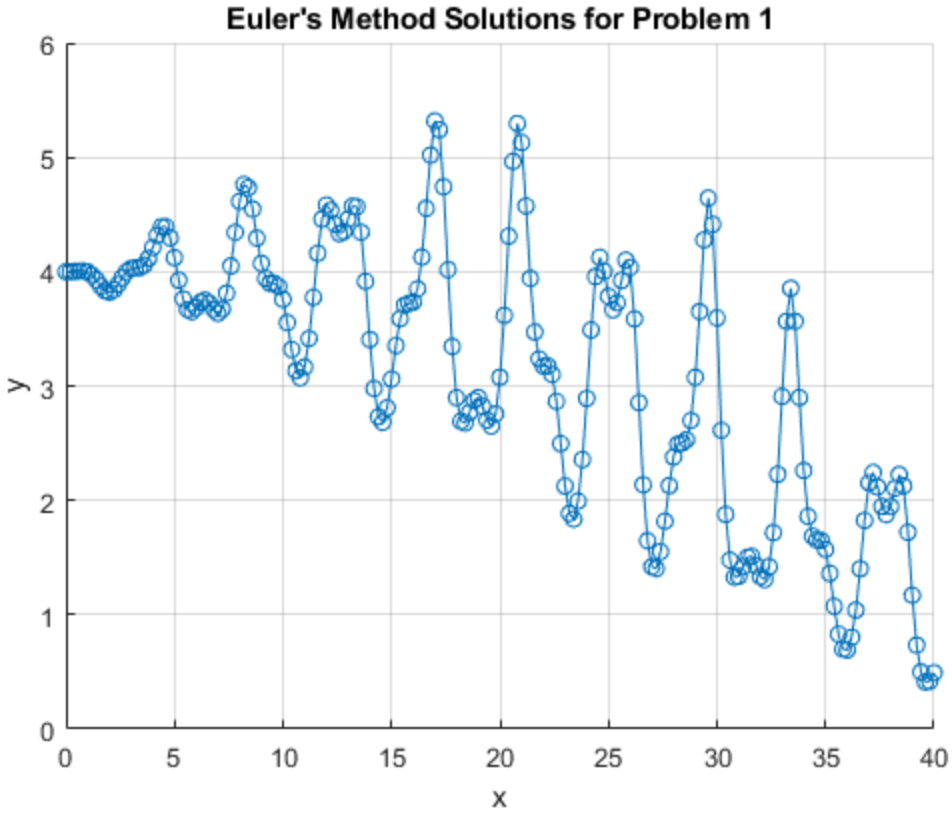
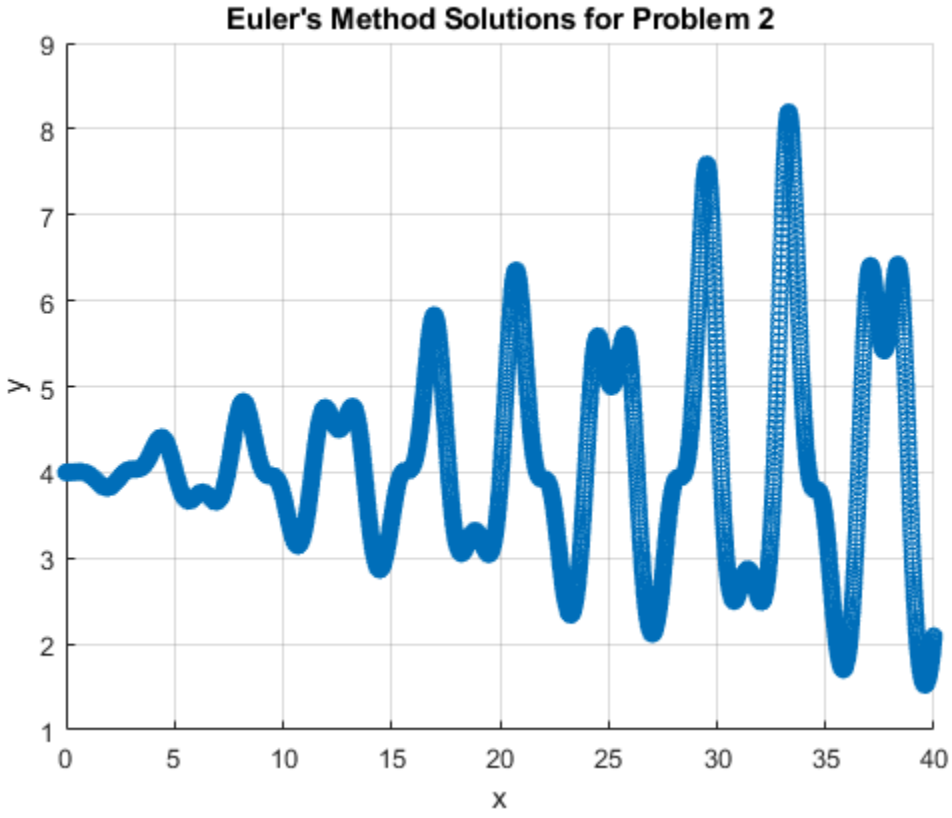
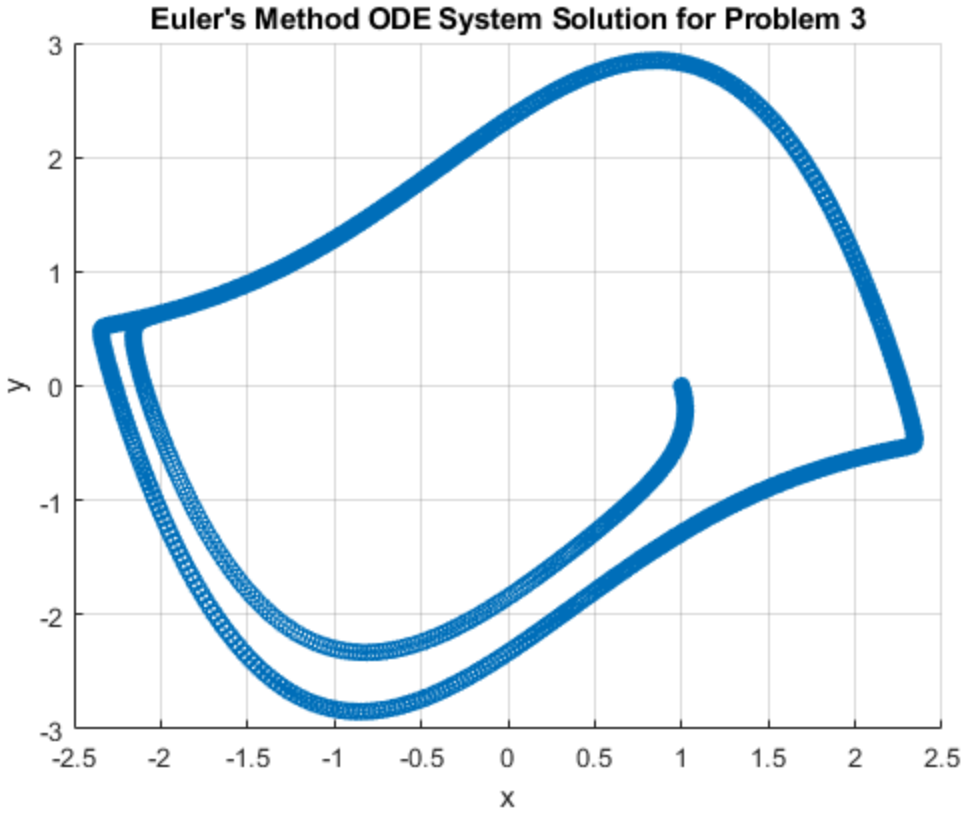
```
% Pierce Zhang, CMOR220, FALL 2023, Competency Euler's Method and ODEs
% competency_eulers_method.m
% Answers to competency on Euler's method and ODEs
% Last modified: 4 October 2023

% Driver
function competency_eulers_method
    % This function uses Euler's method to approximate the solution to an ODE.
    % Inputs:
    % - fun: The function representing the ODE, e.g., @prob1_fun.
    % - x0: Initial value of x.
    % - y0: Initial value of y.
    % - b: The upper limit of the x-range for the approximation.
    % - dx: The step size.
    % Output:
    % - y: The approximate solution to the ODE.

    % PROBLEM 1 SOLUTIONS
    [y] = eulers_method(@prob1_fun, 0, 4, 40, 0.2);
    figure; hold on; grid on;
    plot(0:0.2:40, y, '-o');
    title("Euler's Method Solutions for Problem 1");
    xlabel('x'); ylabel('y');

    clear;

    % PROBLEM 2 SOLUTIONS
    [y] = eulers_method(@prob1_fun, 0, 4, 40, 0.01);
    figure; hold on; grid on;
    plot(0:0.01:40, y, '-o');
    title("Euler's Method Solutions for Problem 2");
    xlabel('x'); ylabel('y');
    % QUESTION ANSWERED: The plot for #2 has more inflection points and is
    % more chaotic, likely because the smaller dx is capturing more of the
    % erratic behavior of the function that would otherwise have been
    % approximated away.

    clear;

    % PROBLEM 3 SOLUTIONS
    [x, y] = eulers_method_sys(@prob3_funx, @prob3_funy, 1, 0, 0, 40, 0.01);
    figure; hold on; grid on;
    plot(x, y, '-o');
    title("Euler's Method ODE System Solution for Problem 3");
    xlabel('x'); ylabel('y');

    clear;

    % PROBLEM 4 SOLUTIONS
    k = 0.035; TA = 70;
    dTdt = @(t, T) -k * (T - TA);
    tspan = [0 120];
```

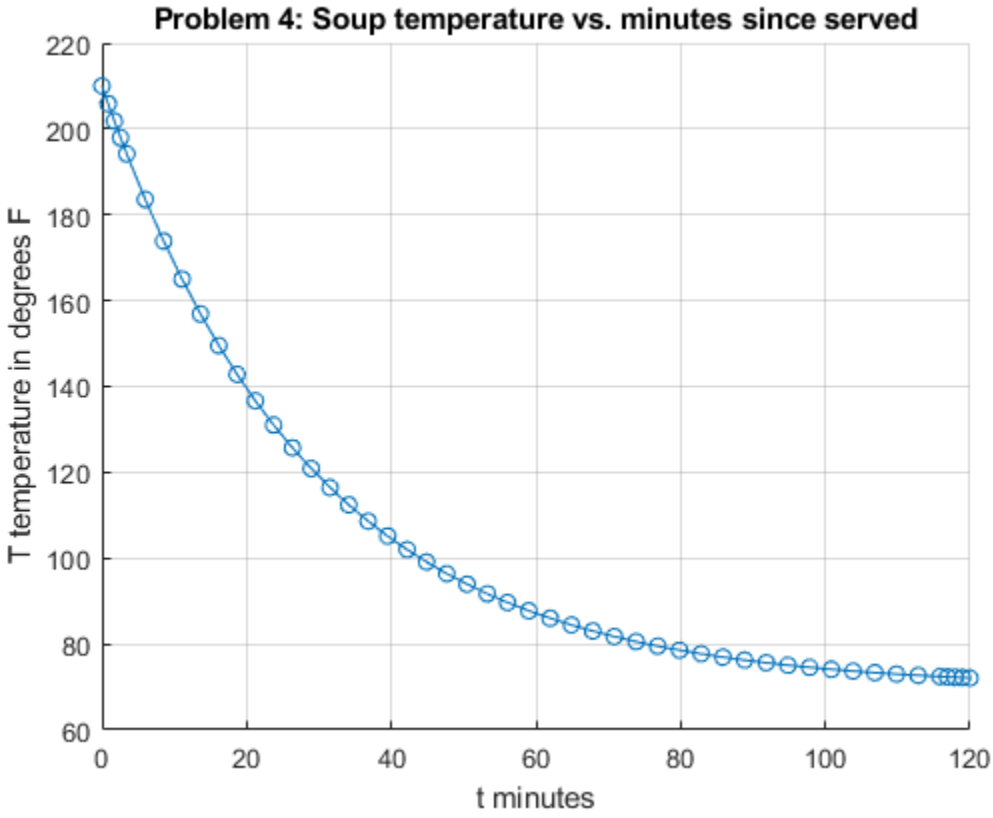
```
    O = odeset('RelTol', 0.00001);
    [tout, yout] = ode45(dTdt, tspan, 210, O);
    figure; hold on; grid on;
    plot(tout, yout, '-o');
    title("Problem 4: Soup temperature vs. minutes since served");
    xlabel('t minutes'); ylabel('T temperature in degrees F');

    clear;

    % PROBLEM 5 SOLUTIONS
    bs = 0.1:0.1:0.6;
    figure;
    set(gcf, 'Position', [350 350 1000 500])
    for b_i = 1:6
        b = bs(b_i);
        M = 7900010; a = 0.7;
        dPdt = @(t, P) [-a/M * P(1) * P(2);
                        a/M * P(1) * P(2) - b * P(2);
                        b * P(2)];
        O = odeset('RelTol', 1e-6);
        [tout, pout] = ode23(dPdt, [0 150], [7900000, 10, 0], O);
        subplot(2, 3, b_i); hold on; grid on;
        plot(tout, pout(:, 1));
        plot(tout, pout(:, 2));
        plot(tout, pout(:, 3));
        title("b=" + num2str(b));
        legend("S(t)", "R(t)", "I(t)");
    end
end

function [yprime] = prob1_fun(x, y)
    % This function defines the ODE for Problem 1.
    % Inputs:
    % - x: The independent variable.
    % - y: The dependent variable.
    % Output:
    % - yprime: The derivative of y with respect to x.
    yprime = 0.05 * x * y * sin(x) * cos(2.5 * x);
end

function [y] = eulers_method(fun, x0, y0, b, dx)
    % This function uses Euler's method to approximate the solution to an ODE.
    % Inputs:
    % - fun: The function representing the ODE, e.g., @prob1_fun.
    % - x0: Initial value of x.
    % - y0: Initial value of y.
    % - b: The upper limit of the x-range for the approximation.
    % - dx: The step size.
    % Output:
    % - y: The approximate solution to the ODE.
    x = x0:dx:b;
    y = zeros(1, length(x));
    y(1) = y0;
    for n = 1:length(x) - 1
```

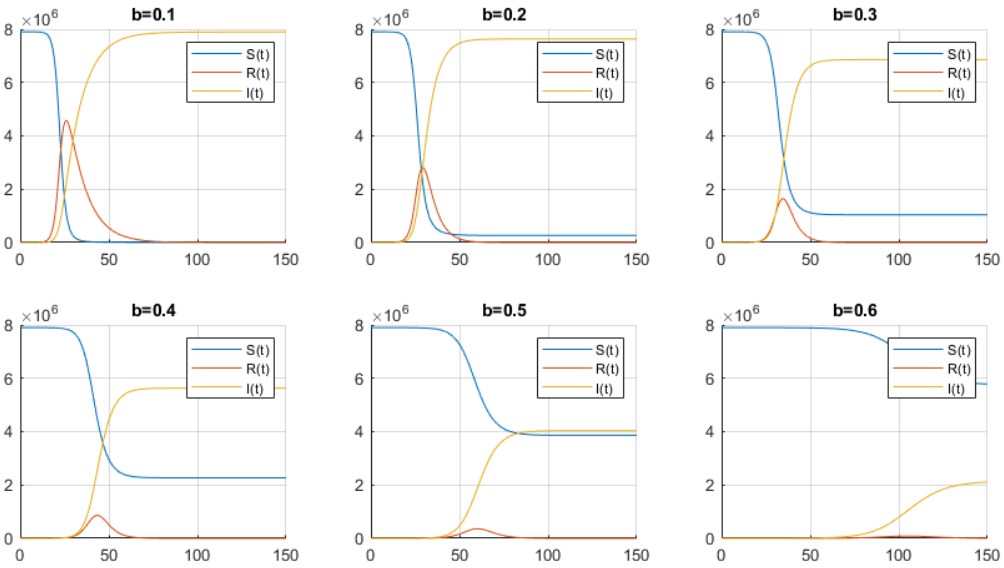
```
        dy = (fun(x(n), y(n))) * dx;
        y(n + 1) = y(n) + dy;
    end
end

function [xprime] = prob3_funx(x, y)
    % This function defines the first ODE for Problem 3.
    % Inputs:
    % - x: The independent variable.
    % - y: The dependent variable.
    % Output:
    % - xprime: The derivative of x with respect to t.
    xprime = y + 0.2 * x;
end

function [yprime] = prob3_funy(x, y)
    % This function defines the second ODE for Problem 3.
    % Inputs:
    % - x: The independent variable.
    % - y: The dependent variable.
    % Output:
    % - yprime: The derivative of y with respect to t.
    yprime = (1 - x^2) * y - x;
end

function [x, y] = eulers_method_sys(funx, funy, x0, y0, t, tf, dt)
    % This function uses Euler's method to approximate a system of ODEs.
    % Inputs:
    % - funx: The function representing the first ODE.
    % - funy: The function representing the second ODE.
    % - x0: Initial value of x.
    % - y0: Initial value of y.
    % - t: Initial time.
    % - tf: Final time.
    % - dt: The time step.
    % Output:
    % - x: The approximate solution for x.
    % - y: The approximate solution for y.
    tdomain = t:dt:tf;
    x = zeros(1, length(tdomain));
    y = zeros(1, length(tdomain));
    x(1) = x0;
    y(1) = y0;
    for n = 1:length(tdomain) - 1
        dx = (funx(x(n), y(n))) * dt;
        dy = (funy(x(n), y(n))) * dt;
        x(n + 1) = x(n) + dx;
        y(n + 1) = y(n) + dy;
    end
end
```


Euler's Method Solutions for Problem 1
y
x
0
1
2
3
4
5
6
0
5
10
15
20
25
30
35
40


Euler's Method Solutions for Problem 2
y
x
1
2
3
4
5
6
7
8
9
0
5
10
15
20
25
30
35
40


Euler's Method ODE System Solution for Problem 3
3
2
1
0
-1
-2
-3
y
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x


Problem 4: Soup temperature vs. minutes since served
220
200
180
160
140
120
100
80
60
T temperature in degrees F
0
20
40
60
80
100
120
t minutes



Published with MATLAB® R2023a