Q1:

$y = [\theta_1, \theta_2, w_1, w_2]$

θ_1, θ_2 are the angles of two pendulum arms.

$w_1 = \theta_1'$, $w_2 = \theta_2'$ are the angular velocities of the pendulum arms.

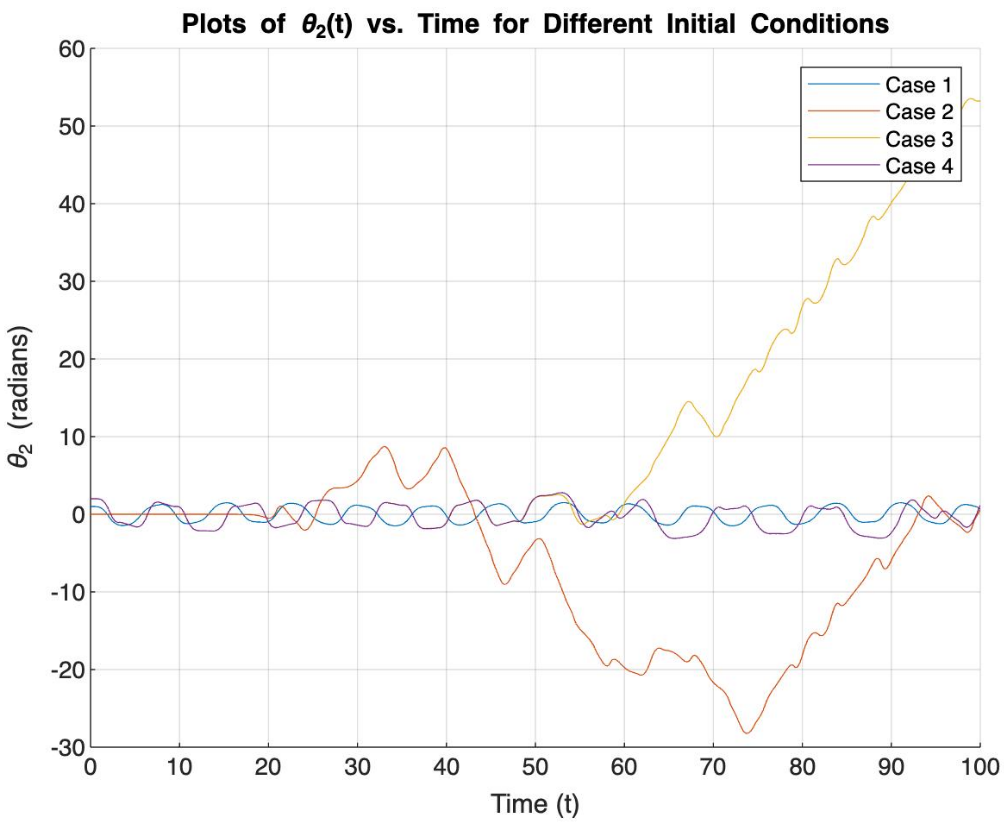
1-order system of equation is given by $\frac{dy}{dt} = f(y)$

$$f(y) = \begin{bmatrix} \theta_1' \\ \theta_2' \\ w_1' \\ w_2' \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \dfrac{-3\sin(\theta_1) - \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2)(w_2^2 + w_1^2\cos(\theta_1 - \theta_2))}{3 - \cos(2\theta_1 - 2\theta_2)} \\ \dfrac{2\sin(\theta_1 - \theta_2)(2w_1^2 + 2\cos(\theta_1) + w_2^2\cos(\theta_1 - \theta_2))}{3 - \cos(2\theta_1 - 2\theta_2)} \end{bmatrix}$$

```
function ydot = fpend(y)
    % Extract variables from input vector y
    theta1 = y(1);
    theta2 = y(2);
    omega1 = y(3);
    omega2 = y(4);

    % Compute derivatives
    delta = theta1 - theta2;

    denominator = 3 - cos(2*delta);

    theta1_ddot = (-3*sin(theta1) - sin(delta) - 2*sin(delta)*(omega2^2 + omega1^2*cos(delta))) / denominator;
    theta2_ddot = (2*sin(delta)*(2*omega1^2 + 2*cos(theta1) + omega2^2*cos(delta))) / denominator;

    % Return the derivative vector
    ydot = [omega1; omega2; theta1_ddot; theta2_ddot];
end
```

Q2:

```
>> % Define time parameters
t0 = 0;
t_end = 100;
h = 0.05;
t = t0:h:t_end;
n = length(t);

% Initial conditions for the four cases
y_init = [
    1, 1, 0, 0;          % Case 1
    pi, 0, 0, 1e-10;     % Case 2
    2, 2, 0, 0;          % Case 3
    2, 2 + 1e-3, 0, 0;   % Case 4
];

% Create a figure for the plot
figure;
hold on;

% Loop over each case
for case_idx = 1:size(y_init, 1)
    % Initialize state matrix y
    y = zeros(n, 4);
    y(1, :) = y_init(case_idx, :);

    % Perform Runge-Kutta integration
    for i = 1:n-1
        k1 = h * fpend(y(i, :)');
        k2 = h * fpend(y(i, :)' + k1/2);
        k3 = h * fpend(y(i, :)' + k2/2);
        k4 = h * fpend(y(i, :)' + k3);
        y(i+1, :) = y(i, :) + (k1 + 2*k2 + 2*k3 + k4)' / 6;
    end

    % Plot \theta_2(t) for the current case
    plot(t, y(:, 2), 'DisplayName', ['Case ', num2str(case_idx)]);
end

% Customize the plot
xlabel('Time (t)');
ylabel('\theta_2 (radians)');
title('Plots of \theta_2(t) vs. Time for Different Initial Conditions');
legend show;

grid on;
hold off;
```

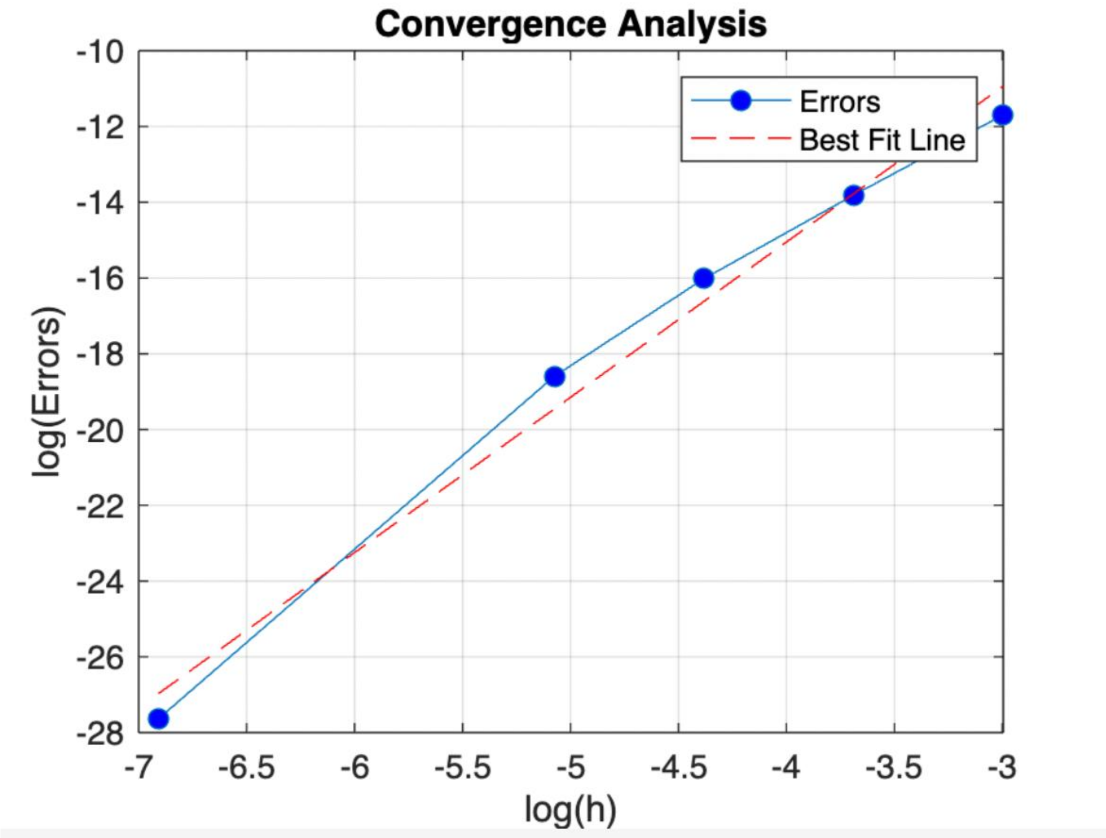


Q3:

```
% Initial conditions for Case 1
initial_conditions = [1, 1, 0, 0];
tspan = [0, 100];  % we can know that from the question
step_sizes = [0.05, 0.05/2, 0.05/4, 0.05/8, 0.001];  % Step sizes used
theta2_final = zeros(size(step_sizes));  % Array to store final theta2 values

tolerance = 1e-12;

for i = 1:length(step_sizes)
    h = step_sizes(i);
    [t, y] = rk4(@fpend, initial_conditions, tspan, h);
    theta2_final(i) = y(end, 2);  % Store the value of theta2 at t = 100
end

% Reference value from the smallest step size
theta2_ref = theta2_final(end);

% Calculate absolute errors
errors = abs(theta2_final - theta2_ref);

% Apply numerical tolerance to avoid zero errors
errors(errors < tolerance) = tolerance;

% Display computed values
disp('Step Sizes (h):');
disp(step_sizes);
disp('Theta2 values at t = 100:');
disp(theta2_final);
disp('Computed Errors:');
disp(errors);

% Log-log plot of errors vs. step sizes
figure;
log_h = log(step_sizes);
log_errors = log(errors);
plot(log_h, log_errors, '-o', 'MarkerFaceColor', 'b');  % Line and circle markers
xlabel('log(h)');
ylabel('log(Errors)');
title('Convergence Analysis');
grid on;
```

```
% Estimate the order of convergence
p = polyfit(log_h, log_errors, 1);
slope = p(1);
intercept = p(2);

% Display estimated slope
disp(['Estimated order of convergence: ', num2str(slope)]);

% Best fit line
hold on;
log_h_fit = linspace(min(log_h), max(log_h), 100);
log_errors_fit = intercept + slope * log_h_fit;
plot(log_h_fit, log_errors_fit, 'r--');  % Best fit line in red dashed
legend('Errors', 'Best Fit Line');
hold off;
```

```
Step Sizes (h):
    0.0500    0.0250    0.0125    0.0063    0.0010

Theta2 values at t = 100:
    0.7100    0.7100    0.7100    0.7100    0.7100

Computed Errors:
   1.0e-05 *

    0.8356    0.1004    0.0111    0.0008    0.0000

Estimated order of convergence: 4.0987
```


Convergence Analysis
Errors
Best Fit Line
log(Errors)
log(h)
-10
-12
-14
-16
-18
-20
-22
-24
-26
-28
-7
-6.5
-6
-5.5
-5
-4.5
-4
-3.5
-3