Student Information

Table of Contents

Student Name: Joaquin Arcilla

Student Number: 1007930820

Exercise 1

Objective: Observe a MATLAB error when you try to access an invalid index.

Details: Try to access elements of the vector `x`, created in the previous step at the command prompt `>>`. Try values of `i` outside the valid range. Try negative values, zero, and large positive values. Copy down the error message in each case, and report this in your submitted work as:

```
% Error for negative index:
% Array indices must be positive integers or logical values.
% Error in lab1 (line 74)
% disp(x(-1))

% Error for zero index:
% Array indices must be positive integers or logical values.
% Error in lab1 (line 76)
% disp(x(0))

% Error for positive index outside of the bounds:
% Index exceeds the number of array elements. Index must not exceed 6.
% Error in lab1 (line 80)
% disp(x(100))
% Only include the error message as a comment (with the percent signs),
% DO NOT INCLUDE THE CODE IN YOUR SUBMISSION.
```

Exercise 2

Objective: Create two vectors and plot one against the other.

Details: Explicitly create the vectors

x = [-5, -3, -1, 0, 2, 4, 6]

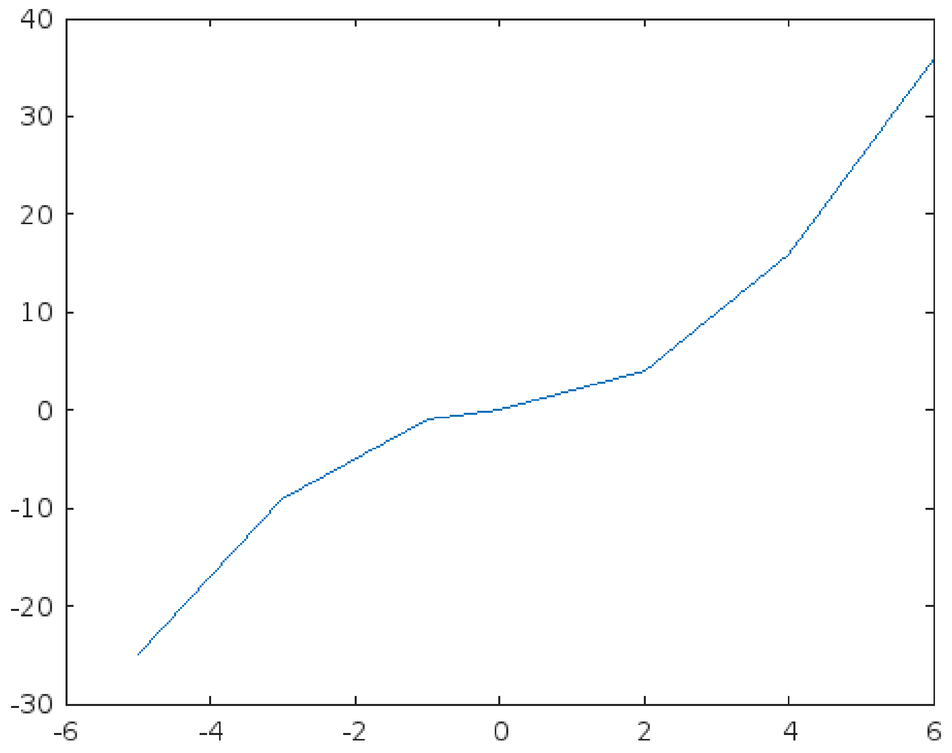
and

y = [-25, -9, -1, 0, 4, 16, 36]

And plot `x` against `y`. Your submission should show both the creation of the vectors and the figure, as in the example.

```
disp("Exercise 2")
x = [-5, -3, -1, 0, 2, 4, 6]
y = [-25, -9, -1, 0, 4, 16, 36]
plot(x, y)
```

```
Exercise 2

x =

    -5    -3    -1     0     2     4     6


y =

   -25    -9    -1     0     4    16    36
```



Exercise 3

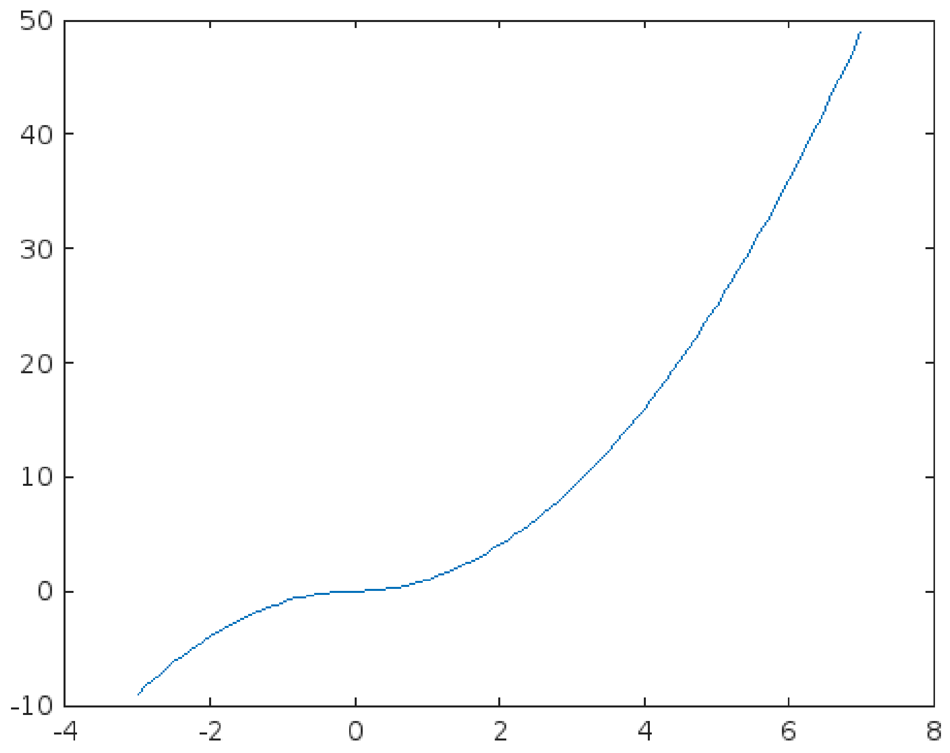
Objective: Plot x.x

Details: Using a for loop, create a vector x of 101 equally spaced points between -3 and 7, inclusive. Then create a vector y, such that `y(i) = abs(x(i))*x(i)` using a for loop. Plot the results.

Your submission should include the code, as in the example, and appropriate comments.

```
disp("Exercise 3")
% Mark values
N = 101;
a = -3;
b = 7;
h = (b - a)/(N-1);
x = zeros(1,N); % Preallocate x

% Create x
for j = 1:N
    x(j) = a + (j-1) * h;
end

% Get y values
y = zeros(1,N);

for j = 1:N
    y(j) = abs(x(j))*x(j);
end

% Plot results
plot(x,y)
```

```
Exercise 3
```



Exercise 4

Objective: Compare the speedup from vectorization for a larger vector.

Details: Create a vector x of 5000001 equally space points between -2 and 3, inclusive, using a for loop. Then time the construction of vector y, whose values are arctan of x, by a for loop and by vectorization. Be sure to clear y between the two runs. (Hint: use the help file to find the command for arctan!)

Your submission should include the code, as in the example, with appropriate comments.

```
disp("Exercise 4");
% Set up x
N = 5000001;
a = -2;
b =3;
h = (b - a)/(N-1);
x = zeros(1,N); % Preallocate x

% Create x
for j = 1:N
    x(j) = a + (j-1) * h;
end

% Time formation of Y: Loops
clear y;
disp("Loops:")
```

```
y = zeros(N,1);

tic;
for j = 1:N
    y(j) = atan(x(j));
end

toc;

% Time formation of Y: Vectorization
clear y;
disp("Vectorization:")
tic;
y = atan(x);
toc;
```

```
Exercise 4
Loops:
Elapsed time is 0.086633 seconds.
Vectorization:
Elapsed time is 0.010026 seconds.
```

Exercise 5

Objective: Solve an ODE and plot three particular solutions.

Details: Manually find the general solution of the ODE:

dy/dt + (2*t)*y = 1/exp(t^2)

and the particular solutions for initial conditions y(1) = -1, y(1) = 0, and y(1) = 1. Plot these three particular solutions on the same axes lettting the independent variable range from 1 to 5.

Once you have the three particular solutions you will need to:

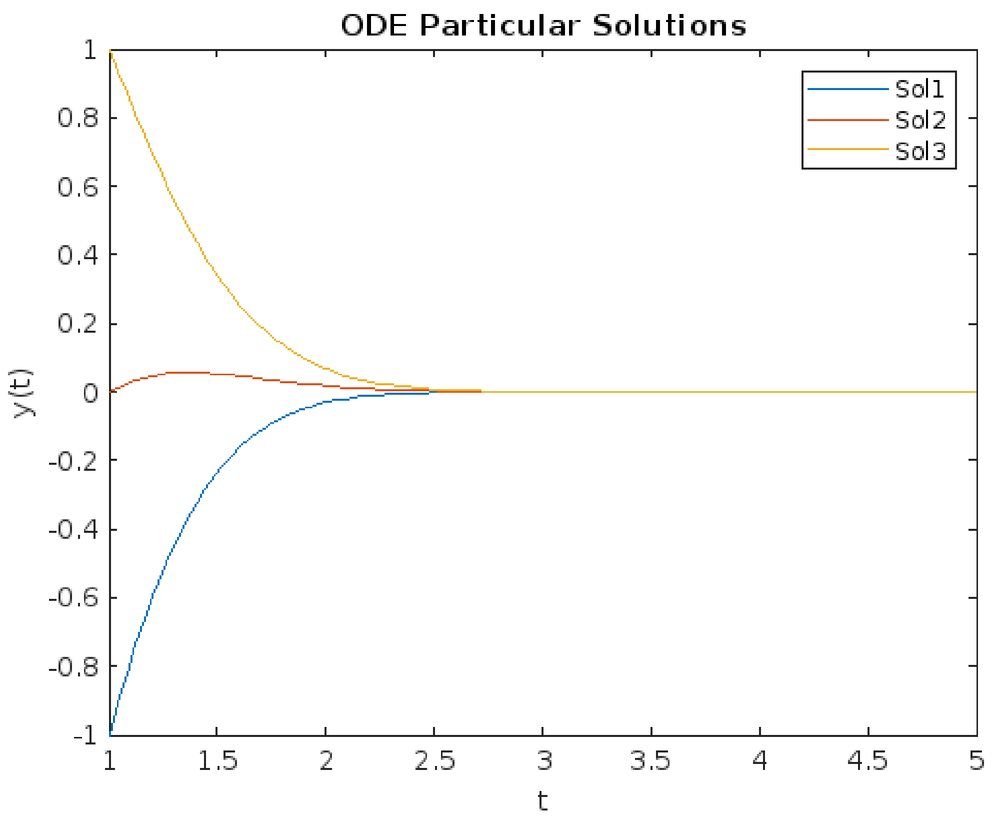
```
1. Create a vector for the independent variable, t.  Use linspace with
a reasonable number of points (100 is fine).
2. Create vectors for the three solutions.  Use vectorized operations.
3. Plot three solutions on the same axes, annotating the figure
 appropriately.
```

Your submission should include, in the description section, the general solution of the ODE and the three particular solutions. It should also have the appropriately commented code.

```
% Setting up graphs
disp('Exercise 5')
x = linspace(1, 5);
% Solutions:
% Gen sol: y = (t + c) / (e^t^2)
% y(1) = -1: c = -1 -e
% y(1) = 0: c = -1
% y(1) = 1: c = 1 + e
```

```
%Constants and c values
e_val = exp(1);
c1 = -1 - e_val;
c2 = -1;
c3 = e_val - 1;

% Solutions
func = @(t, c) (t + c) ./ (exp(t.^2));
sol1 = func(x, c1);
sol2 = func(x, c2);
sol3 = func(x, c3);

% Plot
plot(x,sol1,x,sol2,x,sol3);
xlabel('t');
ylabel('y(t)');
title('ODE Particular Solutions');
legend('Sol1', 'Sol2','Sol3', 'Location','NorthEast');
```

Exercise 5



Exercise 6

Objective: Write and use an inline function of one variable.

Details: Define the inline function for

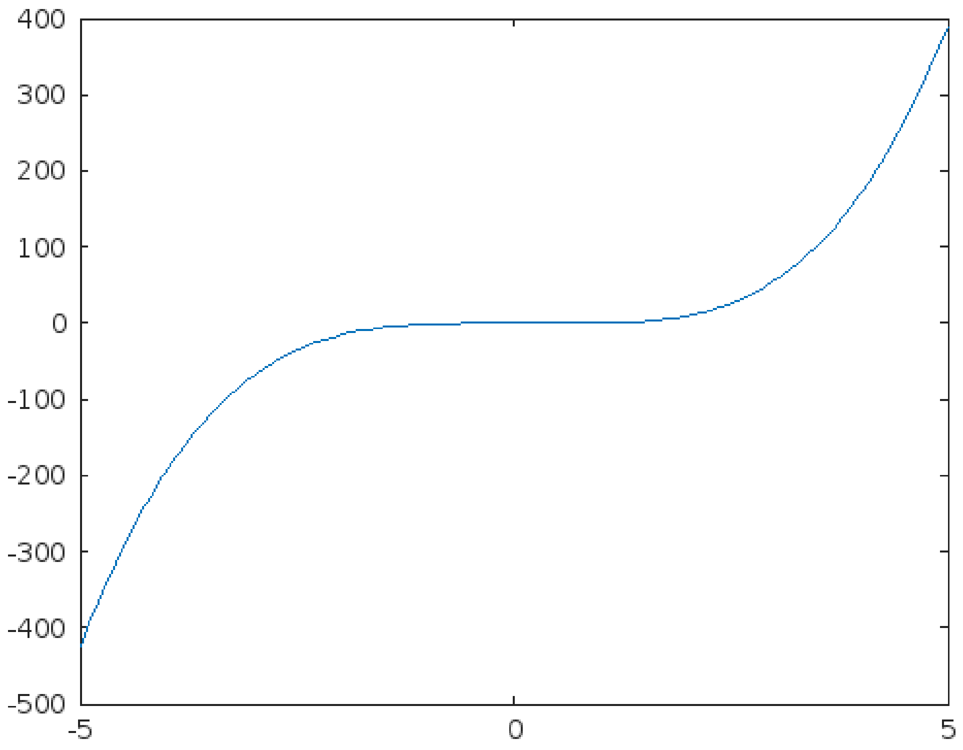
```
f(x) = (x^3 - abs(x)) * ln(x^2+1)
```

Use this function to compute `f(0)`, `f(1)`, and `f(-1)` and plot the function with 100 grid points from -5 to 5. Make sure to use vectorization in your definition, and label your axes.

Your submission should show the definition of the function, and the computation of these values.

```
disp('Exericse 6')
% Inline Function:
x = linspace(-5, 5, 100);
f = @(x) (x.^3 - abs(x)) .* log(x.^2 + 1);
% Display results
disp("f(0): " + f(0))
disp("f(1): " + (f(1)))
disp("f(-1): " + f(-1))
% Plot
plot(x,f(x))
```

```
Exericse 6
f(0): 0
f(1): 0
f(-1): -1.3863
```



Exercise 7

Objective: Write and use an inline function of two variables.

Details: Define the inline function for

```
f(x,y) = y + y^3 - cos x + x^2 - x^4
```

Note that `f(x,y) = c` is the implicit solution to

```
dy/dx = (4x^3 - 2x - sin x)/(1+3y^2).
```

See Examples 1-3 of section 2.1 in Brannan and Boyce for similar examples.

Use this function to compute `f(0, 0)`, `f(pi/2,1)`, and `f(-1,-1)`. Make sure to use vectorization in your definition.

Your submission should show the definition of the function, and the computation of these values.

```
disp("Exercise 7");
func = @(x,y) y + y.^3 - cos(x) + x.^2 - x.^4;
disp("f(0,0): " + func(0, 0));
disp("f(pi/2,1): " + func(pi/2, 1));
disp("f(-1,-1): " + func(-1, -1));
```

```
Exercise 7
f(0,0): -1
f(pi/2,1): -1.6207
f(-1,-1): -2.5403
```

Exercise 8

Objective: Use fzero to solve an equation `f(x,y)=0` for y at different values of x.

Details: Consider the function

```
f(x,y) = y + y^3 - cos x + x^2 - x^4
```

Define the appropriate inline function and compute the single solution at `x = 0` and the two positive solutions at `y = 1/2`.

You will need to experiment with different guesses to get these three solutions.

Your submission should show the definition of the function, and the computation of these three roots.

```
disp("Exercise 8");

%Set values
x0 = 0;
y0 = 1/2;
%Set functions
f = @(x,y) y + y.^3 - cos(x) + x.^2 - x.^4;
g = @(y) f(x0,y);
h = @(x) f(x, y0);

% For y
disp("For y:")
```

```
%Set guess 1
disp("Guess 1: guess = 1")
guess = 1;
y = fzero(g, guess);
disp("y = " + y);

%Set guess 2
disp("Guess 2: guess = 1000")
guess = 1000;
y = fzero(g, guess);
disp("y = " + y);

% I guess y = 0.6823
disp("y = " + y + " at x = 0")

% For x
disp("For x:")
% Guess 1
disp("Guess 1: guess = 1")
guess = 1;
x = fzero(h, guess);
disp("x = " + x)

% Guess 2
disp("Guess 2: guess = 10")
guess = 10;
x = fzero(h, guess);
disp("x = " + x)

% The first root must be 1.0584
disp("The first root must be " + x)

% Guess 3
disp("Guess 3: guess = 0.6")
guess = 0.6;
x = fzero(h, guess);
disp("x = " + x)
% Guess 4
disp("Guess 4: guess = 0.3")
guess = 0.3;
x = fzero(h, guess);
disp("x = " + x)

% The 2nd root must be 0.5764
disp("The second root must be " + x)
```

```
Exercise 8
For y:
Guess 1: guess = 1
y = 0.68233
Guess 2: guess = 1000
y = 0.68233
y = 0.68233 at x = 0
For x:
```

```
Guess 1: guess = 1
x = 1.0584
Guess 2: guess = 10
x = 1.0584
The first root must be 1.0584
Guess 3: guess = 0.6
x = 0.56741
Guess 4: guess = 0.3
x = 0.56741
The second root must be 0.56741
```

Exercise 9

Objective: Plot a portion of an implicit equation.

Details: Plot the portion of the solution to

`f(x,y) = y + y^3 - cos x + x^2 - x^4`

passing through the point `(0,0)` for x from -2 to 2. Use 100 grid points.

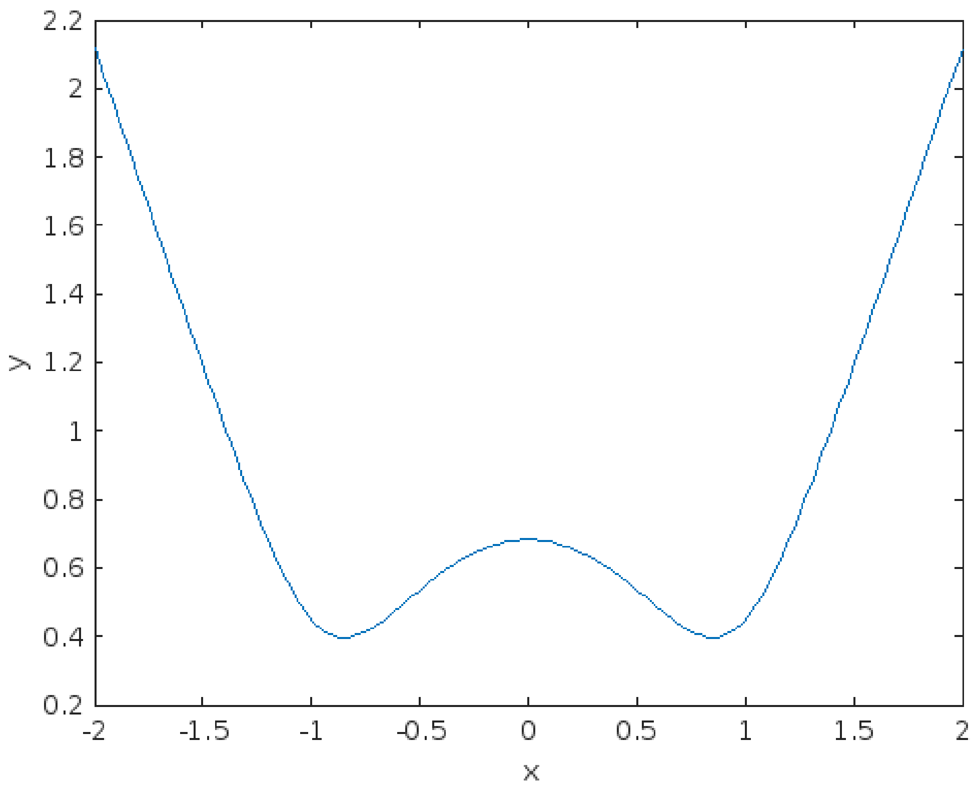
You will need to be careful with the initial guess you use in solving the equation.

Your submission should show the definition of the function, the construction of the arrays, the for loop, and the resultant figure.

Label your axes.

```
disp("Exercise 9");

% Set function
f = @(x,y) y + y.^3 - cos(x) + x.^2 - x.^4;

% Set x
xvals = linspace(-2, 2, 100);

% Preallocate y
yvals = zeros(size(xvals));

% Form Fzero solution
for j = 1:length(xvals)

   yvals(j) = fzero(@(y) f(xvals(j), y), 0);

end

% Plot the results

plot(xvals, yvals);
xlabel('x');
ylabel('y');
```

```
Exercise 9
```



Exercise 10

Objective: Solve a differential equation and plot a portion of it.

Details: Find the general implicit solution of the differential equation

```
dy/dx = (-cos x + 3x^2) y
```

Plot the particular solution passing through y(0) = 1 using 100 grid points for x in [-1.5, 1.25].

Be careful on your choice of guess. You will be penalized if MATLAB shows an error.

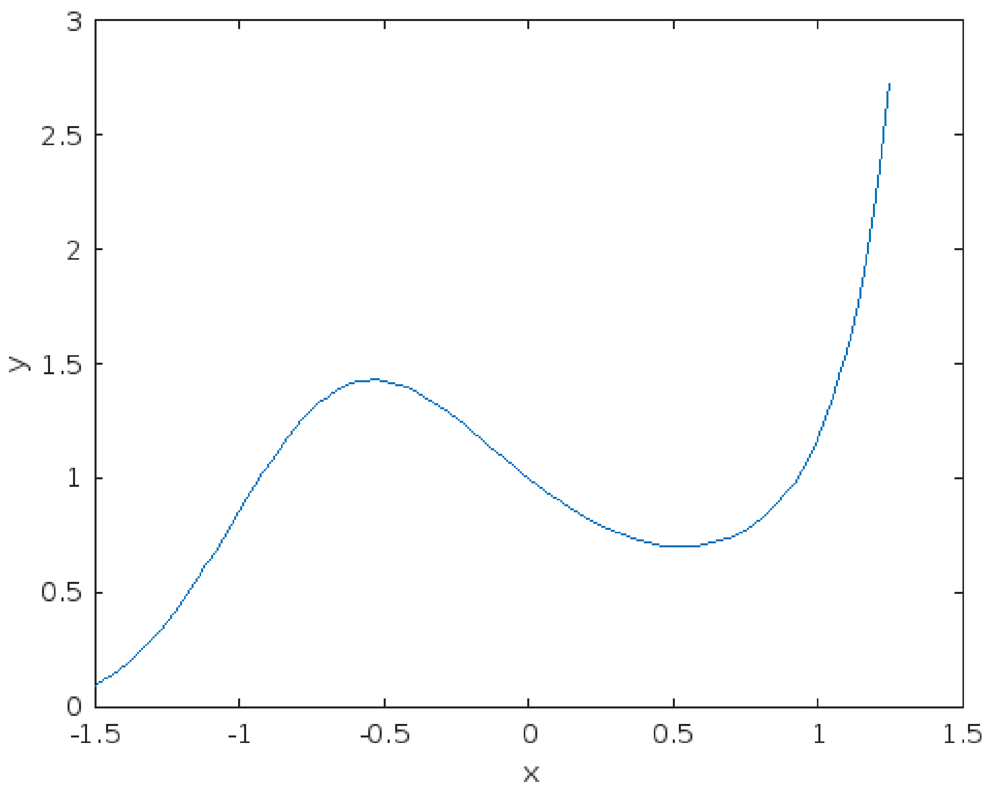
Your submission should show the general and particular solutions, in implicit form, in the comments, the definition of the appropriate inline functions, arrays, the for loop, and the figure.

Label your axes.

```
disp("Exercise 10");

% Gen solution y = exp(x^3 - sinx)+ c:
% Particular Solution y = exp(x^3 - sinx)
f = @(x,y) y - exp(x.^3- sin(x));

% Set x
xvals = linspace(-1.5, 1.25, 100);
```

```
% Preallocate y
yvals = zeros(size(xvals));

% Form yvals
for j = 1:length(xvals)

   yvals(j) = fzero(@(y) f(xvals(j), y), 0);

end

% Plot the results

plot(xvals, yvals);
xlabel('x');
ylabel('y');
```

Exercise 10



Install iode

Objective: install iode

Details: visit https://faculty.math.illinois.edu/iode/ and follow the instructions there to download and install iode.

Downloaded

Exercise 11

Objective: use iode a plot the direction field of a differential equation

Details: After starting iode, click the "direction fields" buttom from the main menu. The default is the direction field for

```
dy/dx = sin(y-x)
```

Click any point in the x-y plane to see a (numerically computed) solution to the differential equation passing through that point.

Starting from the "Equation" menu, plot the direction field of

```
dy/dx = xy^2
```

Describe, in words, the behaviour of the solutions. What happens as x approaches infinity? Explain why solutions are symmetric about the y-axis, i.e. `y(x)=y(-x)` for any solution.

```
% Answers
% Solutions with y > 0 make a half pipe shape at x = 0 and then
% go back towards infinity
% Solutions with y < 0 make a well shape at around x = 0 and then head
% Back towards y = 0
% As x approaches infinity, positive y converges to infinity while
% a negative y converges to 0
% It is symmetric about the y axis because the function uses a y^2
% function which is an even function and is also symmetrical across y axis.
```

Exercise 12

Objective: study the long-run behaviour of a differential equation from its direction field

Details: Using iode, plot the direction field of

```
dy/dx = y^4 - y^3 - 3*y^2 + y + 2
```

By clicking on various initial conditions `y(0) = y_0` (iode plots the solution), determine the long-run behaviour of solutions as a function of `y_0`. In other words, list the constant solutions that are approached as x goes to infinity and how `y_0` determines which constant solution is approached.

If your numerically computed solutions ever appear to cross then change the solution method from Euler to the, much better, Runge-Kutta method. Think about why solutions are not permitted to cross. If they do then this indicates a problem with the numerical solver.

```
% Constant Solutions: as x approches infinity
% y approaches 1 when -1 < y_0 < 2
% y approaches -1 when y_0 <= -1

% While x approaches -ve infinity, y can also approach 2
```

Published with MATLAB® R2022b