Contents

1 Signals

1.1 Elementary signals

1.1.1 CT signals

```
clearvars;
t=-10:0.01:10;
x1=(t==0);

subplot(2,3,1); plot(t,x1);
xlabel('t'); ylabel('x1(t)');
title('unit impulse');

u=(t>=0);

subplot(2,3,2); plot(t,u);
xlabel('t'); ylabel('u(t)');
title('step impulse');

r=t.*u;

subplot(2,3,3); plot(t,r);
xlabel('t'); ylabel('r(t)');
title('unit ramp function');

p=0.5*(t.^2).*u;

subplot(2,3,4); plot(t,p);
xlabel('t'); ylabel('p(t)');
title('parabolic function');

x2=exp(t*0.5);

subplot(2,3,5); plot(t,x2);
xlabel('t'); ylabel('x2(t)');
title('exponential function');

f1=1;
x3=sin(2*pi*f1*t);

subplot(2,3,6); plot(t, x3);
axis([-1 1 -2 2]); xlabel('t'); ylabel('x3(t)');
title('sine wave');
```

1.1.2 DT signals

```
clear all;
n = -10:10;
x = n == 0;
subplot(2, 3, 1); stem(n, x);
axis([-10 10 -2 2]); xlabel('n'); ylabel('x(n)');
title('unit impulse')

u = n >= 0;
subplot(2, 3, 2); stem(n, u);
axis([-10 10 -2 2]); xlabel('n'); ylabel('u(n)');
title('unit step');

r = u .* n;
subplot(2, 3, 3); stem(n, r);
axis([-10 10 -2 20]); xlabel('n'); ylabel('r(n)');
title('unit ramp')

p = 0.5 * (n .^ 2) .* u;
subplot(2, 3, 4); stem(n, p);
xlabel('n'); ylabel('p(n)');
title('parabolic')

x1 = 2 .^ n;
subplot(2, 3, 5); stem(n, x1);
xlabel('n'); ylabel('x1(n)');
title('exponential');

f1 = .05;
x3 = sin(2 * pi * f1 * n);
subplot(2, 3, 6); stem(n, x3);
xlabel('n'); ylabel('x3(n)');
title('sine');
```

1.2 Generation of Signals

```
clearvars; close all;
n = -5:5;
i1 = (n == -2);
i2 = (n == 4);

x1 = 2*i1 - i2;

n1 = 0:20;
u1 = (n1 >= 0);
```

```
u2 = (n1 >= 10);
u3 = (n1 >= 20);
x2 = n1 .* (u1 - u2) + 10*exp(-0.3 * (n1 - 10)) .* (u2 - u3);

n2 = 0:50;
w = randn(size(n2));
x3 = cos(0.04 * pi * n2) + 0.2 * w;

tiledlayout(2,2);
nexttile; stem(n, x1);
xlabel('n'); ylabel('x1(n)');
title('signal A');

nexttile; stem(n1, x2);
xlabel('n'); ylabel('x2(n)');
title('signal B');

nexttile; stem(n2, x3);
xlabel('n'); ylabel('x3(n)');
title('signal C');

n4 = -10:9;
x = 5:-1:1;
x4 = x;

for i=1:3
   x4 = [x4, x];
end
nexttile; stem(n4, x4);
xlabel('n'); ylabel('x4(n)');
title('signal D');
```

1.3 Operations on Signals

1.3.1 Delay & Advance

```
clearvars;
x = input('Enter sequence: ');
d = 2; %input('delay: ');
a = 1; %input('advance: ');
n = 0:4;
n1 = n + d;
n2 = n - a;

subplot(3, 3, 1); stem(n, x);
axis([-7 7 -1 6]); xlabel('n'); ylabel('x(n)');
```

```
title('input');

subplot(3, 3, 4); stem(n1, x);
axis([-7 7 -1 6]); xlabel('n'); ylabel('x(n+k)');
title('delayed');

subplot(3, 3, 7); stem(n2, x);
axis([-7 7 -1 6]); xlabel('n'); ylabel('x(n-k)');
title('advanced');

n1 = 0:4;
y = fliplr(x);
n2 = -4:0;
n = -4:4;
y1 = zeros(1, 9);
y2 = y1;

y1(find((n >= min(n1)) & (n <= max(n1)) == 1)) = x;
y2(find((n >= min(n2)) & (n <= max(n2)) == 1)) = y;

subplot(3, 3, 2); stem(n, y2);
axis([-7 7 -1 6]); xlabel('n'); ylabel('x(-n)');
title('reversal sequence');

z1 = (y1 + y2) / 2;
subplot(3, 3, 5); stem(n, z1);
axis([-7 7 -1 6]); xlabel('n'); ylabel('z1(n)');
title('even');

z2 = (y1 - y2) / 2;
subplot(3, 3, 6); stem(n, z2);
axis([-7 7 -4 4]); xlabel('n'); ylabel('z2(n)');
title('odd');
```

1.3.2 Add, Subtract, Multiply

```
clearvars;
x = [2 4 6 8];
n1 = -2:1;
n2 = 0:3;
y = [1 3 5 7];
n = -2:3;
y1 = zeros(1, 6);
y2 = y1;
y1(find((n >= min(n1)) & (n <= max(n1)) == 1)) = x;
y2(find((n >= min(n2)) & (n <= max(n2)) == 1)) = y;
```

```
subplot(3, 2, 1); stem(n1, x);
xlabel('n'); ylabel('x(n)');
title('input seq 1');

subplot(3, 2, 2); stem(n2, y);
xlabel('n'); ylabel('y(n)');
title('input seq 2');

z1 = y1+y2;
z2 = y1-y2;
z3 = y1.*y2;

subplot(3, 2, 3); stem(n, z1);
xlabel('n'); ylabel('y(n)');
title('signal addition');

subplot(3, 2, 4); stem(n, z2);
xlabel('n'); ylabel('y(n)');
title('signal subtraction');

subplot(3, 2, 5); stem(n, z3);
xlabel('n'); ylabel('y(n)');
title('signal multiplication');
```

2 Systems

2.1 Implementation of systems

```
clc; clear all; close all;
n  = -5:15;
a  = [1, -1, 1/4]     ; b  = [1, 1/4, -1/8];
x1 = (n == 0)         ; x2 = (n >= 0);
h  = filter(b, a, x1); s  = filter(b, a, x2);

subplot(2, 1, 1) ; stem (n, h);
xlabel('n')      ; ylabel('y(n)');
title('impulse response');

subplot(2, 1, 2) ; stem(n, s);
xlabel('n')      ; ylabel('s(n)');
title('Step Response');

figure; zplane(b, a);
figure; freqz(b, a);

m = filt(b, a);
disp('zeros')   ; zero(m);
disp('poles')   ; pole(m);
disp('residues'); residuez(b, a);

if(max(abs(pole(m))) <= 1)
    disp('All poles lie inside unit circle, system is stable');
else
    disp('Poles lie outside unit circle, system is unstable');
end

if (h(n < 0) == 0); disp("Causal");
else; disp("non-causal");
end
```

2.2 Properties of System

2.2.1 Linearity

```
clc; clear all; close all;

alpha = 4; beta = 4;
% system A
```

```
a  = [2 3]           ; b = [1 2];
x1 = rand(1, 10)     ; x2 = rand(1, 10);
y1 = filter(b, a, x1); y2 = filter(b, a, x2);
x  = alpha * x1 + beta * x2;
y3 = alpha * y1 + beta * y2;
y  = filter(b, a, x);

if (abs(y3 - y) < 0.0001) disp('system A is linear')
else disp('system A is not linear')
end

subplot(2, 2, 1); stem(y);
xlabel('n')     ; ylabel('y(n)');
title('output due to sum of inputs - A');

subplot(2, 2, 2); stem(y3);
xlabel('n')     ; ylabel('y3(n)');
title('sum of outputs - A');

% system B
x1 = rand(1, 10); x2 = rand(1, 10);
y1 = 2 * x1 + 4 ; y2 = 2 * x2 + 4;
x  = alpha * x1 + beta * x2;
y3 = alpha * y1 + beta * y2;
y  = 2 * x + 4;

if (abs(y3 - y) < 0.0001) disp('system B is linear')
else disp('system B is not linear')
end

subplot(2, 2, 3); stem(y);
xlabel('n')     ; ylabel('y(n)') ; ylim([0 50]);
title('output due to sum of inputs - B');

subplot(2, 2, 4); stem(y3);
xlabel('n')     ; ylabel('y3(n)'); ylim([0 50]);
title('sum of outputs - B');
```

2.2.2 Time Invariance

```
clc; clear all; close all;
k = 5;
% system A
a = [1 2 3]    ; b = [2 3];
x = rand(1, 10); y = filter(b, a, x);
```

```
xShifted   = [zeros(1, k), x]; oShifted = [zeros(1, k), y];
owxShifted = filter(b, a, xShifted); %o/p when x shifted

subplot(2, 2, 1); stem(oShifted);
xlabel('n');      ylabel('rhs');
title('A: shifted output');

subplot(2, 2, 2); stem(owxShifted);
xlabel('n');      ylabel('lhs');
title('A: output due to shifted input');
if (abs(owxShifted - oShifted) < 0.0001)
    disp('system A is time invariant');
else
    disp('system A is time variant');
end

% system B
n1 = 1:(10 + k); n2 = 1:10;
x  = rand(1, 10); y = 2 .* x .* n2;

oShifted   = [zeros(1, k), y];
xShifted   = [zeros(1, k), x];
owxShifted = 2 .* n1 .* xShifted;

subplot(2, 2, 3); stem(oShifted);
xlabel('n');      ylabel('rhs');
title('B: shifted output'); ylim([0 25]);

subplot(2, 2, 4); stem(owxShifted);
xlabel('n');      ylabel('x2');
title('B: output due to shifted input');

if (abs(oShifted - owxShifted) < 0.0001) disp('system B is time invariant');
else disp('system B is time variant');
end
```

3 Fourier Transforms

3.1 Discrete Fourier Transform - DFT

```
clc; clear all; close all;
x = [1; 2; 3; 4]; N = 4;
n = 0:(N - 1)   ; k = 0:(N - 1);
W = exp(-1i * 2 * pi / N) .^ (n' * k);
X = W * x; X2 = fft(x);

disp('Using DFT matrix:'); disp(X)
disp('Using built-in function - fft:'); disp(X2)

subplot(2, 2, 1); stem(k, abs(X));
xlabel('k')     ; ylabel('|X(k)|');
title('magnitude response');

subplot(2, 2, 2); stem(k, angle(X));
xlabel('k')     ; ylabel('angle X(k) in rad');
title('phase response'); ylim([-4 4]);

subplot(2, 2, 3); stem(k, abs(X2));
xlabel('k')     ; ylabel('|X(k)|');
title('fft - magnitude response');

subplot(2, 2, 4); stem(k, angle(X2));
xlabel('k')     ; ylabel('angle X(k) in rad');
title('fft - phase response'); ylim([-4 4]);
```

3.2 Discrete Time Fourier Transform - DTFT

```
clc; clear all; close all;
x = [1, 2, 3, 4, 5];
n = 0:4; M = 500;
k = 0:M; w = (pi / M) * k;
W = exp(-1j * pi / M) .^ (n'*k);
X = x * W; s = w / pi;

figure;
subplot(2, 2, 1);
plot(s, real(X)); xlabel('w (normalised)');
ylabel('re X(e^{jw})'); title('real response');

subplot(2, 2, 2);
```

```
plot(s, imag(X)); xlabel('w (normalised)');
ylabel('im X(e^{jw})'); title('imaginary response');

figure;
subplot(2, 1, 1); plot(s, 10 * log(abs(X)));
xlabel('Normalised frequency (\times \pi rad/sample)');
ylabel('|X(e^{jw})| in dB'); ylim([7 28]);
title('magnitude response'); xlim([0 1]); grid;

subplot(2, 1, 2); plot(s, unwrap(angle(X))*180/pi);
xlabel('Normalised frequency (\times \pi rad/sample)');
ylabel('\angle X(e^{jw}) in deg'); ylim([-750, 30])
title('phase response'); grid;

figure; freqz(x);
```

3.3 Properties of DTFT

3.3.1 Periodicity

```
clc; clear all; close all;
n = 0:10; M = 100;
x = (0.9 * exp(1j * pi / 3)).^n;
k = -2*M:2*M; w = (pi / M) * k;
W = exp(-1j * pi/M).^(n'*k);
X = x * W;
figure

subplot(2, 1, 1); plot(w / pi, abs(X));
xlabel('w (normalised)'); ylabel('|X(e^{jw})|');
title('magnitude response'); xlim([-2 2]);

subplot(2, 1, 2); plot(w / pi, angle(X));
xlabel('w (normalised)'); ylabel('\angle X(e^{jw}) in rad');
title('phase response'); xlim([-2 2]);
```

3.3.2 Conjugate Symmetry

```
clc; clearvars; close all;
n = -10:10   ; M = 100;
x = (-0.9).^n; k = -2*M:2*M;
w = (pi / M) * k;
W = exp(-1j * pi/M).^(n'*k);
X = x * W;

subplot(2, 1, 1); plot(w / pi, abs(X));
```

```
xlabel('w (normalised)'); ylabel('|X(e^{jw})|');
title('magnitude response'); xlim([-2 2]);

subplot(2, 1, 2); plot(w / pi, unwrap(angle(X)));
xlabel('w (normalised)'); ylabel('\angle X(e^{jw}) in rad');
title('phase response'); xlim([-2 2]);
```

4 Convolution

4.1 Linear Convolution

```
clearvars;
nxb = -3; nxe = 3;
nx = nxb:nxe;
x = [3, 11, 7, 0, -1, 4, 2];
nhb = -1; nhe = 4;
nh = nhb:nhe;

h = [2, 3, 0, -5, 2, 1];
ny = (nxb + nhb):(nxe + nhe);
y = conv(x,h);
l1 = length(x);
l2 = length(h);
l3 = l1 + l2 - 1; %resultant length

tiledlayout(4, 1);
stem_(nx, x, "x", "Input Sequence");
stem_(nh, h, "h", "Impulse Response");
stem_(ny, y, "y", "linear convolution output");

disp('using built in function - conv')
disp(y)

%frequency domain approach
x = [x,zeros(1, l2 - 1)];
h = [h,zeros(1, l1 - 1)];
Y = fft(x) .* fft(h);
y2 = ifft(Y);
disp('Frequency domain method')

disp(y2)
stem_(ny, y2, "y2", 'frequency domain method')

z1= zeros(1, l3);
for i=1:l3
    for j=1:i
        z1(i) = z1(i) + x(j) * h(i-j+1);
    end
end
disp("using loop: ");
disp(z1);
```

```
function stem_(x,y,yl,tl)
    nexttile;
    stem(x, y);
    xlabel('n');
    ylabel(yl+'(n)');
    title(tl);
end
```

4.2 Circular Convolution

```
clearvars;
x1 = [1, 2, 2];
x2 = [1 2 3 4];
N1 = length(x1);
N2 = length(x2);
N = max(N1,N2);
y = cconv(x1,x2,N);

n1=0:N1-1;
n2=0:N2-1;
n=0:N-1;
tiledlayout(2,2);
stem_(n1,x1,"x1","input sequence 1");
stem_(n2,x2,"x2","input sequence 2");
stem_(n,y,"y","Circular convolution");

% without using built in function
x1=[x1 zeros(1, N-N1)];
x2=[x2 zeros(1, N-N2)];
y1 = zeros(1, N);
for n=1:N
    for m= 1 : N
        i=mod(n-m , N);
        i=i+1;
        y1(n) = y1(n) + x1(m) * x2(i);
    end
end
disp(y1)

function stem_(x,y,yl,tl)
    nexttile;
    stem(x, y);
    xlabel('n');
    ylabel(yl+'(n)');
```

```
    title(tl);
end
```

4.3 Linear and Circular Convolution Equivalence

```
clearvars;

x1=[1 2 2 1];
x2=[1 -1 -1 1];
N1=length(x1);
N2=length(x2);
N=N1+N2-1;
y1 = conv(x1,x2);
y = cconv(x1,x2,N);
n1=0:N1-1;
n2=0:N2-1;
n = 0:N-1;
tiledlayout(2,2);
stem_(n1,x1,"x1","input sequence 1");
stem_(n2,x2,"x2","input sequence 2");
stem_(n,y1,"y1","linear convolution");
stem_(n,y,"y","circular convolution");

function stem_(x,y,yl,tl)
    nexttile;
    stem(x, y);
    xlabel('n');
    ylabel(yl+'(n)');
    title(tl);
end
```

5 Sampling Theorem

```
clearvars;

fm = 1;
f = @(x) cos(2*pi*fm * x);
n=-2; m=2;
ctTicks = n:0.01:m;

fs1 = fm;   % fs < 2fm
fs2 = 2*fm; % fs = 2fm
fs3 = 10*fm; % fs > 2fm

dtTicks1 = n:1/fs1:m;
dtTicks2 = n:1/fs2:m;
dtTicks3 = n:1/fs3:m;
tiledlayout(2, 2);

nexttile;
plot(ctTicks, f(ctTicks));
title("CT signal");
xlabel("t"); ylabel("x(t)");

plotFx(f(dtTicks1), dtTicks1*fs1, "f_s < 2f_m");
plotFx(f(dtTicks2), dtTicks2*fs2, "f_s = 2f_m");
plotFx(f(dtTicks3), dtTicks3*fs3, "f_s > 2f_m");

function plotFx(fx, T, name)
    nexttile;
    stem(T, fx);
    hold on;
    plot(T, fx, ":")
    title("DT signal x(n) with " + name);
    xlabel("n");
    xlim([min(T), max(T)]);
    ylabel("x(n)");
    hold off;
end
```

6 FIR Filter Design

6.1 Window Functions

```
clearvars;

M = 31; % window size
n = 0:M - 1;
W_rect = n >= 0;
W_hann = 0.5 - 0.5 * cos(2*pi*n/(M-1));
W_hamm = 0.54 - 0.46 * cos(2*pi*n/(M-1));
W_barl = 1 - abs(M-1 - 2*n) / (M-1);
W_bkmn = 0.42 - .5 * cos(2*pi*n/(M-1)) + 0.08 * cos(4*pi*n/(M-1));

% builtin
Wb_rect = rectwin(M);
Wb_hann = hann(M);
Wb_hamm = hamming(M);
Wb_barl = bartlett(M);
Wb_bkmn = blackman(M);

tiledlayout(2, 5);
plotF(W_rect, "Rectangular");
plotF(W_hann, "Hanning");
plotF(W_hamm, "Hamming");
plotF(W_barl, "Bartlett");
plotF(W_bkmn, "Blackman");

plotF(Wb_rect, "Rectangular");
plotF(Wb_hann, "Hanning");
plotF(Wb_hamm, "Hamming");
plotF(Wb_barl, "Bartlett");
plotF(Wb_bkmn, "Blackman");

function plotF(fx, title_)
    nexttile;
    stem(fx);
    ylabel("W(n)");
    xlabel("n");
    title(title_);
end
```

6.2 Low-pass Filter

```
clearvars;
close all;
wp = 0.2*pi;
ws = 0.3*pi;
fc = (wp + ws) / 2/pi;
TW = ws - wp;

M = ceil(8*pi/TW);
n = 0:M-1;
m = n - ceil((M-1)/2);
hd = fc*sinc(fc*m);
w = hamming(M)';
h = hd.*w;

tiledlayout(3, 1);
plotF(n, hd, "hd", "ideal response");
plotF(n, w, "w", "hamming window");
plotF(n, h, "h", "finite impulse response");

figure;
freqz(h);

function plotF(x, y, yl, nam)
    nexttile; stem(x,y);
    xlabel('n'); ylabel(yl+'(n)');
    title(nam);
end
```

6.3 Band-pass Filter

```
clearvars; close all;

wls = 0.2*pi;
wlp = 0.35*pi;
wup = 0.65*pi;
wus = 0.8*pi;

fcl = (wls + wlp) / pi / 2;
fcu = (wus + wup) / pi / 2;
TW = wus - wup;

M = ceil(11*pi/TW);
n = 0:M-1;
m = n - ceil((M-1)/2);
```

```
hd = fcu * sinc(fcu*m) - fcl * sinc(fcl*m);
w = blackman(M)';

h = hd .* w;

tiledlayout(3, 1);
plotF(n, hd, "hd", "ideal impulse response");
plotF(n, w, "w", "blackman window");
plotF(n, h, "h", "Finite Impulse Response")

figure
freqz(h)

function plotF(x, y, yl, nam)
    nexttile; stem(x,y);
    xlabel('n'); ylabel(yl+'(n)');
    title(nam);
end
```

6.4 High-pass Filter

```
clearvars; close all;

wp = 0.4*pi;
ws = 0.6*pi;
fc = (wp + ws) / 2 / pi;

TW = ws - wp;

M = ceil(6.1*pi/TW);
n = 0:M-1;
m = n - ceil((M-1)/2);
hd = sinc(m) - fc * sinc(fc*m);
w = bartlett(M)';

h = hd .* w;

tiledlayout(3, 1);
plotF(n, hd, "hd", "ideal impulse response");
plotF(n, w, "w", "Bartlett window");
plotF(n, h, "h", "Finite Impulse Response")

figure
freqz(h)
```

```
function plotF(x, y, yl, nam)
    nexttile; stem(x,y);
    xlabel('n'); ylabel(yl+'(n)');
    title(nam);
end
```

6.5 Band-stop Filter

```
clearvars; close all;

wcl = pi/3;
wcu = 2*pi/3;
fcl = wcl / pi;
fcu = wcu / pi;

M = 45;
n = 0:M-1;
m = n - ceil((M-1)/2);
hd = sinc(m) - (fcu * sinc(fcu*m) - fcl * sinc(fcl*m));
w = hann(M)';

h = hd .* w;

tiledlayout(3, 1);
plotF(n, hd, "hd", "ideal impulse response");
plotF(n, w, "w", "Hann window");
plotF(n, h, "h", "Finite Impulse Response")

figure;
freqz(h)

function plotF(x, y, yl, nam)
    nexttile; stem(x,y);
    xlabel('n'); ylabel(yl+'(n)');
    title(nam);
end
```

7 Correlation And Spectral Analysis

7.1 Cross-correlation

```
clearvars; close all;
x=[1 2 3 4];
h=[1 4 2 1];
y=fliplr(xcorr(x, h));
tiledlayout(3, 1);
drawF(x, "Input sequence 1")
drawF(h, "Input sequence 2");
drawF(y, "Output sequence");
disp('Resultant: ');
disp(y);

function drawF(f, title_)
    nexttile;
    stem(f);
    xlabel('n'); ylabel('Amplitude');
    title(title_);
end
```

7.2 Auto-correlation

```
clearvars; close all;
x=[1 2 1 5];
y=fliplr(xcorr(x, x));
tiledlayout(3, 1);
drawF(x, "Input sequence")
drawF(y, "Output sequence");
disp('The resultant is');
disp(y);

function drawF(f, title_)
    nexttile;
    stem(f);
    xlabel('n');
    ylabel('Amplitude');
    title(title_);
end
```