

```

``matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WEEK 8 – DISCRETE-TIME SIGNALS (EXERCISES) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Common helpers
u = @(n) double(n>=0); % unit step
delta = @(n) double(n==0); % unit impulse

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 1
% Plot signals (fig a–e) and generate time-scaling / shifting versions.
% --- You must DEFINE each original signal x_a, x_b, ... based on your figs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Example template for ONE signal (repeat for each fig a–e by changing defs)

% ---- FIG (a) ----
n_a = -10:10; % choose suitable range for fig (a)
x_a = ...; % <== FILL from your fig (a) definition

% size (length) of signal
len_xa = length(x_a);

% required transforms:
x_a_rev = fliplr(x_a); % x(-n)
x_a_nplus6 = x_a; % x(n+6) : handle by shifting n -> (n-6)
n_a_nplus6 = n_a - 6;
x_a_nminus6 = x_a; % x(n-6) : n -> (n+6)
n_a_nminus6 = n_a + 6;
% x(3n) and x(n/2) – do by re-sampling indices
n_big = -30:30;
x_a_3n = arrayfun(@(k) x_a(n_a==3*k), n_big, 'UniformOutput', true);
x_a_half = zeros(size(n_big));
for k = 1:length(n_big)
    if mod(n_big(k),2)==0
        idx = (n_big(k)/2);
        x_a_half(k) = x_a(n_a==idx);
    end
end

x_a_affine = -3 * ... + 5; % -3x(-0.2n+12)+5 (create n' = -0.2n+12, map)

% x(3-n): again via index mapping
x_a_3minusn = zeros(size(n_big));
for k = 1:length(n_big)
    arg = 3 - n_big(k);
    if any(n_a==arg)
        x_a_3minusn(k) = x_a(n_a==arg);
    end
end

% plotting (example)
figure;
subplot(3,3,1); stem(n_a,x_a,'filled'); title('x_a[n]');
subplot(3,3,2); stem(fliplr(-n_a),x_a_rev); title('x_a[-n]');
subplot(3,3,3); stem(n_a_nplus6,x_a_nplus6); title('x_a[n+6]');

```

```

subplot(3,3,4); stem(n_a_nminus6,x_a_nminus6);title('x_a[n-6]');
subplot(3,3,5); stem(n_big,x_a_3n); title('x_a[3n]');
subplot(3,3,6); stem(n_big,x_a_half); title('x_a[n/2]');
subplot(3,3,7); stem(n_big,x_a_affine); title('x_a[-3x(-0.2n+12)+5]');
subplot(3,3,8); stem(n_big,x_a_3minusn); title('x_a[3-n]');

```

% Repeat same style for fig (b), (c), (d), (e): change n_b, x_b, etc.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 2
% Signals: (1)^n, (-1)^n, u[n], (-1)^n u[n], cos(pi*n/3 + pi/6)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
n = -20:20;
```

```

x1 = (1).^n; % (1)^n
x2 = (-1).^n; % (-1)^n
x3 = u(n); % u[n]
x4 = (-1).^n .* u(n); % (-1)^n u[n]
x5 = cos(pi*n/3 + pi/6); % cos(pi*n/3 + pi/6)

```

```

figure;
subplot(5,1,1); stem(n,x1,'filled'); title('(1)^n');
subplot(5,1,2); stem(n,x2,'filled'); title('(-1)^n');
subplot(5,1,3); stem(n,x3,'filled'); title('u[n]');
subplot(5,1,4); stem(n,x4,'filled'); title('(-1)^n u[n]');
subplot(5,1,5); stem(n,x5,'filled'); title('cos(pi*n/3 + pi/6)');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 3
% a) (1/2) * sin( (pi/2) n ) [interpreted typical lab signal]
% b) n (n-1) v^n u[n-2] for v = 0.8 and -0.8
% c) (u[n] + (-1)^(n+1) u[n]) * cos(pi n / 2)
% d) n {u[n] - u[n-7]}
% e) (-n+8){u[n-6] - u[n-9]}
% f) (n-2){u[n-2] - u[n-6]} + (-n+8){u[n-6] - u[n-9]}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
n = -10:20;
```

```

% (a)
xa = 0.5 * sin((pi/2)*n);
% even / odd
xa_even = 0.5*(xa + fliplr(xa));
xa_odd = 0.5*(xa - fliplr(xa));

```

```

figure;
subplot(3,1,1); stem(n,xa,'filled'); title('x_a[n]');
subplot(3,1,2); stem(n,xa_even,'filled'); title('Even{x_a[n]}');
subplot(3,1,3); stem(n,xa_odd,'filled'); title('Odd{x_a[n]}');

```

```

% (b) v = 0.8 and v = -0.8
v1 = 0.8; v2 = -0.8;
xb1 = n.*(n-1).*(v1.^n).*u(n-2);
xb2 = n.*(n-1).*(v2.^n).*u(n-2);

```

```
figure;
subplot(2,1,1); stem(n,xb1,'filled'); title('x_b[n], v=0.8');
subplot(2,1,2); stem(n,xb2,'filled'); title('x_b[n], v=-0.8');
```

```
% Even/odd for x_b with v=0.8 (example)
xb1_even = 0.5*(xb1 + fliplr(xb1));
xb1_odd = 0.5*(xb1 - fliplr(xb1));
```

```
figure;
subplot(3,1,1); stem(n,xb1,'filled'); title('x_b[n], v=0.8');
subplot(3,1,2); stem(n,xb1_even,'filled'); title('Even{x_b}');
subplot(3,1,3); stem(n,xb1_odd,'filled'); title('Odd{x_b}');
```

```
% (c)
xc = (u(n) + (-1).^(n+1).*u(n)) .* cos(pi*n/2);
xc_even = 0.5*(xc + fliplr(xc));
xc_odd = 0.5*(xc - fliplr(xc));
```

```
figure;
subplot(3,1,1); stem(n,xc,'filled'); title('x_c[n]');
subplot(3,1,2); stem(n,xc_even,'filled'); title('Even{x_c}');
subplot(3,1,3); stem(n,xc_odd,'filled'); title('Odd{x_c}');
```

```
% (d)
xd = n.*(u(n) - u(n-7));
xd_even = 0.5*(xd + fliplr(xd));
xd_odd = 0.5*(xd - fliplr(xd));
```

```
figure;
subplot(3,1,1); stem(n,xd,'filled'); title('x_d[n]');
subplot(3,1,2); stem(n,xd_even,'filled'); title('Even{x_d}');
subplot(3,1,3); stem(n,xd_odd,'filled'); title('Odd{x_d}');
```

```
% (e)
xe = (-n+8).*(u(n-6) - u(n-9));
xe_even = 0.5*(xe + fliplr(xe));
xe_odd = 0.5*(xe - fliplr(xe));
```

```
figure;
subplot(3,1,1); stem(n,xe,'filled'); title('x_e[n]');
subplot(3,1,2); stem(n,xe_even,'filled'); title('Even{x_e}');
subplot(3,1,3); stem(n,xe_odd,'filled'); title('Odd{x_e}');
```

```
% (f)
xf = (n-2).*(u(n-2) - u(n-6)) + (-n+8).*(u(n-6) - u(n-9));
xf_even = 0.5*(xf + fliplr(xf));
xf_odd = 0.5*(xf - fliplr(xf));
```

```
figure;
subplot(3,1,1); stem(n,xf,'filled'); title('x_f[n]');
subplot(3,1,2); stem(n,xf_even,'filled'); title('Even{x_f}');
subplot(3,1,3); stem(n,xf_odd,'filled'); title('Odd{x_f}');
```

```
%%%%%%%%%%
%%%%%%%%%%
```

```
%% EXERCISE 4
```

```
%  $y[n] = -(1/2)x[-3n + 2]$ 
```

```
% General template: given  $y[n]$  samples, recover  $x[n]$ , sketch even/odd, etc.
```

```
%%%%%%%%%%
%%%%%%%%%%
```

[illegible]

```

%% Helper: iterative solution for 2nd-order difference eq
%  $y[n+2] + a_1 y[n+1] + a_0 y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$ 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [n,y] = solve_diff2_iter(a1,a0,b0,b1,b2,x_fun,IC,Nmax)
% IC: struct with fields n1,n2,y1,y2 (y[n1], y[n2]); assume n1=-2, n2=-1
n = IC.n1:Nmax;          % e.g., -2:50
L = length(n);
y = zeros(L,1);
y(1) = IC.y1;            % y[n1]
y(2) = IC.y2;            % y[n2]
x = x_fun(n).';          % column vector

```

```

for k = 1:L-2             %  $k \leftrightarrow n(k)$ 
    % indices:  $n(k) \rightarrow x[n-2]$ 
    %           $n(k+1) \rightarrow x[n-1]$ 
    %           $n(k+2) \rightarrow x[n]$ 
    y(k+2) = -a1*y(k+1) - a0*y(k) ...
              + b0*x(k+2) + b1*x(k+1) + b2*x(k);
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 1 – IMPULSE RESPONSE & ZERO-INPUT RESPONSE
% ICs:  $y[-1] = -1$ ;  $y[-2] = 2$  (use these two for 2nd-order systems)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%% 1)  $y[n+2] - 1.9 y[n+1] + 0.9 y[n] = 2x[n] - 3x[n-1] + x[n-2]$ 
% =>  $a = [1 \ -1.9 \ 0.9]$ ,  $b = [2 \ -3 \ 1]$ 

```

```

a1 = -1.9; a0 = 0.9;
b0 = 2;   b1 = -3; b2 = 1;

```

```

% ---- Impulse response (zero-state) via iterative method
IC_zero.y1 = 0; IC_zero.y2 = 0; IC_zero.n1 = -2; %  $y[-2]=0$ ,  $y[-1]=0$ 
x_imp_fun = @(n) delta(n);          %  $\delta[n]$ 

```

```

[n_imp,y_imp] = solve_diff2_iter(a1,a0,b0,b1,b2,x_imp_fun,IC_zero,50);

```

```

figure;
stem(n_imp,y_imp,'filled'); xlabel('n'); ylabel('h[n]');
title('Impulse Response (Iterative) – System 1');

```

```

% ---- Impulse response via filter (zero-state)
b = [b0 b1 b2];
a = [1 a1 a0];

```

```

n0 = 0:50;
h_filter = filter(b,a,delta(n0));

```

```

figure;
stem(n0,h_filter,'filled'); xlabel('n'); ylabel('h[n]');
title('Impulse Response (filter) – System 1');

```

```

% ---- Zero-input response (ICs non-zero,  $x[n] = 0$ )
IC_ZIR.y1 = 2;          %  $y[-2] = 2$ 

```

```

IC_ZIR.y2 = -1;    % y[-1] = -1
IC_ZIR.n1 = -2;
x_zero_fun = @(n) 0*n;    % x[n] = 0

[n_zir,y_zir] = solve_diff2_iter(a1,a0,b0,b1,b2,x_zero_fun,IC_ZIR,50);

figure;
stem(n_zir,y_zir,'filled'); xlabel('n'); ylabel('y_{zi}[n]');
title('Zero-Input Response (Iterative) – System 1');

%% 2) y[n+2] -1.732 y[n+1] + y[n] = 3x[n] +2x[n-1]
% and   y[n+2] +1.732 y[n+1] + y[n] = 3x[n] +2x[n-1]

% First system
a1_1 = -1.732; a0_1 = 1;
b0_1 = 3;    b1_1 = 2; b2_1 = 0;

% Impulse (iterative, zero-state)
[n_imp2,y_imp2] = solve_diff2_iter(a1_1,a0_1,b0_1,b1_1,b2_1,x_imp_fun,IC_zero,50);

% Impulse via filter
b2sys1 = [b0_1 b1_1 b2_1];
a2sys1 = [1 a1_1 a0_1];
h2_filter = filter(b2sys1,a2sys1,delta(n0));

% Zero-input (iterative)
[n_zir2,y_zir2] = solve_diff2_iter(a1_1,a0_1,b0_1,b1_1,b2_1,x_zero_fun,IC_ZIR,50);

% Second system (with +1.732)
a1_2 = 1.732; a0_2 = 1;
b0_2 = 3;    b1_2 = 2; b2_2 = 0;

[n_imp2b,y_imp2b] = solve_diff2_iter(a1_2,a0_2,b0_2,b1_2,b2_2,x_imp_fun,IC_zero,50);
b2sys2 = [b0_2 b1_2 b2_2];
a2sys2 = [1 a1_2 a0_2];
h2b_filter = filter(b2sys2,a2sys2,delta(n0));
[n_zir2b,y_zir2b] = solve_diff2_iter(a1_2,a0_2,b0_2,b1_2,b2_2,x_zero_fun,IC_ZIR,50);

% (Similarly define for systems 3, 4; for 5 use 3rd-order version.)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 2 – ZERO-STATE & TOTAL RESPONSE
% Same systems, now with specific x[n] (u[n], 0.9^n u[n], etc.)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Example for System 1 with x[n] = u[n] and x[n] = 0.9^n u[n]

a1 = -1.9; a0 = 0.9;
b0 = 2;    b1 = -3; b2 = 1;

IC_zero.y1 = 0; IC_zero.y2 = 0; IC_zero.n1 = -2;
IC_ZIR.y1 = 2; IC_ZIR.y2 = -1; IC_ZIR.n1 = -2;

% x[n] = u[n]
x_u_fun = @(n) u(n);

[n_ys_u,ys_u] = solve_diff2_iter(a1,a0,b0,b1,b2,x_u_fun,IC_zero,50);
[n_zir_u,yzir_u] = solve_diff2_iter(a1,a0,b0,b1,b2,x_zero_fun,IC_ZIR,50);

```

```

y_total_u = ys_u + yzir_u;      % total response

figure;
subplot(3,1,1); stem(n_ys_u,ys_u,'filled'); title('Zero-State (u[n]) – Sys1');
subplot(3,1,2); stem(n_zir_u,yzir_u,'filled'); title('Zero-Input – Sys1');
subplot(3,1,3); stem(n_ys_u,y_total_u,'filled'); title('Total Response (u[n]) – Sys1');

% x[n] = 0.9^n u[n]
x_g_fun = @(n) (0.9.^n).*u(n);

[n_ys_g,ys_g] = solve_diff2_iter(a1,a0,b0,b1,b2,x_g_fun,IC_zero,50);
y_total_g = ys_g + yzir_u;      % same ZIR

figure;
subplot(2,1,1); stem(n_ys_g,ys_g,'filled'); title('Zero-State (0.9^n u[n]) – Sys1');
subplot(2,1,2); stem(n_ys_g,y_total_g,'filled'); title('Total Response – Sys1');

% (Repeat same pattern for other systems and their given inputs.)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% EXERCISE 3 – ZERO STATE RESPONSE VIA CONVOLUTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n = -20:40;
u = @(n) double(n>=0);

% 1) h[n] = (-2)^n u[n-1] ; x[n] = e^{-n} u[n+1]
h1 = (-2).^n .* u(n-1);
x1 = exp(-n) .* u(n+1);

y1 = conv(h1,x1);
n_y1 = (n(1)+n(1)):(n(end)+n(end));

figure;
stem(n_y1,y1,'filled');
xlabel('n'); ylabel('y_1[n]');
title('Zero-State Response (Exercise 3, part 1)');

% 2) h[n] = 1/2{δ[n-2]-(-2)^{n+1}}u[n-3] ; x[n] = 3^{n-1} u[n+2]
h2 = 0.5*(delta(n-2) - (-2).^(n+1)).*u(n-3);
x2 = (3.^(n-1)).*u(n+2);
y2 = conv(h2,x2);
n_y2 = n(1)+n(1):n(end)+n(end);

figure;
stem(n_y2,y2,'filled');
title('Zero-State Response (Exercise 3, part 2)');

% 3) h[n] = {(2)^{n-2} + 3(-5)^{n+2}}u[n-1] ; x[n] = 3^{n+2}u[n+1]
h3 = ((2).^(n-2) + 3*(-5).^(n+2)).*u(n-1);
x3 = (3.^(n+2)).*u(n+1);
y3 = conv(h3,x3);
n_y3 = n(1)+n(1):n(end)+n(end);

figure;
stem(n_y3,y3,'filled');
title('Zero-State Response (Exercise 3, part 3)');

```

```
% 4)  $h[n] = 3(n-2)2^{n-3} u[n-4]$  ;  $x[n] = 3^{-n+2} u[n+3]$ 
h4 = 3.*(n-2).*(2).^(n-3).*u(n-4);
x4 = 3.^(-n+2).*u(n+3);
y4 = conv(h4,x4);
n_y4 = n(1)+n(1):n(end)+n(end);
```

```
figure;
stem(n_y4,y4,'filled');
title('Zero-State Response (Exercise 3, part 4)');
```

```
% 5)  $h[n] = (3)^n \cos(\pi/3 - 0.5)u[n]$  ;  $x[n] = 2^n u[n-1]$ 
h5 = (3.^n).*cos(pi/3 - 0.5).*u(n);
x5 = (2.^n).*u(n-1);
y5 = conv(h5,x5);
n_y5 = n(1)+n(1):n(end)+n(end);
```

```
figure;
stem(n_y5,y5,'filled');
title('Zero-State Response (Exercise 3, part 5)');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EXERCISE 4 – Classical method vs numerical (template)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Example: compare closed-form  $y_{cl}[n]$  (if you derive it manually) with
% numerically computed  $y_{num}[n]$  from solve_diff2_iter for any one system.
```

```
% Suppose you derived  $y_{cl}(n)$  for system 1 with input  $u[n]$ :
y_cl_fun = @(n) ...; % <== put your analytic formula here, vectorized
```

```
[n_num,y_num] = solve_diff2_iter(a1,a0,b0,b1,b2,x_u_fun,IC_zero,50);
y_cl = y_cl_fun(n_num);
```

```
figure;
stem(n_num,y_num,'filled'); hold on;
stem(n_num,y_cl,'o'); hold off;
legend('Numerical','Classical');
title('Comparison of Classical & Numerical Solutions – Example System');
'''
```