AE332 – Modeling and Analysis Lab

Name: Shingala Vaidik ID : SC21B054

1. Solving a First Order ODE: Write a “deriv” function for the given differential equation, $\dot{x} = cos(t)$. Obtain numerical solution and verify it against actual solution. Change “rtol” and “atol” given to “ode” and study their effect on accuracy of solution.

Matlab Code:

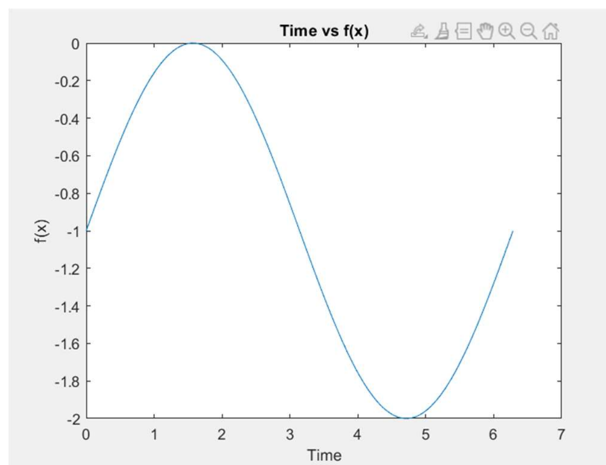
```
% Define the differential equation
f = @(t, x) cos(t);

% Actual solution of the differential equation
actual_solution = @(t) sin(t) - 1;

% Given initial condition
t0 = 0;    % Initial time
x0 = -1;   % Initial value of x

% Time span for integration
tspan = [t0, 2*pi]; % Choose the appropriate time span

% Varying rtol and atol values
rtol_values = [1e-3, 1e-6, 1e-9];
atol_values = [1e-3, 1e-6, 1e-9];

% Initialize an array to store maximum errors
max_errors = zeros(length(rtol_values), length(atol_values));

% Loop to calculate maximum errors for each combination of rtol and
atol
for i = 1:length(rtol_values)
    for j = 1:length(atol_values)
        opts = odeset('RelTol', rtol_values(i), 'AbsTol',
atol_values(j));

        % Solve the differential equation using ode45 with current
options
        [t, x] = ode45(f, tspan, x0, opts);

        % Calculate the actual solution
        actual_x = actual_solution(t);

        % Calculate errors
        errors = abs(actual_x - x);

        % Store the maximum error for this combination
        max_errors(i, j) = max(errors);
    end
```

```
end

% Display the maximum errors and find the best combination
disp('Maximum Errors:');
disp(max_errors);

[min_error, min_idx] = min(max_errors(:));
[min_rtol_idx, min_atol_idx] = ind2sub(size(max_errors), min_idx);

best_rtol = rtol_values(min_rtol_idx);
best_atol = atol_values(min_atol_idx);

disp(['Best combination: rtol = ' num2str(best_rtol) ', atol = '
num2str(best_atol)]);

figure;
plot(t,x);
xlabel('Time');
ylabel('f(x)');
title('Time vs f(x)');
```

Output:

```
>> que1
Maximum Errors:
   1.0e-05 *

    0.7548    0.7548    0.7548
    0.7673    0.6123    0.5751
    0.7673    0.7673    0.0007

Best combination: rtol = 1e-09,
atol = 1e-09
```

Graph:



Hence we can see that after giving more tolerance the maximum error is significantly decreased.

2. Solving a System of Coupled ODEs: Write a "deriv" routine for this system. Selecting a set of random values for the coefficients aij, and the initial conditions (t0,x0), solve the system using "ode". Study variation of accuracy of solution with "atol" and "rtol", by comparing with exact solution.

How do we solve

$$\frac{dy}{dt} = -y + 3x \quad (1)$$

$$\frac{dx}{dt} = 4x - 2y \quad (2)$$

with initial conditions $y(0) = 2$ **and** $x(0) = 1$**?**

Step 1: First make x the subject of (1), $x = \frac{1}{3}\left(\frac{dy}{dt} + y\right)$.

Step 2: Substitute in (2) to get $\frac{d}{dt}\left[\frac{1}{3}\left(\frac{dy}{dt} + y\right)\right] = 4\left[\frac{1}{3}\left(\frac{dy}{dt} + y\right)\right] - 2y$ **which simplifies to** $\frac{d^2y}{dt^2} - 3\frac{dy}{dt} + 2y = 0$ **with initial conditions** $y(0) = 2$ **and** $\frac{dy}{dt}(0) = -y(0) + 3x(0) = -2 + 3 = 1$.

Step 3: The roots of the auxiliary equation $m^2 - 3m + 2 = 0$ **are 2, 1. Hence the solution to the homogeneous problem is** $y = Ae^t + Be^{2t}$.

Step 4: Substituting the initial conditions gives $A = 3, B = -1$ **i.e.** $y = 3e^t - e^{2t}$.

Step 5: Now we have $x = \frac{1}{3}\left[\frac{dy}{dt} + y\right] = \frac{1}{3}\left[\frac{d}{dt}\left(3e^t - e^{2t}\right) + 3e^t - e^{2t}\right] = 2e^t - e^{2t}$. **Hence the solution is** $y(t) = 3e^t - e^{2t}$ **and** $x(t) = 2e^t - e^{2t}$.

Matlab Code:

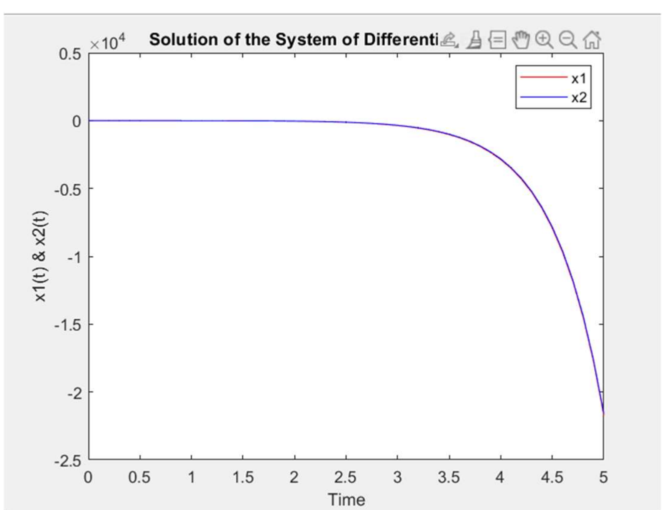
```
% Define the coefficients
a11 = 4;
a12 = -2;
a21 = 3;
a22 = -1;

% Define the system of differential equations
f = @(t, x) [a11*x(1) + a12*x(2); a21*x(1) + a22*x(2)];

x0 = [1; 2];    % Initial values of x1 and x2
t0 = 0;         % Initial time

% Time span for integration
tspan = (t0:0.1:5);

[t,x]= ode45(f,tspan,x0);
plot(t,x(:,1), '-r',t,x(:,2),'-b');
legend('x1', 'x2');
xlabel('Time');
ylabel('x1(t) & x2(t)');
title('Solution of the System of Differential Equations');

% Actual solutions
actual_x1_sol = @(t) 2*exp(t) - exp(2*t);
actual_x2_sol = @(t) 3*exp(t) - exp(2*t);
```

```
% Calculate the actual solution
actual_x1 = actual_x1_sol(t);
actual_x2 = actual_x2_sol(t);

err1 = abs(actual_x1-x(:,1));
err2 = abs(actual_x2-x(:,2));
e1 = max(err1);
e2 = max(err2);

disp(['Max Error for x1 without tolerance: ' num2str(e1)]);
disp(['Max Error for x2 without tolerance: ' num2str(e2)]);

% Adding tolerances
tol = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);
[t,x1]= ode45(f,tspan,x0,tol);
err3 = abs(actual_x1-x1(:,1));
err4 = abs(actual_x2-x1(:,2));
e3 = max(err3);
e4 = max(err4);

disp(['Max Error for x1 with tolerance: ' num2str(e3)]);
disp(['Max Error for x2 with tolerance: ' num2str(e4)]);
```

Output:

```
>> que2
Max Error for x1 without tolerance: 3.7219
Max Error for x2 without tolerance: 3.7202
Max Error for x1 with tolerance: 1.6943e-06
Max Error for x2 with tolerance: 1.6941e-06
```

Graph:



Hence we can see that after giving more tolerance the maximum error is significantly decreased.

3. Simulating a Simple Pendulum : Consider a simple pendulum of length l, swinging in a vertical plane. The equation of motion (for a frictionless case) is

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta)$$

Let $Z_1 = \theta$ and, $Z_2 = \dot{\theta}$ Then,

$$\dot{Z_1} = Z_2; \qquad \dot{Z_2} = -\frac{g}{l}\sin(Z_2)$$

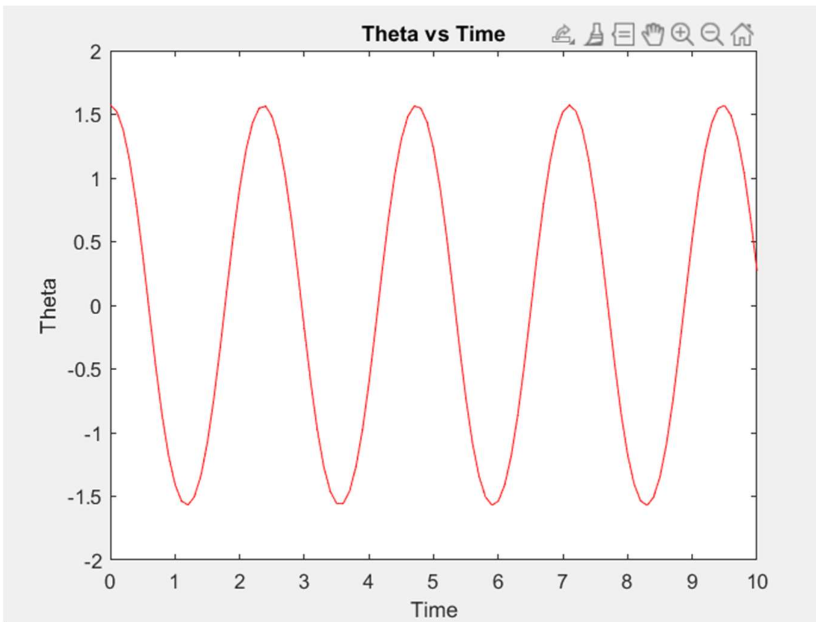
Matlab Code:

```
% Define the coefficients
a11 = 0;
a12 = 1;
a21 = -9.81;
a22 = 0;

% Define the system of differential equations
f = @(theta, z) [a11*z(1) + a12*z(2); a21*sin(z(1)) + a22*z(2)];

z0 = [pi/2; 0];    % Initial values of theta and theta_dot
theta_0 = 0;       % Initial time

% Time span for integration
theta_span = [theta_0:0.1:10];

[t, z] = ode45(f, theta_span, z0);
plot(t, z(:, 1), '-r');
xlabel('Time');
ylabel('Theta');
title('Theta vs Time');

% Calculate the energy
Energy1 = 0.5 * z(:, 2).^2 + 10 * (1 - cos(z(:, 1)));
e1 = max(Energy1) - min(Energy1);

disp(['Max Error for Energy without tolerance: ' num2str(e1)]);

% Adding tolerances
tol = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
[t, z1] = ode45(f, theta_span, z0, tol);

% Calculate the energy with tolerance
Energy2 = 0.5 * z1(:, 2).^2 + 10 * (1 - cos(z1(:, 1)));
e2 = max(Energy2) - min(Energy2);

disp(['Max Error for Energy with tolerance: ' num2str(e2)]);
```

Output:

>> que3
Max Error for Energy without tolerance: 0.20596
Max Error for Energy with tolerance: 0.18996

Graph:



Hence we can see that after giving more tolerance the maximum error is significantly decreased.