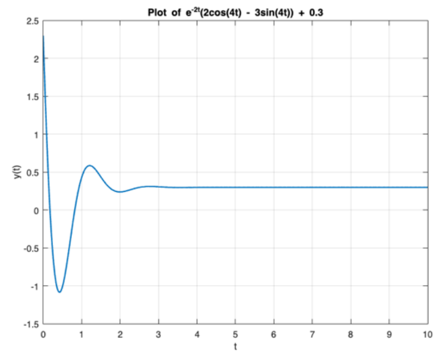
Q3(b)

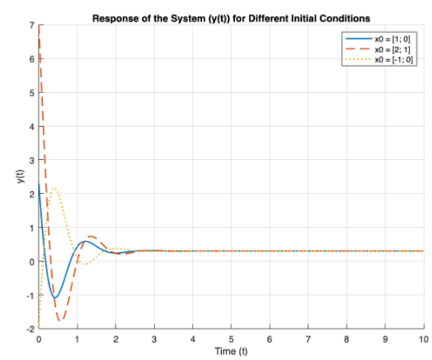
```
% Define the time range
t = linspace(0, 10, 1000);

% Define the function
y = exp(-2*t) .* (2*cos(4*t) - 3*sin(4*t)) + 0.3;

% Plot the function
figure;
plot(t, y, 'LineWidth', 1.5);
xlabel('t');
ylabel('y(t)');
title('Plot of e^{-2t}(2cos(4t) - 3sin(4t)) + 0.3');
grid on;
```



Q3 C)的圖上有包含 b)的圖再畫一次 （藍線）

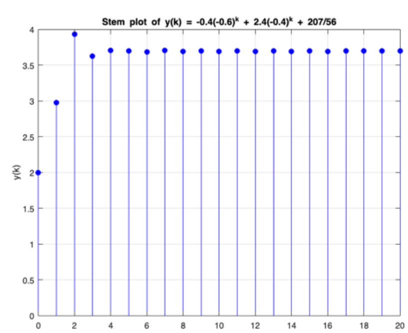
Q3 c)

```
% Define system matrices
A = [-2 4; -4 -2];
B = [1; 1];
C = [2 3];
D = 0;

% Define the system as dx/dt = Ax + Bu ,u=1
system = @(t, x) A*x + B;

% Time span for simulation
t_span = [0 10];

% Solve the system using ode45
[t, x] = ode45(system, t_span, x0);

% Compute output y(t)
t11 = linspace(0, 10, 1000);
y11 = exp(-2*t11) .* (2*cos(4*t11) - 3*sin(4*t11)) + 0.3;
y = C*x';

% Define new initial conditions
x0_1 = [2; 1]; % initial condition 1
x0_2 = [-1; 0]; % initial condition 2

% Simulate for first initial condition
[t1, x1] = ode45(system, t_span, x0_1);
y1 = C*x1';

% Simulate for second initial condition
[t2, x2] = ode45(system, t_span, x0_2);
y2 = C*x2';

% Plot y(t) for different initial conditions
figure;
hold on;
plot(t11, y11, 'LineWidth', 1.5);
plot(t1, y1, '--', 'LineWidth', 1.5);
plot(t2, y2, ':', 'LineWidth', 1.5);
title('Response of the System (y(t)) for Different Initial Conditions');
xlabel('Time (t)');
ylabel('y(t)');
legend('x0 = [1; 0]', 'x0 = [2; 1]', 'x0 = [-1; 0]');
grid on;
```



Q4 b)

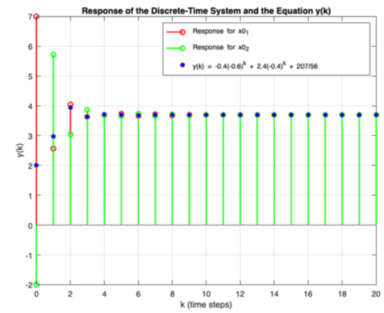
```
% Define the values for k
k = 0:20;

% Define the equation for y(k)
y = -0.4 * (-0.6).^k + 2.4 * (-0.4).^k + 207/56;

% Set y(0) to be 2
y(k == 0) = 2;

% Plot the values using a stem plot
stem(k, y, 'filled','blue');
xlabel('k');
ylabel('y(k)');
title('Stem plot of y(k) = -0.4(-0.6)^k + 2.4(-0.4)^k + 207/56');
grid on;
```



Q4 C)的圖上有包含 b)的圖再畫一次(藍點)

Q4 c)

```
% Define system matrices for discrete-time
A_d = [0 1; -0.24 -1];
B_d = [1; 1];
C_d = [2 3];

% Define different initial conditions
x0_1 = [2; 1]; % Initial condition 1
x0_2 = [-1; 0]; % Initial condition 2

num_steps = 20;

% Initialize variables
x_k1 = x0_1;
x_k2 = x0_2;
x_history1 = zeros(2, num_steps + 1); % Store state history
y_history1 = zeros(1, num_steps + 1); % Store output history
x_history2 = zeros(2, num_steps + 1); % Store state history
y_history2 = zeros(1, num_steps + 1); % Store output history

% Simulate the system
x_history1(:, 1) = x0_1; % Store initial state (k = 0)
y_history1(1) = C_d * x0_1; % Compute and store output for k = 0
x_history2(:, 1) = x0_2; % Store initial state (k = 0)
y_history2(1) = C_d * x0_2; % Compute and store output for k = 0

for k = 1:num_steps
    % Update state
    x_k1 = A_d * x_k1 + B_d;
    x_k2 = A_d * x_k2 + B_d;

    % Store state
    x_history1(:, k + 1) = x_k1;
    x_history2(:, k + 1) = x_k2;

    % Compute and store output
    y_history1(k + 1) = C_d * x_k1;
    y_history2(k + 1) = C_d * x_k2;
end

% Define the values for k (for the state equation plot)
k = 0:num_steps;  % Make sure it matches the number of steps

% Define the equation for y(k) (the second plot)
y = -0.4 * (-0.6).^k + 2.4 * (-0.4).^k + 207/56;
```



Q4 c)code續

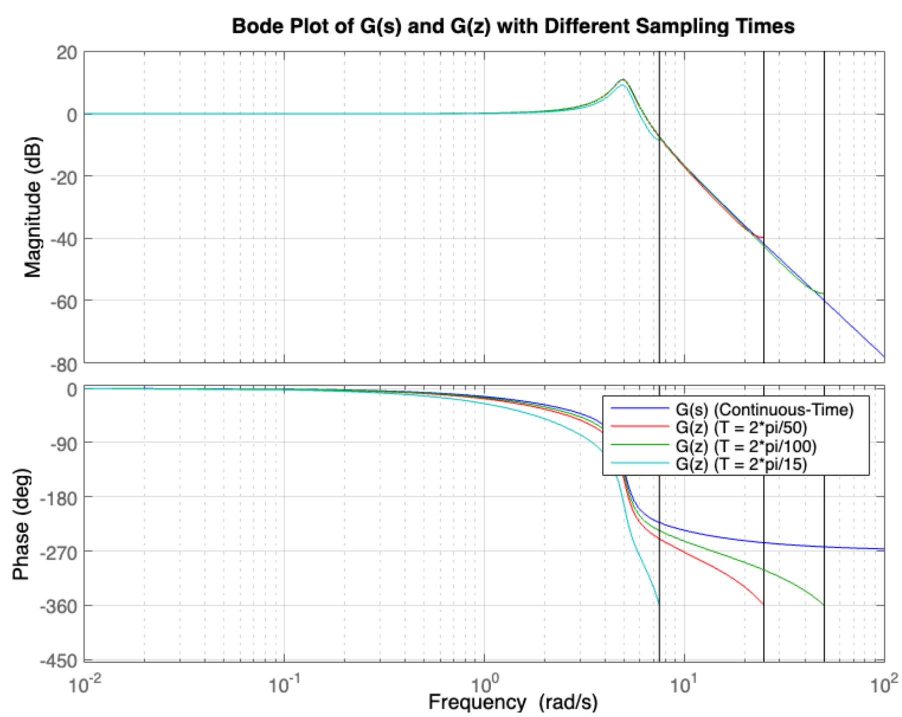
```
% Set y(0) to be 2 (as per the original equation)
y(k == 0) = 2;

% Plot all responses in the same figure
figure;

% Plot the response from the system (initial condition 1)
stem(0:num_steps, y_history1, 'LineWidth', 1.5, 'Color', 'r'); hold on;

% Plot the response from the system (initial condition 2)
stem(0:num_steps, y_history2, 'LineWidth', 1.5, 'Color', 'g');

% Plot the equation y(k)
stem(k, y, 'Marker', 'o', 'MarkerFaceColor', 'b', 'LineStyle', 'none');

% Labels and title
title('Response of the Discrete-Time System and the Equation y(k)');
xlabel('k (time steps)');
ylabel('y(k)');
grid on;

% Add a legend for clarity
legend('Response for x0_1', 'Response for x0_2', 'y(k) = -0.4(-0.6)^k + 2.4(-0.4)^k + 207/56');
```

Q5 b) c)

```
% Define continuous transfer function G(s)
num = 125; % Numerator of G(s)
den = [1 6 30 125]; % Denominator of G(s)
G_s = tf(num, den);

% Sampling times
T1 = 2*pi/50;
T2 = 2*pi/100;
T3 = 2*pi/15;

% Compute discrete transfer functions using ZOH
G_z_T1 = c2d(G_s, T1, 'zoh');
G_z_T2 = c2d(G_s, T2, 'zoh');
G_z_T3 = c2d(G_s, T3, 'zoh');

% Display results
disp('Discrete transfer function G(z) with T = 2*pi/50:');
disp(G_z_T1);
disp('Discrete transfer function G(z) with T = 2*pi/100:');
disp(G_z_T2);
disp('Discrete transfer function G(z) with T = 2*pi/15:');
disp(G_z_T3);

% Specify poles and zeros
[z1, p1, k1] = zpkdata(G_z_T1, 'v');
[z2, p2, k2] = zpkdata(G_z_T2, 'v');
[z3, p3, k3] = zpkdata(G_z_T3, 'v');

% Display poles and zeros
disp('Poles and Zeros for T = 2*pi/50:');
disp(['Zeros: ', num2str(z1')]);
disp(['Poles: ', num2str(p1')]);
disp('Poles and Zeros for T = 2*pi/100:');
disp(['Zeros: ', num2str(z2')]);
disp(['Poles: ', num2str(p2')]);
disp('Poles and Zeros for T = 2*pi/15:');
disp(['Zeros: ', num2str(z3')]);
disp(['Poles: ', num2str(p3')]);

% Plot Bode Plot
figure;
bode(Gs, 'b', G_z_T1, 'r',G_z_T2, 'g', G_z_T3, 'c');
grid on;
legend('G(s) (Continuous-Time)', 'G(z) (T = 2*pi/50)', 'G(z) (T = 2*pi/100)','G(z) (T = 2*pi/15)');
title('Bode Plot of G(s) and G(z) with Different Sampling Times');
```

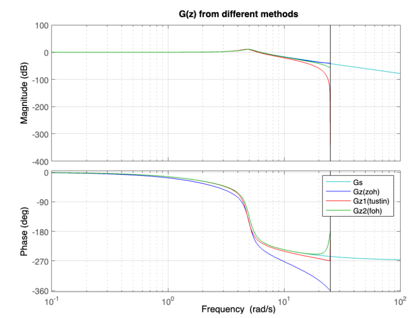
```
Discrete transfer function G(z) with T = 2*pi/50:
  tf with properties:

       Numerator: {[0 0.0338 0.1105 0.0232]}
     Denominator: {[1 -2.0565 1.6944 -0.4705]}
        Variable: 'z'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0.1257
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]
Discrete transfer function G(z) with T = 2*pi/100:
  tf with properties:

       Numerator: {[0 0.0047 0.0170 0.0039]}
     Denominator: {[1 -2.5746 2.2861 -0.6859]}
        Variable: 'z'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0.0628
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]
Discrete transfer function G(z) with T = 2*pi/15:
  tf with properties:

       Numerator: {[0 0.6824 1.2592 0.2102]}
     Denominator: {[1 0.6731 0.5597 -0.0810]}
        Variable: 'z'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
```

```
Discrete transfer function G(z) with T = 2*pi/15:
  tf with properties:

       Numerator: {[0 0.6824 1.2592 0.2102]}
     Denominator: {[1 0.6731 0.5597 -0.0810]}
        Variable: 'z'
         IODelay: 0
      InputDelay: 0
     OutputDelay: 0
       InputName: {''}
       InputUnit: {''}
      InputGroup: [1×1 struct]
      OutputName: {''}
      OutputUnit: {''}
     OutputGroup: [1×1 struct]
           Notes: [0×1 string]
        UserData: []
            Name: ''
              Ts: 0.4189
        TimeUnit: 'seconds'
    SamplingGrid: [1×1 struct]
Poles and Zeros for T = 2*pi/50:
Zeros: -3.0467     -0.2256
Poles: 0.76148-0.54959i         0.76148+0.54959i              0.53349+0i
Poles and Zeros for T = 2*pi/100:
Zeros: -3.384    -0.24481
Poles: 0.92211-0.29801i         0.92211+0.29801i               0.7304+0i
Poles and Zeros for T = 2*pi/15:
Zeros: -1.6596    -0.18557
Poles: -0.39812-0.7066i         -0.39812+0.7066i              0.12314+0i
```



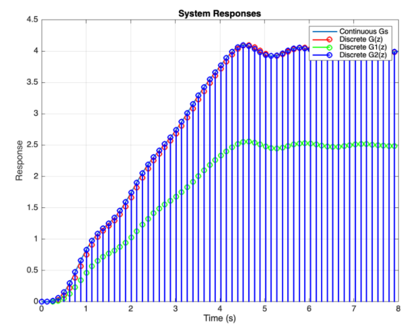
Q5 e)

```
T=2*pi/50;
s=tf('s');
Gs=125/(s^3+6*s^2+30*s+125);
Gz=c2d(Gs,T,'zoh');
Gz1=c2d(Gs,T,'tustin');
Gz2=c2d(Gs,T,'foh');
figure
bode(Gs,'c',Gz,'b',Gz1,'r',Gz2,'g')
grid on;
legend('Gs','Gz(zoh)','Gz1(tustin)','Gz2(foh)');
title('G(z) from different methods');
```



Q5 f)

```
% Continuous-time system transfer function Gs
num_cont = 125;
den_cont = [1, 6, 30, 125];
Gs = tf(num_cont, den_cont);

% Discrete-time system transfer functions G(z), G1(z), G2(z)
num_Gz = [0.03376, 0.1105, 0.02321];
den_Gz = [1, -2.056, 1.694, -0.4705];
Ts = 0.12566; % Sample time
Gz = tf(num_Gz, den_Gz, Ts);

num_G1z = [0.02031, 0.06094, 0.02031];
den_G1z = [1, -2.074, 1.702, -0.4654];
G1z = tf(num_G1z, den_G1z, Ts);

num_G2z = [0.008812, 0.08221, 0.07082, 0.005611];
den_G2z = [1, -2.056, 1.694, -0.4705];
G2z = tf(num_G2z, den_G2z, Ts);

% Time vector and input signal u(t)
t_cont = 0:0.01:8; % Time for continuous system
u_cont = t_cont .* (t_cont <= 4) + 4 .* (t_cont > 4); % u(t): piecewise function

t_disc = 0:Ts:8; % Time for discrete system
u_disc = t_disc .* (t_disc <= 4) + 4 .* (t_disc > 4); % Discrete version of u(t)

% Simulate continuous-time system
[y_cont, ~, ~] = lsim(Gs, u_cont, t_cont);

% Simulate discrete-time systems
y_Gz = lsim(Gz, u_disc, t_disc);
y_G1z = lsim(G1z, u_disc, t_disc);
y_G2z = lsim(G2z, u_disc, t_disc);

% Plotting results
figure;
plot(t_cont, y_cont, 'LineWidth', 1.5); hold on; % Continuous system
stem(t_disc, y_Gz, 'r', 'LineWidth', 1.5); % G(z)
stem(t_disc, y_G1z, 'g', 'LineWidth', 1.5); % G1(z)
stem(t_disc, y_G2z, 'b', 'LineWidth', 1.5); % G2(z)

xlabel('Time (s)');
ylabel('Response');
title('System Responses');
legend('Continuous Gs', 'Discrete G(z)', 'Discrete G1(z)', 'Discrete G2(z)');
grid on;
hold off;
```



y(t)是 continuous Gs 那條，連續時間以淺藍實線表示,discrete 以 stem 表示

discrete G(z), G1(z), G2(z),分別為 y(k) of G(z), G1(z), G2(z)