Instructions on what to turn in:

Download the folder lab2files to your machine. Inside you will find the following files:

- lab2_lastname.mlx (rename this file to include your last name)
- fpi.m (you will need this code to work on problem 3)

What you will need to turn in on Canvas. Upload the following 2 files:

- lab2_lastname.pdf (instructions on how to generate this below)
- lab2_lastname.zip (a zip folder containing bisection.m, fpi.m, lab2_lastname.mlx)

How to generate lab2_lastname.pdf.

Once you have completed all the parts of the assignment and are satisfied with your work, go **View** in the top bar and click **Output Inline** - this makes it so that the grader can easily follow your work. Run lab2_lastname.mlx one more time then go to **Live Editor** and click **Export** then choose PDF. A pdf file with the same name as the Live script will be generated with answers to all the parts.

Specific instructions for the assignment:

Problem 2 depends on successful completion of Problem 1. To make your work easy to grade, please suppress all outputs except for the parts that explictly say not to suppress output. Your code for Problems 2 and 3 should go in the grey boxes.

This is a **live script** like you saw in the OnRamp tutorial. A couple of pointers:

- Ignore Orange warnings that have to do with suppressing output or not using certain variables. Do not ignore red warnings!
- If you accidentally delete a code block (in grey), you can re-add it by going to **Insert** and selecting **Code**.
- You can also add text by clicking **Insert** and select **Text**.
- To clear all outputs, right click on anywhere in the script and choose **Clear All Output.** Alternatively to clear output for a specific part, go the code block that generated the output, right click and choose **Clear Output.**
- You can run each problem/section individually by selecting **Live Editor** and **Run Section** while you're in the section
- To generate the pdf at the end, run the whole file

Problem 1

In a separate file called `bisection.m` copy and paste the following starter code to create a Matlab function that implements the Bisection Method. Make sure to suppress any outputs using semicolons.

```
function [p,nit] = bisection(f,a,b,Nmax,tol)
%BISECTION
% This program solves the scalar-valued equation f(x)=0 using the
% bisection method to a given tolerance accuracy
```

```
%
% INPUTS:
%         f - Root-finding function defined as an anonymous Matlab function.
%         a - Left end point of interval containing solution
%         b - Right end point of interval containing solution
%      Nmax - Maximum number of iterations (an integer)
%       tol - Solution tolerance
% OUTPUTS
%         p - Solution to the problem
%     nit - Number of iterations it takes to get convergence
%
% Written by
% MATH3043 @ Temple University
% Fall 2022

% Translate the Bisection Algorithm pseudocode on page 49 of
% Burden, Faires, Burden's 10th edition Numerical Analysis to complete this program.
%%%% YOUR CODE goes here %%%%%%
end
```

Test your code with the following problem to find $f(x) = \cos x - x = 0$ in the interval $[0, 1]$ to a tolerance of 10^{-6}. The answer approximate to six correct digits is $p = 0.739085$.

```
format long
f = @(x) cos(x) - x;
[p,nit] = bisection(f,0,1,50,1e-6)
```

```
p =
   0.739085197448730
nit =
    19
```

Problem 2

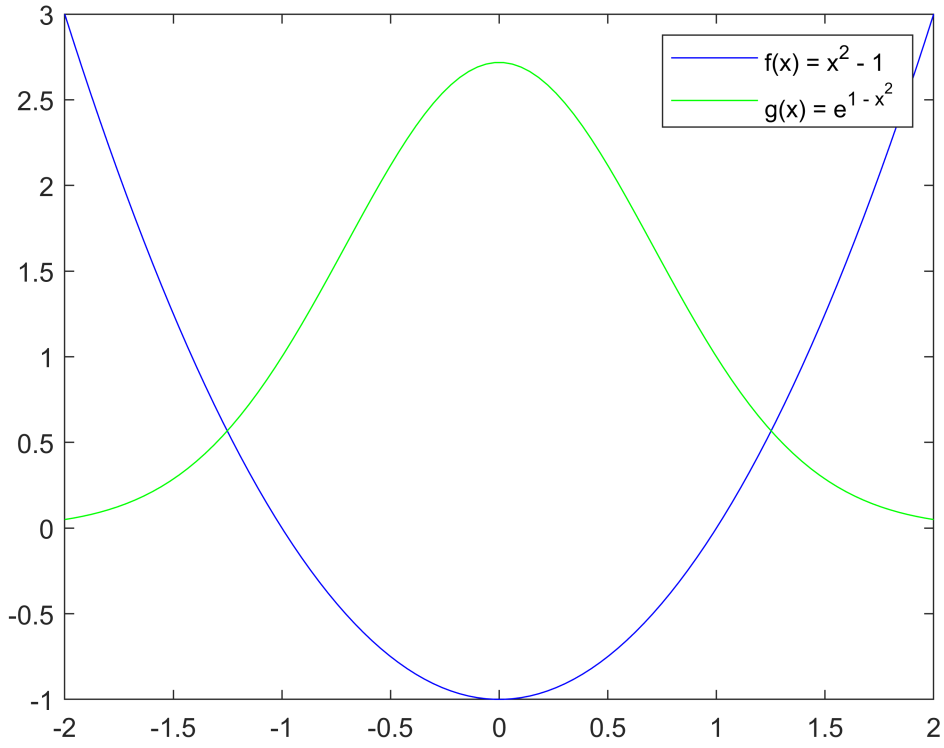
The following are Matlab anonymous function definitions for $f(x) = x^2 - 1$ and $g(x) = e^{1-x^2}$:

```
f = @(x) x.^2 - 1;
g = @(x) exp(1-x.^2);
```

a) Include code below to plot these functions on one graph in the interval $[-2, 2]$ with 100 points. Use a blue line for f and a green for g. Add a legend. Feel free to refer back to the OnRamp tutorial to review plotting

```
% Plotting routine here
x_axis = linspace(-2, 2, 100);
f_y = arrayfun(f, x_axis);
g_y = arrayfun(g, x_axis);
```

```
%just now realized i could have done f(x), g(x)
plot(x_axis, f_y, 'b', x_axis, g_y, 'g');
legend('f(x) = x^2 - 1','g(x) = e^{1 - x^2}');
```



b) Define two intervals $[a1, b1]$ and $[a2, b2]$ of length two where each of the points of intersection of f and g lie. Intervals can intersect but there should be only one point of intersection per interval.

For example if one intersection point is in $[-3, -1]$ and the other in $[3, 5]$, define the following variables

```
a1 = -3
b1 = -1
a2 = 3
b2 = 5
```

```
% Definitions of a1,b1,a2,b2
a1 = -1.5;
b1 = -1;
a2 = 1;
b2 = 1.5;
```

c) Define an anonymous function h where $h(x) = 0$ gives the roots of the intersection points of the graphs of f and g. Make sure the function can take vector valued inputs.

```
% Definition of h(x)
h = @(x) f(x) - g(x);
```

d) The search interval for the solution to $h(x) = 0$ is $[-2, 0]$. Define a and b below to correspond to the left and right endpoints respectively :

```
% Define a and b
a = -2;
b = 0;
```

e) How many iterations are required to find an approximation to $h(x) = 0$ to within 10^{-3} in the interval $[-2, 0]$? Name this variable $Nmax$ and define it below

```
% Define Nmax
Nmax = 1000;
```

f) Define a variable named tol with a value of 10^{-3}

```
% Define tol
tol = 10.^(-3);
```

g) Find the solution to $h(x) = 0$ within $[-2, 0]$ using your Bisection Method implementation from Problem 1 and variables $a, b, Nmax, tol$ defined above

(Do not suppress output)

```
% Call to bisection code, Remember not to suppress output here
[p, nit] = bisection(h, a, b, Nmax, tol)
```

```
p =
  -1.251953125000000
nit =
     9
```

Problem 3

```
clear % clears the workspace variables for a new problem
```

The following four fixed point problems $(g_k(x) = x)$ were proposed to compute $21^{\frac{1}{3}}$:

- $g_1(x) = \dfrac{20x + \dfrac{21}{x^2}}{21}$
- $g_2(x) = x - \dfrac{x^3 - 21}{3x^2}$
- $g_3(x) = x - \dfrac{x^4 - 21x}{x^2 - 21}$
- $g_4(x) = \left(\dfrac{21}{x}\right)^{1/2}$

a) Define anonymous functions that correspond to each of these problems

```
% Define g1, g2, g3, g4
g1 = @(x) (20*x + ( (21)/(x.^2) )) / 21;
g2 = @(x) x - ((x.^3 - 21)/(3*(x.^2)));
g3 = @(x) x - ((x.^4 - 21*x) / (x.^2 - 21));
g4 = @(x) sqrt(21 / x);
```

b) Define the following variables $p0 = 1, Nmax = 200, tol = 10^{-8}$

```
% Define p0, Nmax, tol
p0 = 1; Nmax = 200; tol = 10.^(-8);
```

c) For each of the g_k's, use the given function `fpi.m` to compute the fixed point iteration with the variables defined above. *(Do not suppress outputs).* For example assuming g1 is defined above

```
% Code for fpi with g1
[p,nint] = fpi(p0,g1,Nmax,tol)   % Here ninit is number of iterations it takes to converge
```

```
% Run fpi with g1, (Do not suppress output)
[p,nint] = fpi(p0,g1,Nmax,tol)
```

```
p =
   2.758924124121746
nint =
   106
```

```
% Run fpi with g2 (Do not suppress output)
[p,nint] = fpi(p0,g2,Nmax,tol)
```

```
p =
```

```
   2.758924176381121
nint =
     8
```

```
% Run fpi with g3 (Do not suppress output)
[p,nint] = fpi(p0,g3,Nmax,tol)
```

```
p =
     0
nint =
     2
```

```
% Run fpi with g4 (Do not suppress output)
[p,nint] = fpi(p0,g4,Nmax,tol)
```

```
p =
   2.758924173773539
nint =
    30
```

d) Do all the methods converge to a reasonable solution? Rank the g_k's according to their observed speed of convergence (fastest to slowest).

Your answer below (as text):

Rankings (Fastest to Slowest/no Convergence):

1. g_2
2. g_4
3. g_1
4. g_3 (converages to the *wrong* fixed point)

e) Find the derivatives to each of the g_k's and define anonymous functions $f_k(x) = g_k'(x)$.

```
% Define anonymous functions for the derivatives : f1, f2, f3, f4
f1 = @(x) (20 - ( 42 / x^.3)) / 21;
f2 = @(x) (2.0/3) - (14/x.^3);
f3 = @(x) 1 - ((2*x.^5 - 84*x.^3 + 21*x.^2 + 441) / (x.^2 - 21).^2);

c = (-1.0/2) * sqrt(21);

f4 = @(x) c*(x.^(-3.0/2));
```

f) Evaluate each f_k at $p = 2.7589241$, the exact solution of $21^{\frac{1}{3}}$ to 8 significant digits. *(Do not suppress outputs)*

```
p= 2.7589241
```

```
p =
   2.758924100000000
```

```
f1(p)
```

```
ans =
  -0.522673557695755
```

```
f2(p)
```

```
ans =
    -5.537022385038171e-08
```

```
f3(p)
```

```
ans =
   5.705587200751359
```

```
f4(p)
```

```
ans =
  -0.500000020763834
```

g) What is the observed relation between the absolute value of the derivative at the fixed point $p = 2.7589241$ and your ranking of the speed of convergence above?

Your answer (as text):

The closer the derivative is to 0 the less iterations it takes for the fixed point approxmation to converge to the desired value where $g(p) = p$. This phenomenon is likely to the orders of convergence in Chapter 2.4.