Q1

% PART 1: Projectile motion for fixed angle (θ = 45°)

```
% Constants
v = 20;            % initial speed in m/s
theta = 45;         % launch angle in degrees
g = 9.8;           % gravity in m/s^2

% Convert angle to radians
theta_rad = deg2rad(theta);

% Time of flight
T_flight = 2 * v * sin(theta_rad) / g;

% Time array
t = linspace(0, T_flight, 500);

% Equations of motion
x = v * cos(theta_rad) * t;
y = v * sin(theta_rad) * t - 0.5 * g * t.^2;



figure;
subplot(3,1,1)
plot(t, x,  'b','LineWidth', 1.5);
xlabel('Time','FontSize',10);
xlim([min(t) max(t)])
ylabel('Horizontal Distance (m)','FontSize',10);
title('Horizontal distance with time','FontSize',12);

subplot(3,1,2)
plot(t, y,  'r','LineWidth', 1.5);
xlabel('Time','FontSize',10);
xlim([min(t) max(t)])
ylabel('Vertical height (m)','FontSize',10);
title('Height with time','FontSize',12);

subplot(3,1,3)
plot(x, y,  'k','LineWidth', 1.5);
xlabel('Horizontal distance (m)','FontSize',10);
xlim([min(x) max(x)])
ylabel('Vertical height (m)','FontSize',10);
title('Horizontal distance vs height','FontSize',12);
```

% PART 2: Projectile motion for multiple launch angles

```
% Parameters
v = 40;  % initial speed in m/s
g = 9.8;
angles = [0 15 30 45 60 75 90];  % in degrees

% Create figure for all trajectories
figure;
hold on;

% Use color scheme
colors = lines(length(angles));

% Loop over angles
for i = 1:length(angles)
```

```
    theta = angles(i);
    theta_rad = deg2rad(theta);

    % Time of flight for current angle
    T = 2 * v * sin(theta_rad) / g;
    t = linspace(0, T, 300);

    % Trajectory
    x = v * cos(theta_rad) * t;
    y = v * sin(theta_rad) * t - 0.5 * g * t.^2;

    % Plot trajectory
    plot(x, y, 'Color', colors(i,:), 'LineWidth', 1.5, ...
        'DisplayName', sprintf('\\theta = %d°', theta));

    % Compute and display max height
    y_max = max(y);  % numerical value from trajectory
    fprintf('Angle: %2d° → Max Height: %.2f m\n', theta, y_max);
end

% Final plot settings
xlabel('Horizontal Distance (m)');
ylabel('Vertical Height (m)');
title('Projectile Trajectories for Various Launch Angles');
legend('Location', 'best');
grid on;
axis equal;
```

Q2

```
% Damped Harmonic Oscillator: Varying Amplitude A

% Fixed parameters
gamma = 0.2;      % Damping coefficient
omega = 2;        % Angular frequency (rad/s)
T = 10;           % Total time duration
t = linspace(0, T, 500);  % Time array

% Different amplitudes to compare
A_values = [0.5, 1.0, 1.5, 2.0, 5.0];

% Create figure
figure;
hold on;

% Loop over amplitudes and plot each x(t)
for i = 1:length(A_values)
    A = A_values(i);
    x = A * exp(-gamma * t) .* cos(omega * t);
    plot(t, x, 'LineWidth', 1.5, 'DisplayName', sprintf('A = %.1f', A));
end

% Axis labeling and title
xlabel('Time (s)', 'FontSize', 12);
ylabel('Displacement x(t)', 'FontSize', 12);
title('Damped Harmonic Oscillator: Effect of Amplitude A', 'FontSize', 14);
legend('Location', 'northeast');
```

```
grid on;
```

% Damped Harmonic Oscillator: Underdamped, Critically Damped, Overdamped

```
% Natural frequency
omega0 = 2;  % rad/s

% Time array
t = linspace(0, 10, 500);

% Damping values for three regimes
gamma_values = [0.5, 2.0, 4.0];

% Create figure
figure;
hold on;

for i = 1:length(gamma_values)
  gamma = gamma_values(i);

  if gamma < omega0  % Underdamped
    omega_d = sqrt(omega0^2 - gamma^2);
    x = exp(-gamma * t) .* (cos(omega_d * t) + (gamma / omega_d) * sin(omega_d * t));
    label = sprintf('Underdamped (\\gamma = %.1f)', gamma);

  elseif gamma == omega0  % Critically damped
    x = (1 + gamma * t) .* exp(-gamma * t);
    label = sprintf('Critically Damped (\\gamma = %.1f)', gamma);

  else  % Overdamped
    lambda1 = gamma + sqrt(gamma^2 - omega0^2);
    lambda2 = gamma - sqrt(gamma^2 - omega0^2);

    % Solve D1 and D2 using:
    % D1 + D2 = 1
    % -lambda1*D1 - lambda2*D2 = 0
    A = [1, 1; -lambda1, -lambda2];
    b = [1; 0];
    D = A \ b;  % Solves for D1 and D2

    D1 = D(1);
    D2 = D(2);

    x = D1 * exp(-lambda1 * t) + D2 * exp(-lambda2 * t);
    label = sprintf('Overdamped (\\gamma = %.1f)', gamma);
  end

  plot(t, x, 'LineWidth', 1.8, 'DisplayName', label);
end

% Plot settings
xlabel('Time (s)', 'FontSize', 12);
ylabel('Displacement x(t)', 'FontSize', 12);
title('Damped Harmonic Oscillator: Three Damping Regimes', 'FontSize', 14);
legend('Location', 'northeast');
grid on;
```

Q3

```
% Number of days and measurements per day
numDays = 5;
numMeasurements = 3;

% Generate random energy measurements (e.g., values between 5 and 15)
% Each row corresponds to a day, each column to a measurement
Energy = 5 + 10 * rand(numDays, numMeasurements);

% Plot the grouped bar chart
figure;
%bar(Energy, 'stacked');
bar(Energy, 'grouped');
title('Energy Measurements Over 5 Days');
xlabel('Day');
ylabel('Energy (arb. units)');

% Label x-axis ticks
xticklabels({'Day 1', 'Day 2', 'Day 3', 'Day 4', 'Day 5'});

% Add legend for measurement categories
legend({'Measurement 1', 'Measurement 2', 'Measurement 3'}, 'Fontsize',8,...
    'Location', 'northwest');


total_energy = sum(Energy, 2); % sum across columns (measurements)
[maxEnergy, maxDay] = max(total_energy);
fprintf('Day %d has the highest total energy: %.2f units\n', maxDay, maxEnergy);
```