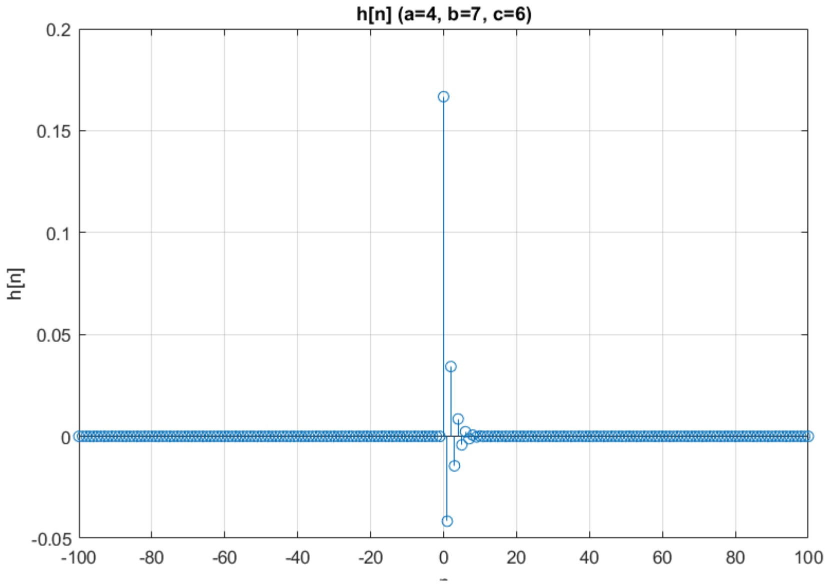
y[n] = −(1/a)y[n − 1] + (1/b)y[n − 2] + (1/c)x[n].

x[n] = δ[n] için recursive elde edilecek y[n]:

h[n] = -(1/a)h[n-1] + (1/b)h[n-2] + (1/c)δ[n]

recursion base case: h[-1] = 0; h[-2] = 0.

```
n_start = -100;
n_end = 100;
n_vals = n_start:n_end;
h_n = zeros(size(n_vals));
idx_0 = find(n_vals == 0);
h_n(idx_0) = 1 / c;
if (idx_0 + 1) <= length(n_vals)
    h_n(idx_0 + 1) = -(1 / a) * h_n(idx_0);
end

for k = (idx_0 + 2):length(n_vals)
    h_n(k) = -(1 / a) * h_n(k-1) + (1 / b) * h_n(k-2);
end

figure;
stem(n_vals, h_n);
title(sprintf('h[n] (a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('h[n]');
grid on;
xlim([-100 100]);
```



- h[n] = -(1/a)h[n-1] + (1/b)h[n-2] + (1/c)δ[n]
- n < 0 için h[n] = 0 (nedensellik varsayımı)
- n = 0 için: h = -(1/a)h[-1] + (1/b)h[-2] + (1/c)δ = 0 + 0 + (1/c) * 1 = 1/c
- n = 1 için: h = -(1/a)h + (1/b)h[-1] + (1/c)δ = -(1/a) * (1/c) + 0 + 0 = -1/(ac)
- n = 2 için: h = -(1/a)h + (1/b)h + (1/c)δ = -(1/a) * (-1/(ac)) + (1/b) * (1/c) + 0 = 1/(a^2c) + 1/(bc)
- ...

3. **Frekans Yanıtı H(e^jω)'nin Teorik Olarak Elde Edilmesi ve Çizdirilmesi**

   Sistemin giriş çıkış ilişkisi frekans uzayına (DTFT) alınarak frekans yanıtı H(e^jω) teorik olarak elde edilmiştir. Ardından |H(e^jω)|'nin grafiği çizilmiştir.

3) Yukarıdaki eşitliği frekans uzayına alarak (ayrık zaman Fourier dönüşümü) sistemin frekans yanıtını H(e^jw)'yı teorik olarak elde edin. Frekans yanıtının grafiğini w ∈ (−π,π) için 0.01 radyan/saniye aralıklarla Matlab'da plot komutunu kullanarak çizdirin. Frekans yanıtını kısmi kesirlere ayrıştırıp (partial fraction expansion) ters Fourier dönüşümü uygulayarak dürtü yanıtını elde edin.

y[n] = -(1/a)y[n-1] + (1/b)y[n-2] + (1/c)x[n]

DTFT{x[n-k]} = X(e^jw)e^(-jwk))

DTFT{y[n]} = DTFT{-(1/a)y[n-1]} + DTFT{(1/b)y[n-2]} + DTFT{(1/c)x[n]}

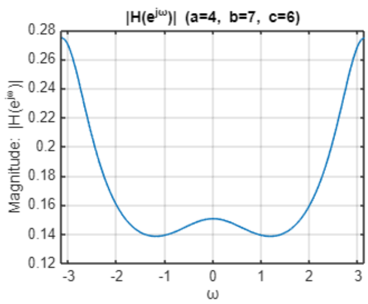
Y(e^jw) = -(1/a)Y(e^jw)e^(-jw) + (1/b)Y(e^jw)e^(-j2w) + (1/c)X(e^jw)

Y(e^jw) + (1/a)Y(e^jw)e^(-jw) - (1/b)Y(e^jw)e^(-j2w) = (1/c)X(e^jw)

Y(e^jw) [1 + (1/a)e^(-jw) - (1/b)e^(-j2w)] = (1/c)X(e^jw)

H(e^jw) = Y(e^jw) / X(e^jw) = (1/c) / [1 + (1/a)e^(-jw) - (1/b)e^(-j2w)]

```
w_dtft = -pi:0.01:pi;
numerator = (1/c);
denominator = (1 + (1/a) * exp(-1j*w_dtft) - (1/b) * exp(-1j*2*w_dtft));
H_ejw = numerator ./ denominator;

figure;
plot(w_dtft, abs(H_ejw));
title(sprintf('|H(e^{jω})| (a=%d, b=%d, c=%d)', a, b, c));
xlabel('ω');
ylabel('Magnitude: |H(e^{jω})|');
grid on;
xlim([-pi pi]);
```



4. **Giriş Sinyali x[n] = (1/2)^n u[n] iken Çıkış Sinyalinin Teorik Hesabı**
   Giriş sinyali x[n] = (1/2)^n u[n] olduğunda çıkış sinyali y[n] teorik olarak hesaplanmıştır. Bu hesaplama için Y(e^jω) = H(e^jω)X(e^jω) özelliği kullanılmış, ardından Y(e^jω) kısmi kesirlere ayrıştırılıp ters DTFT uygulanmıştır.

   x[n] = (1/2)^n u[n] sinyalinin DTFT'si: X(e^jω) = 1/(1-(1/2)e^(-jω))
   Y(e^jω) = H(e^jω)X(e^jω) çarpımı:
   Y(e^jω) = [(1/c) / (1 + (1/a)e^(-jω) - (1/b)e^(-j2ω))] * [1 / (1 - (1/2)e^(-jω))]
   Bu ifade z^(-1) cinsinden yazıp kısmi kesirlere ayrılabilir, ancak zaman kazanmak açısından MATLAB'a çözdürmek tercih edildiğinden bu işi yapacak komutlar araştırılmış ve residuez bulunmuştur. sonrasında da ters z dönüşümü uygulanması gerektiğinden bunun MATLAB karşılığı da araştırılmıştır.
   Y(z) = (1/c) / ((1 + (1/a)z^(-1) - (1/b)z^(-2)) * (1 - (1/2)z^(-1)))
   Y(z) = R1/(1-p1*z^(-1)) + R2/(1-p2*z^(-1)) + ...
   y[n] = (R1*p1^n + R2*p2^n + ...)u[n]

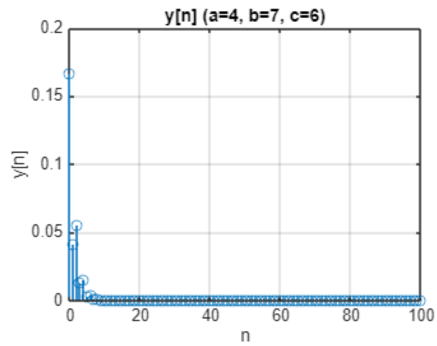
4) Giriş sinyali x[n] = (1/2)^n u[n] olduğu durumda çıkış sinyalini teorik olarak hesaplama

den_Y_z: Payda = (1 - 0.5z^(-1)) * (1 + (1/a)z^(-1) - (1/b)z^(-2))

= 1 + (1/a - 0.5)z^(-1) + (-1/b - 0.5/a)z^(-2) + (0.5/b)z^(-3)

```
num_Y_z = 1/c;
den_Y_z = [1, (1/a - 0.5), (-1/b - 0.5/a), (0.5/b)];
[R, P, K_direct] = residuez(num_Y_z, den_Y_z);

fprintf('Polars: %s\n', mat2str(P));
fprintf('Residuals: %s\n', mat2str(R));
fprintf('K_direct: %s\n', mat2str(K_direct));

n_start_y = 0;
n_end_y = 100;
n_vals_y = n_start_y:n_end_y;

y_n_theoretical = zeros(size(n_vals_y));

for i = 1:length(n_vals_y)
    current_n = n_vals_y(i);
    if current_n >= 0
        sum_terms = 0;
        for j = 1:length(R); sum_terms = sum_terms + R(j) * (P(j)^current_n); er
        if current_n == 0 && ~isempty(K_direct); sum_terms = sum_terms + K_direc
        y_n_theoretical(i) = sum_terms;
    end
end

figure;
stem(n_vals_y, real(y_n_theoretical));
title(sprintf('y[n] (a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('y[n]');
grid on;
xlim([n_start_y n_end_y]);
```

```
Polars: [-0.523098157314428;0.5;0.273098157314428]
Residuals: [0.0559857711355947;0.179487179487179;-
K_direct: []
```



5. **Matlab filter Komutu Kullanımı ile Çıkış Sinyali Hesabı**
   Matlab'daki filter komutu DZD sistemlerin fark denklemleriyle tanımlanan çıkış sinyallerini hesaplamak için kullanılır. Komutun

kullanımı y = filter(b, a, x) şeklindedir; b vektörü sistemin pay katsayılarını (giriş tarafındaki katsayıları), a vektörü ise payda katsayılarını (çıkış tarafındaki katsayılar (y[n]'nin katsayısı 1 olacak şekilde)) temsil eder.
Verilen sistem denklemi: y[n] = -(1/a)y[n-1] + (1/b)y[n-2] + (1/c)x[n].
Yeniden düzenlenmiş hali:y[n] + (1/a)y[n-1] - (1/b)y[n-2] = (1/c)x[n]
Buna göre b = [1/c] ve a = [1, 1/a, -1/b].
x[n] = (1/2)^n u[n] giriş sinyalinden n = -100'den 100'e kadar örnekler oluşturulmuş ve filter kullanılarak çıkış sinyali hesaplanmıştır.

5) filter komutu kullanımı, x[n] = (1/2)^n u[n] n = −100, ... ,100 için örnekler oluşturup xn adından bir vektöre kaydet, filter ile çıkış sinyalini hesapla, n = -20, ... , 20 için stem ile çizdir

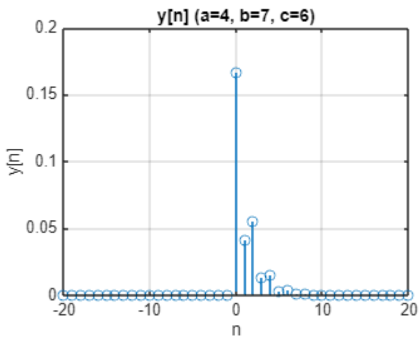
filter açıklama:

y = filter(b, a, x):

b: pay katsayıları vektörü, a: payda katsayıları vektörü, y[n] in katsayısı yani 1 ile başlatılmalı, x: input olarak verilen sinyal, y: output olarak çıkan sinyal

y[n] = -(1/a)y[n-1] + (1/b)y[n-2] + (1/c)x[n] ; y[n] + (1/a)y[n-1] - (1/b)y[n-2] = (1/c)x[n]

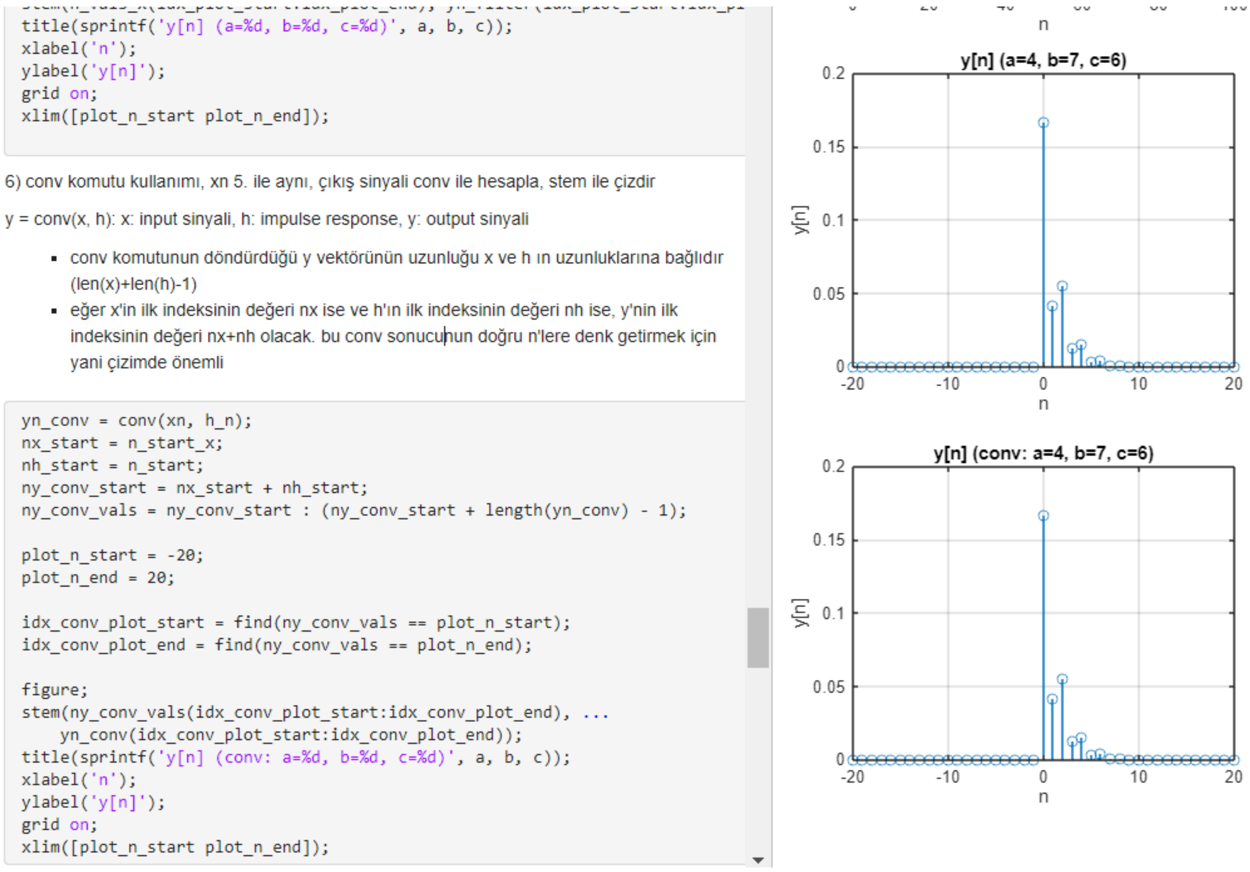
```
n_start_x = -100;
n_end_x = 100;
n_vals_x = n_start_x:n_end_x;
xn = zeros(size(n_vals_x));
idx_n_ge_0 = find(n_vals_x >= 0);
xn(idx_n_ge_0) = (1/2) .^ n_vals_x(idx_n_ge_0);

b_filter = 1/c;
a_filter = [1, (1/a), -(1/b)];

yn_filter = filter(b_filter, a_filter, xn);
plot_n_start = -20;
plot_n_end = 20;

idx_plot_start = find(n_vals_x == plot_n_start);
idx_plot_end = find(n_vals_x == plot_n_end);

figure;
stem(n_vals_x(idx_plot_start:idx_plot_end), yn_filter(idx_plot_start:idx_plot_er
title(sprintf('y[n] (a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('y[n]');
grid on;
xlim([plot_n_start plot_n_end]);
```



6. **Matlab conv Komutu Kullanımı ile Çıkış Sinyali Hesabı**

Matlab'daki conv komutu iki sinyalin ayrık zamanlı evrişimini hesaplamak için kullanılır. Komutun kullanımı y = conv(x, h) şeklindedir; x giriş sinyali, h sistemin dürtü yanıtı ve y çıkış sinyalidir. y vektörünün uzunluğu length(x) + length(h) - 1 kadardır. Evrişim sonucunun doğru zaman indekslerine karşılık gelmesi için giriş sinyali x'in başlangıç indeksi (nx_start) ve dürtü yanıtı h'nin başlangıç indeksi (nh_start) göz önünde bulundurularak çıkış sinyalinin başlangıç indeksi (ny_conv_start = nx_start + nh_start) belirlenmesi gerektiği göz önünde bulundurulmuştur.

x[n] = (1/2)^n u[n] giriş sinyalinden n = -100'den 100'e kadar örnekler oluşturulmuş ve daha önce hesaplanan h[n] dürtü yanıtı ile conv komutu kullanılarak çıkış sinyali hesaplanmıştır.



```
title(sprintf('y[n] (a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('y[n]');
grid on;
xlim([plot_n_start plot_n_end]);
```

6) conv komutu kullanımı, xn 5. ile aynı, çıkış sinyali conv ile hesapla, stem ile çizdir

y = conv(x, h): x: input sinyali, h: impulse response, y: output sinyali

- conv komutunun döndürdüğü y vektörünün uzunluğu x ve h ın uzunluklarına bağlıdır (len(x)+len(h)-1)
- eğer x'in ilk indeksinin değeri nx ise ve h'ın ilk indeksinin değeri nh ise, y'nin ilk indeksinin değeri nx+nh olacak. bu conv sonucunun doğru n'lere denk getirmek için yani çizimde önemli

```
yn_conv = conv(xn, h_n);
nx_start = n_start_x;
nh_start = n_start;
ny_conv_start = nx_start + nh_start;
ny_conv_vals = ny_conv_start : (ny_conv_start + length(yn_conv) - 1);

plot_n_start = -20;
plot_n_end = 20;

idx_conv_plot_start = find(ny_conv_vals == plot_n_start);
idx_conv_plot_end = find(ny_conv_vals == plot_n_end);

figure;
stem(ny_conv_vals(idx_conv_plot_start:idx_conv_plot_end), ...
    yn_conv(idx_conv_plot_start:idx_conv_plot_end));
title(sprintf('y[n] (conv: a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('y[n]');
grid on;
xlim([plot_n_start plot_n_end]);
```

7. **FFT/IFFT Kullanarak Çıkış Sinyali Hesabı**

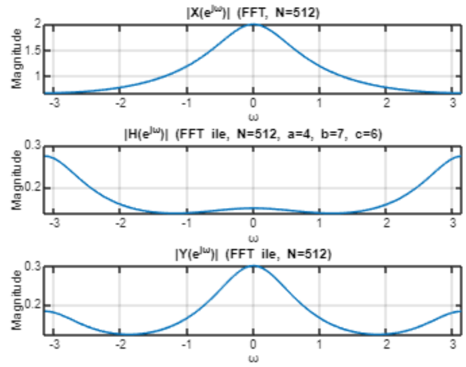
Zamanda evrişimin frekansta çarpmaya karşılık gelmesi özelliği kullanılarak çıkış sinyali y[n] hesaplanmıştır. x[n] giriş sinyali ve h[n] dürtü yanıtı için FFT alınmış, frekans uzayındaçarpım yapılmış (Y[k] = X[k] .* H[k]) ve ardından ifft ile y[n] bulunmuştur Frekans uzayında daha iyi bir örnekleme sağlanması açısından yine zero padding yapılmaya dikkat edilmiştir.

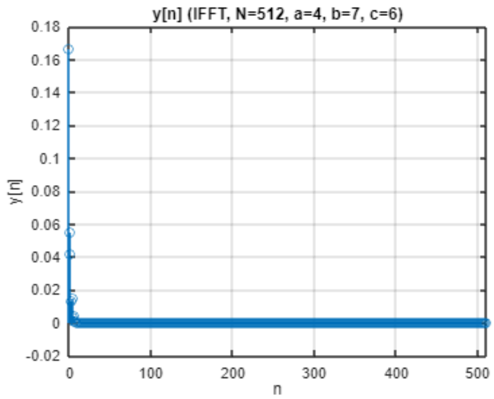
201 eleman n=512 -> zero padding

fftshift kullanımı: matlabın verdiği fft çıktı indexleri

x[n] = (1/2)^n * u[n], n<0 durumunda 0 ve hesaplamaya gerek yok

```
idx_n_ge_0_xn = find(n_vals_x >= 0);
xn_for_fft = xn(idx_n_ge_0_xn);
hn_for_fft = h_n(idx_0_h : end);
N_fft = 512;
Xk = fft(xn_for_fft, N_fft);
Hk = fft(hn_for_fft, N_fft);
w_fft_axis_shifted = (2*pi/N_fft) * ((-N_fft/2):(N_fft/2-1));

figure;
subplot(3,1,1);
plot(w_fft_axis_shifted, abs(fftshift(Xk)));
title(sprintf('|X(e^{jω})| (FFT, N=%d)', N_fft));
xlabel('ω');
ylabel('Magnitude');
grid on;
xlim([-pi pi]);

subplot(3,1,2);
plot(w_fft_axis_shifted, abs(fftshift(Hk)));
title(sprintf('|H(e^{jω})| (FFT ile, N=%d, a=%d, b=%d, c=%d)', N_fft, a, b, c));
xlabel('ω');
ylabel('Magnitude');
grid on;
xlim([-pi pi]);

Yk = Xk .* Hk;
subplot(3,1,3);
plot(w_fft_axis_shifted, abs(fftshift(Yk)));
title(sprintf('|Y(e^{jω})| (FFT ile, N=%d)', N_fft));
xlabel('ω');
ylabel('Magnitude');
grid on;
xlim([-pi pi]);

yn_ifft = ifft(Yk);
figure;
stem(0:(N_fft-1), real(yn_ifft));
title(sprintf('y[n] (IFFT, N=%d, a=%d, b=%d, c=%d)', N_fft, a, b, c));
xlabel('n');
ylabel('y[n]');
grid on;
xlim([0 N_fft-1]);
```



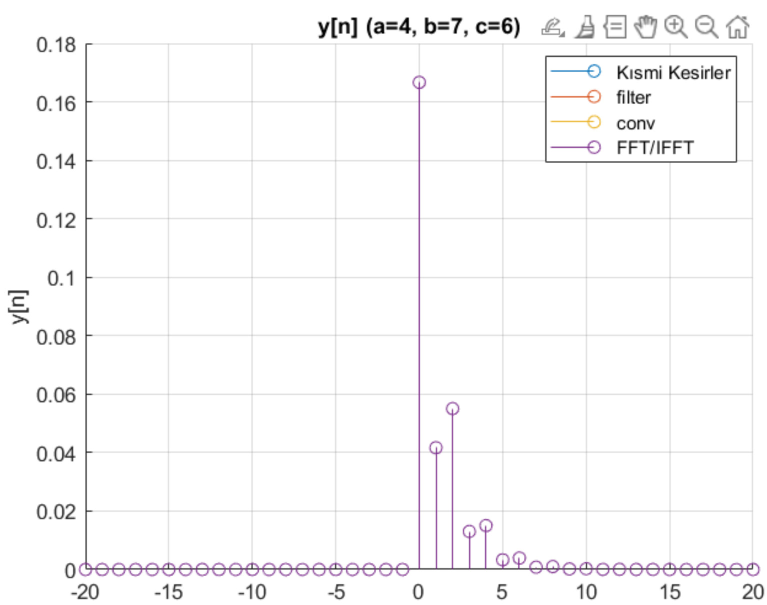


8. Çıkış Sinyallerinin Karşılaştırılması

Önceki dört maddede elde edilen y[n] çıkış sinyalleri aynı grafik üzerinde karşılaştırılmıştır. Bu karşılaştırma farklı hesaplama yöntemlerinin sonuçlarını görsel olarak incelemek ve yorumlamak için yapılmıştır.

8) Önceki dört maddedeki çıktıları Matlab'da aynı grafikte çizdirin. Sonuçları karşılaştırın.

```
compare_n_start = -20;
compare_n_end = 20;
compare_n_vals = compare_n_start:compare_n_end;

% 4. madde
yn_theoretical_plot = zeros(size(compare_n_vals));
idx_n0_compare = find(compare_n_vals == 0);
len_to_copy = min(length(y_n_theoretical), compare_n_end + 1);
yn_theoretical_plot(idx_n0_compare : (idx_n0_compare + len_to_copy - 1)) = ...
    real(y_n_theoretical(1 : len_to_copy));

% 5. madde - filter
idx_filter_plot_start = find(n_vals_x == compare_n_start);
idx_filter_plot_end = find(n_vals_x == compare_n_end);
yn_filter_plot = yn_filter(idx_filter_plot_start:idx_filter_plot_end);

% 6. madde - conv
idx_conv_plot_start = find(ny_conv_vals == compare_n_start);
idx_conv_plot_end = find(ny_conv_vals == compare_n_end);
yn_conv_plot = yn_conv(idx_conv_plot_start:idx_conv_plot_end);

% 7. madde - ifft
yn_ifft_plot = zeros(size(compare_n_vals));
ifft_source_start_idx = 1;
ifft_source_end_idx = min(length(yn_ifft), compare_n_end + 1);
target_start_idx_ifft = find(compare_n_vals == 0);
ifft_data_to_copy = real(yn_ifft(ifft_source_start_idx : ifft_source_end_idx));
yn_ifft_plot(target_start_idx_ifft : (target_start_idx_ifft + length(ifft_data_to_copy) - 1))

figure;
hold on;
stem(compare_n_vals, yn_theoretical_plot, 'DisplayName', 'Kısmi Kesirler');
stem(compare_n_vals, yn_filter_plot, 'DisplayName', 'filter');
stem(compare_n_vals, yn_conv_plot, 'DisplayName', 'conv');
stem(compare_n_vals, yn_ifft_plot, 'DisplayName', 'FFT/IFFT');

title(sprintf('y[n] (a=%d, b=%d, c=%d)', a, b, c));
xlabel('n');
ylabel('y[n]');
legend('Location', 'best');
grid on;
xlim([compare_n_start compare_n_end]);
hold off;
```



8.2. Sonuçların Karşılaştırılması ve Yorum:
Dört farklı yöntemin verdiği y[n] sonuçları görsel olarak incelendiğinde birbirine benzer oldukları görülmektedir. Her yöntemin kullanılan duruma göre sağlayabileceği avantajları ve dezavantajları vardır.

- **Teorik (Kısmi Kesirler):** y[n]'nin tam analitik ifadesini sunar, bu yüzden en hassas ve kesin sonucu verir, ancak karmaşık sistemler için computationally intensive olma ihtimali vardır. Floating point precision hataları burada en az etkiye sahip olacaktır.
- **filter:** Doğrudan fark denklemini uygular ve DT sistemler için en doğru ve kararlı sayısal çözümlerden biridir. Nedensellik varsayımı doğal olarak ele alınır
- **conv:** Giriş sinyali ile dürtü yanıtının evrişimini hesaplar. Sinyal uzunluklarının doğru yönetimi önemlidir, aksi durumda sinyal kırpılabilir veya zaman yanlış kayabilir.
- **FFT/IFFT:** Zamanda evrişimin frekansta çarpmaya dönüşme özelliğini kullanır. Bu yöntem büyük sinyaller için daha hızlı olabilir ancak doğru N_fft boyutunda zero padding yapılması gerekir (circular convolution olmaması için) fftshift kullanımı spektrumun görselleştirilmesi içindir ve ifft için zorunlu değildir.
- **Farklılıklar ve Sınırlamalar:**
  - **Nümerik Hassasiyet:** Tüm sayısal yöntemler MATLAB sayesinde float hatalarına açıktır ancak bu proje için etkileri göz ardı edilebilir düzeydedir.
  - **Zero Padding (FFT/IFFT için):** FFT/IFFT'de kullanılan zero padding miktarı ve sinyallerin zaman aralığı frekans çözünürlüğünü ve sonucun doğruluğunu etkiler.
  - **conv Komutunun Kenar Etkileri:** conv komutunun tanımı gereği giriş ve dürtü yanıtı sinyallerinin başlangıç ve bitiş noktalarına göre çıkış sinyalinin indeksi kayar ve sinyal uzunluğu artar ve indeksleme bu göz önüne alınarak yapılmazsa grafikte görsel farklılıklar oluşabilir.
  - **Teorik Modelin Varsayımları:** Teorik çözüm sistemin nedensel olduğu ve başlangıç koşullarının sıfır olduğu varsayımına dayanır. Başka uygulamalarda bu varsayımlardan sapmalar sonuçları etkileyebilir.