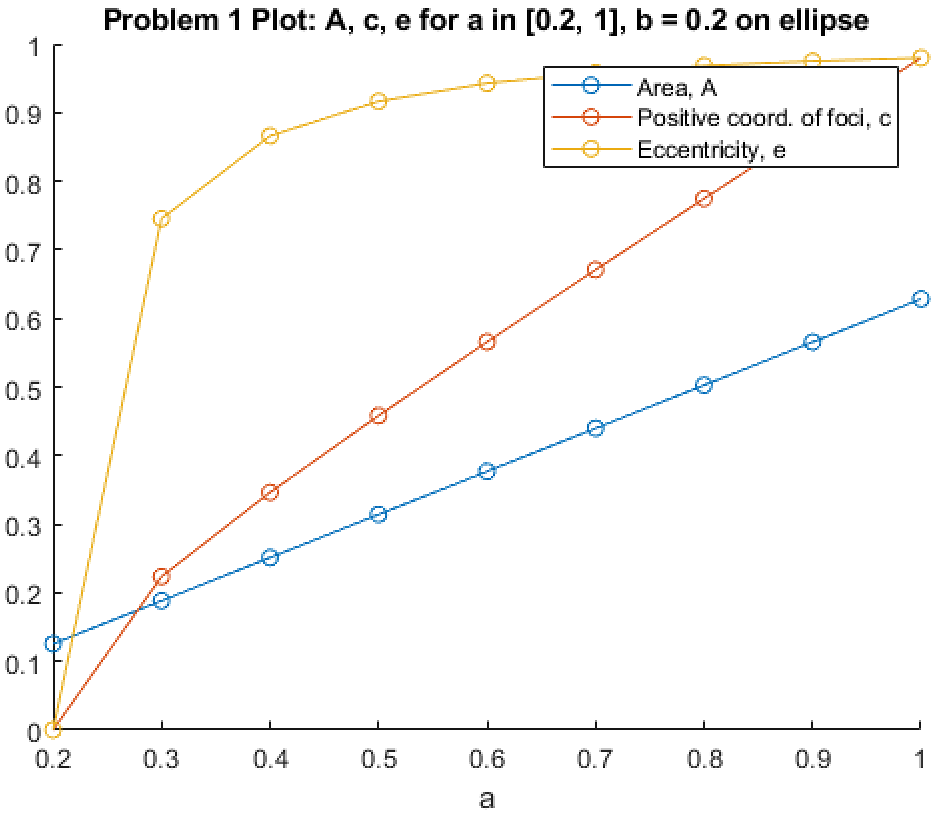
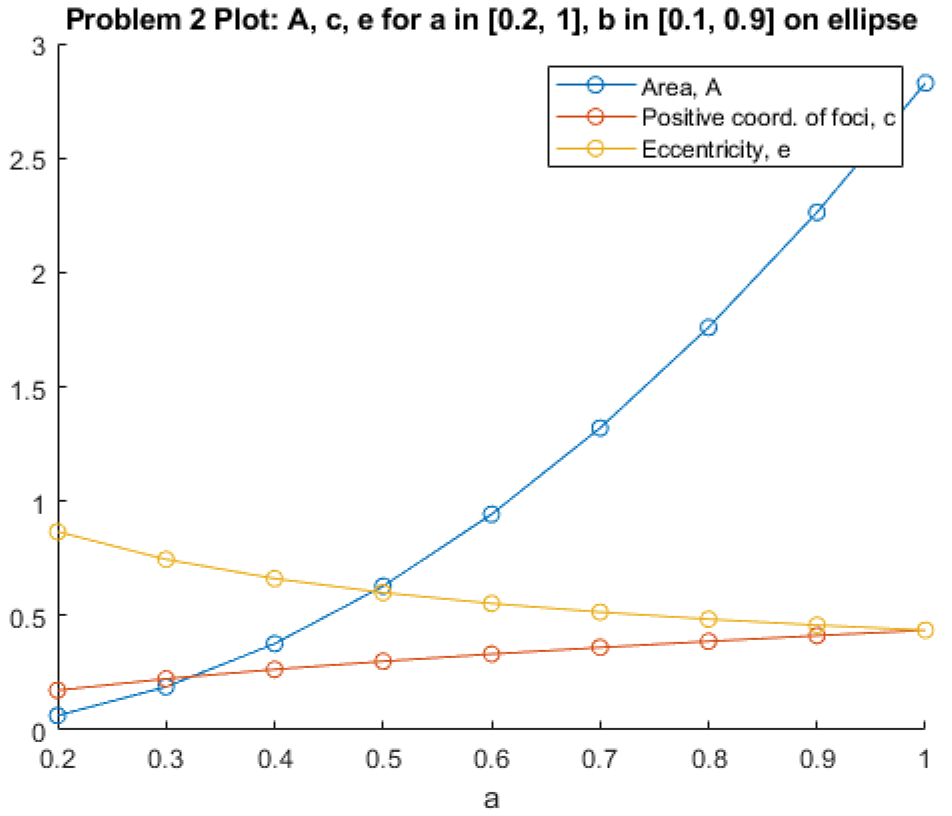
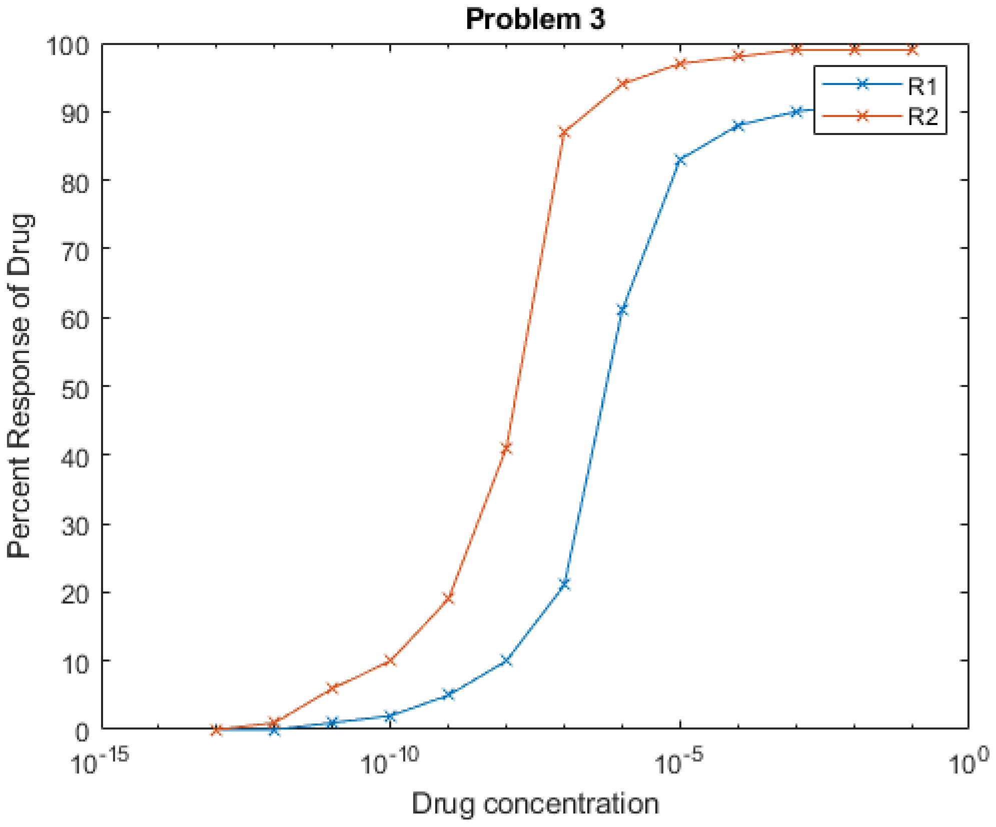
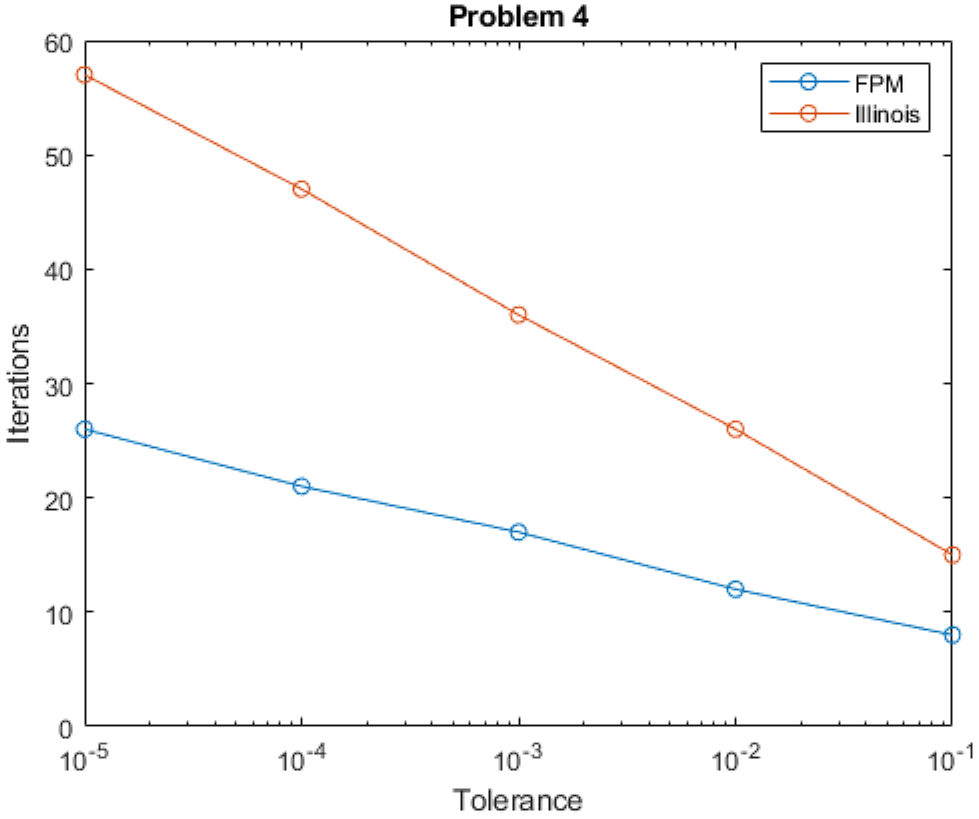
```
% Pierce Zhang, CMOR220, FALL 2023, Compentency on real roots
% competency_real_roots.m
% Answers to real roots competency
% Last modified: 16 September 2023

% Driver to answer the questions
function competency_real_roots
    problem1()
    problem2()
    problem3()
    problem4()
end

% Inputs: a and b, major axis and minor axis, respectively, of an ellipse
% centered at (0,0), OR vectors thereof
% Outputs: A, area, c, positive coordinate of the foci, e, eccentricity, of
% the ellipse centered at (0,0), OR vectors thereof
function [A,c,e] = ellipse(a,b)
    % Calculates the ellipse properties based on the value of a and b as
    % specified by the competency.
    A = pi .* a .* b;
    c = sqrt(a.^2 - b.^2);
    e = c./a;
end

% Inputs: none.
% Outputs: none.
function problem1
    % Declares a to be a vector from 0.2 to 1 with increment 0.1. Then,
    % graphs the ellipse properties of the ellipse formed from a, b against
    % a itself.
    a = 0.2:0.1:1;
    b = 0.2;

    [A,c,e] = ellipse(a,b);

    figure()
    hold on
    title("Problem 1 Plot: A, c, e for a in [0.2, 1], b = 0.2 on ellipse")

    plot(a, A, "-o"); plot(a, c, "-o"); plot(a, e, "-o");
    xlabel("a");

    legend("Area, A","Positive coord. of foci, c","Eccentricity, e")

end

% Inputs: none.
% Outputs: none.
function problem2
    % Declares a to be a vector from 0.2 to 1 with increment 0.1. Then,
    % graphs the ellipse properties against a. b is changing from 0.1 to
    % 0.9 in this case.
    a = 0.2:0.1:1;
    b = 0.1:0.1:0.9;

    [A,c,e] = ellipse(a,b);

    figure()
    hold on

    title("Problem 2 Plot: A, c, e for a in [0.2, 1], b in [0.1, 0.9] on
 ellipse")
    plot(a, A, "-o"); plot(a, c, "-o"); plot(a, e, "-o");
    xlabel("a");
    legend("Area, A","Positive coord. of foci, c","Eccentricity, e")

    % ANSWER TO QUESTION
    % The curves are different from those in the other problem because the
    % values of b are now shifting for each a instead of staying static at
    % 0.2, thereby causing different outputs when A, c, e are calculated.
    % Obviously mathematical results like c = sqrt(a^2 + b^2) will change when
    % the value of b changes, for the same a.
end

% Inputs: none
% Outputs: none
function problem3
    % This function plots 13 values of x from 10^-1 to 10^-13 along with
    % the predetermined R1 and R2 values. Values of x are on the x-axis and
    % R1 and R2 on the y-axis pointwise.
    figure()

    x = 10 .^ -(1:13);
    R1 = [92 91 90 88 83 61 21 10 5 2 1 0 0];
    R2 = [99 99 99 98 97 94 87 41 19 10 6 1 0];
    semilogx(x, R1, "-x");
    hold on
    semilogx(x, R2, "-x");
    hold on

    legend("R1","R2")
    xlabel("Drug concentration")
    ylabel("Percent Response of Drug")

    title("Problem 3")
end

% Inputs: f, the function, a, the left end of the domain, b, the right end
% of the domain, tol, the tolerance to consider a value to be close enough
% to the root
% Outputs: root, the estimated first root, and iter, the number of
% iterations it took to reach it
function [root, iter] = FPM(f,a,b,tol)
    % This function implements the false position method.
    c = (a*f(b) - b*f(a)) / (f(b) - f(a));
    iter = 0;
    while (abs(f(c)) > tol)
        if (f(a) * f(b) < 0)
            b = c;
        else
            a = c;
        end
        iter = iter + 1;
        c = (a*f(b) - b*f(a)) / (f(b) - f(a));
    end
    root = c;
end

% Inputs: f, the function, a, the left end of the domain, b, the right end
% of the domain, tol, the tolerance to consider a value to be close enough
% to the root
% Outputs: root, the estimated first root, and iter, the number of
% iterations it took to reach it
function [root, iter] = IA(f, a, b, eps)
    % This function implements the Illinois algorithm.
    c = (a*f(b) - (0.5*b*f(a))) / (f(b) - (0.5*f(a)));
    iter = 0;
 while (abs(f(c))) > eps
        iter = iter + 1;
  if f(a) * f(c) <= 0
   b = c;
  else
   a = c;
        end

        c = (a*f(b) - 0.5*b*f(a)) / (f(b) - 0.5*f(a));
 end
    root = c;
end

% Inputs: x
% Outputs: y
function [y] = f(x)
    % A simple construction of elementary functions
    y = cos(x) - 4*x^2;
end

% Inputs: none
% Outputs: none
function problem4
    % Generates a list of the number of iterations it took for either the
    % FPM and Illinois algorithms to come up with a tolerable root within
    % the domain 10^-1 to 10^-5 and plots it.
    f = @(x) cos(x) - 4*x^2;

    tol_list = 10.^-(1:5);
    iter_list_1 = [];
    iter_list_2 = [];
    for tol_i = tol_list
```

```
        [~,iter] = FPM(f,0,2,tol_i);
        iter_list_1 = [iter_list_1 iter];
        [~,iter2] = IA(f,0,2,tol_i);
        iter_list_2 = [iter_list_2 iter2];
    end

    figure()
    semilogx(tol_list, iter_list_1,"-o")
    hold on
    semilogx(tol_list, iter_list_2,"-o")
    hold on

    title("Problem 4")
    xlabel("Tolerance")
    ylabel("Iterations")

    legend("FPM","Illinois")
end
```




Problem 2 Plot: A, c, e for a in [0.2, 1], b in [0.1, 0.9] on ellipse
Area, A
Positive coord. of foci, c
Eccentricity, e
0
0.5
1
1.5
2
2.5
3
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
a


Problem 3
R1
R2
Percent Response of Drug
0
10
20
30
40
50
60
70
80
90
100
10^{-15}
10^{-10}
10^{-5}
10^{0}
Drug concentration



Published with MATLAB® R2023a