ENG1060: COMPUTING FOR ENGINEERS

Lab 7 – Week 8

2020 OCT NOV

Welcome to lab 7. This is the first lab for Part B. Remember that laboratories continuously build on previously learned concepts and lab tasks. Therefore, it is crucial that you complete all previous labs before attempting the current one.

Self-study:
Students are expected to attempt these questions during their own self-study time, prior to this lab session. There may be questions that require functions not covered in the workshops. Remember to use MATLAB's built-in help for documentation and examples.

Learning outcomes:
1. To revise user-defined functions and apply good programming practices
2. To identify the mathematics that represents the problem to be solved
3. To differentiate between closed (bracketed) and open root-finding methods
4. To summarise the requirements and limitations of each root-finding method
5. To apply root-finding methods to solve equations by hand and with MATLAB

Background:
Engineers frequently need to solve equations. For example, solve x in $f(x) = ax^2 + bx + c = 0$. Here, the values of x that satisfy the equation are known as the roots to the equation. Simple equations can be solved by rearranging for the unknown variable. However, this may not be possible for more complicated equations and therefore we require root-finding techniques to approximate the roots.

Primary workshops involved:
- Workshop 3: Functions, commenting, debugging and strings
- Workshop 7: Roots and optimisation

Assessment:
This laboratory comprises **2.5%** of your final grade. The questions are designed to test your recollection of the workshop material and to build upon important programming skills. You will be assessed on the quality of your programming style as well as the results produced by your programs during your laboratory session by the demonstrators. Save your work in **m-files** named ***lab1t1.m***, ***lab2t2.m**,* etc. Inability to answer the demonstrator's questions will result in zero marks, at the demonstrator's discretion.

Team tasks begin at the start of the lab session so please ensure you arrive on time to form your groups. Students who arrive late will not be able to participate in the team tasks as teams will have already formed and will therefore forfeit all associated marks. These tasks will be assessed during class.

Lab submission instructions

Follow the instructions below while submitting your lab tasks.

Team tasks:
The team tasks are designed for students to test and demonstrate their understanding of the fundamental concepts specific to that lab. These tasks will occur at the start of the lab and will be assessed on the spot. Demonstrators will advise on how these will be conducted. Most team tasks do not require the use of MATLAB but MATLAB should be used for checking purposes.

Individual tasks:
The individual tasks are designed for students to apply the fundamentals covered in the team tasks in a variety of contexts. These tasks should be completed in separate m-files. There is typically one m-file per task unless the task requires an accompanying function file (lab 3 onwards). Label the files appropriately. E.g. lab6t1.m, lab6t2.m, eridium.m, etc.

Deadline:
The lab tasks are due next Friday at 9am (MYT) or 12pm (AEDT). Late submissions will not be accepted. Students will need to apply for special consideration after this time.

Submission:
Submit your lab tasks by:

1) Answering questions in Google Form, and
2) Submitting one .zip file which includes all individual tasks.

The lab .zip file submission links can be found on Moodle under the weekly sections, namely Post-class: Lab participation & submission. The submission box ("Laboratory 7") will only accept one .zip file. Zipping instructions are dependent on the OS you are using.

Your zip file should include the separate m-files for the individual tasks including function files.

It is good practice to download your own submission and check that the files you have uploaded are correct. Test run your m-files that you download. You are able to update your submission until the deadline. Any update to the submission after the deadline will be considered late.

Grade and feedback:
The team will endeavour to grade your lab files by Tuesday of the following week. Grades and feedback can be viewed through the Moodle Gradebook, which is available on the left side pane on the ENG1060 Moodle site.

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

Lab 7 – Assessed questions

TASK 1

[2 MARKS – L07TE]

Note: Team tasks are designed for students to recall material that they should be familiar with through the workshops and practice of the individual questions prior to this lab session.

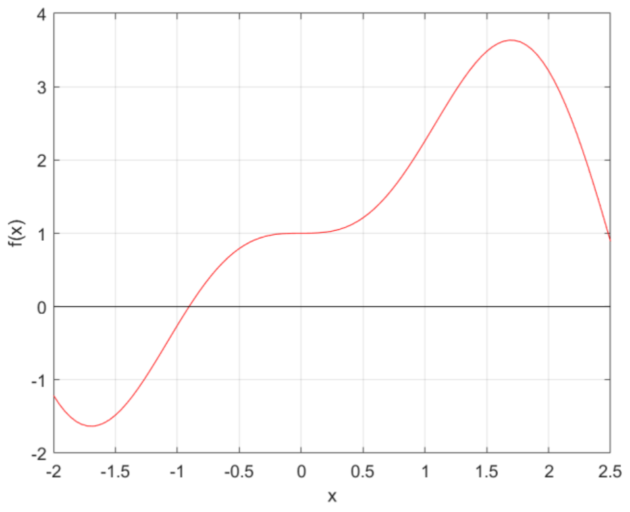
Students will be split into groups of 3-4 for the team tasks. Students in each group must explain aspects of the question below to receive the marks. Ensure that everyone has equal learning opportunities. Additionally, ask your table for help.

$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$	$x_{i+1} = x_i - \frac{\delta f(x_i)}{f(x_i + \delta) - f(x_i)}$	$x_R = \frac{x_L + x_U}{2}$
$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$	$x_R = \frac{f(x_U)x_L - f(x_L)x_U}{f(x_U) - f(x_L)}$	

<u>Root-finding methods:</u>

Each group will be assigned <u>ONE bracketed</u> and <u>ONE open</u> root-finding method from the list below.

Consider $f(x) = \sin(x) - x\cos(2x) + 1$ as shown on the right.



<u>Bracketed method</u>

- Bisection method with $x_L = -2$ and $x_U = 2$
- False-position method with $x_L = -1.5$ and $x_U = 2$

<u>Open method</u>

- Newton-Raphson method with $x_i = 0.5$
- Secant method with $x_{i-1} = -1.5$ and $x_i = 2$
- Modified secant method with $x_{i-1} = 1.4$ and $\delta = 0.1$

Complete the following:

1. Copy the plot for each root-finding method into the slide.
2. Type the root-finding method and the corresponding equation. List the starting parameters.
3. Illustrate two iterations of the root-finding method. Draw circles to indicate initial guess(es) and estimated root (visual approximation, do not calculate) for the first iteration. Use squares for the second iteration. Label the points with the appropriate variables (x_L, x_U, x_i, etc.) and draw a line connecting these points (only if it is not the bisection method)
4. Type the function header declaration and while *condition* (just the condition, not the entire loop!) for the root-finding method
5. Discuss the task and explore any misunderstandings. Also, browse the work of other teams related to the other Codes and ensure that you have understood it as concepts from all sets may be required for the individual tasks.
6. Have a demonstrator assess your understanding

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

Remember good programming practices for all tasks even if not specifically stated. This includes, but is not limited to:

- using clc, close all, and clear all, where appropriate
- suppressing outputs where appropriate
- labelling all plots, and providing a legend where appropriate
- fprintf statements containing relevant answers

PRELIMINARY

Create function files implementing each of the following root-finding methods following the function declarations given. This will help you consolidate your understanding of the techniques involved.

• Bisection method	function [root, iter] = bisection(f,xl,xu, precision)
• False-position method:	function [root, iter] = falseposition(f,xl,xu, precision)
• Newton-Raphson method:	function [root, iter] = newraph(f,df,xi, precision)
• Secant method:	function [root, iter] = secant(f,xi,xi_1, precision)
• Modified-secant method:	function [root, iter] = modisecant(f,xi,pert, precision)

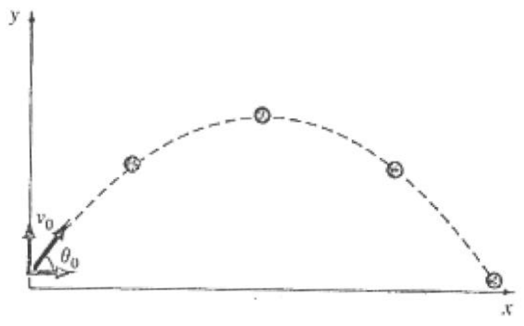
TASK 2

[3 MARKS]

Aerospace engineers sometimes compute the trajectories of projectiles such as rockets. A related problem deals with the trajectory of a thrown ball. The trajectory of a ball is defined by the (*x*,*y*) coordinates as shown in the figure below. The trajectory model can be modelled as:

$$y = \tan(\theta)\,x - \frac{g}{2v_0^2\cos^2(\theta)}x^2 + y_0$$

where θ_0 is the throwing angle in degrees, v_0 is the initial velocity, y_0 is the thrower's elevation, x is the distance from the thrower to the catcher and y is the catcher's elevation, g is the gravitational acceleration (g=9.81 ms^{-2}).



A. Use the secant method to find the required throwing angle given v_0 =20m/s, y_0=2m, to reach a catcher at point x=35m, y=1m. Use θ_i = 15 and θ_{i-1} = 60 with a precision of 1e-4. Use fprintf to print the initial throwing angle required and the number of iterations taken.

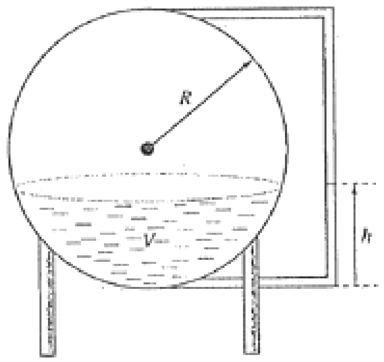
B. Also, plot the root-finding equation against the initial throwing angles of $0 \le \theta \le 60$. Mark the estimated root with a black diamond.

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

TASK 3

[4 MARKS]

You are designing a spherical tank of radius *R*=3m (see figure below) to hold water for a small village in a developing country.



The volume of liquid contained in the tank can be expressed as

$$V = \pi h^2 \frac{3R - h}{3}$$

where *h* is the height of the water in the tank and *V* is the corresponding volume of water. You are interested in determining the height of the water in the tank which will give a total volume of 30m^3. The support of tank will become unstable or collapse if the tank has water height of more than 5m.

A. Plot the root-finding equation to be solved using a black continuous line. Print this equation in the title of the plot. *Hint: think about what the independent and dependent variables are in this problem to determine what variables to plot*

B. Use the Newton—Raphson method to solve this problem. Prompt the user for the initial guess *xi* and use a precision of 1e-4. Mark this root as an upward triangle on the previous figure.

C. Print a statement containing the height of the water required, and the corresponding number of iterations taken for the root finding method to converge. Example:
The depth of water required to fill the tank with 30m^3 of water is ???m, and it took ??? iterations to solve using the Newton-Raphson method.

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

TASK 4

[4 MARKS]

The specific growth rate *g* [day^{-1}] of a yeast bacterial culture that produces an antibiotic is a function of food concentration, *c* [mg/L] according to

$$g(c) = \frac{2c}{4 + 0.8c + c^2 + 0.2c^3}$$

Growth goes to zero at very low concentration due to food limitation. It also goes to zero at high concentration due to toxicity effects.

Write an m-file to determine the value of *c* at which the growth is maximum using the false position method. For your calculation use an initial guess of x_l=0 mg/L, x_u=5 mg/L and a precision of 1e-4. Plot the following three graphs in one figure to verify that your answer is correct. Ensure you label your plots.

- Plot *g* against *c*, where *c* ranges from 0 to 10.
- Plot the derivative of *g* against *c* for the same range above.
- Plot a vertical line at the solution found by false position method

Write a statement which describes the value of the concentration when the growth is a maximum, and the number of iterations it took to obtain the value.

Note: The derivative of *g*(*c*) is

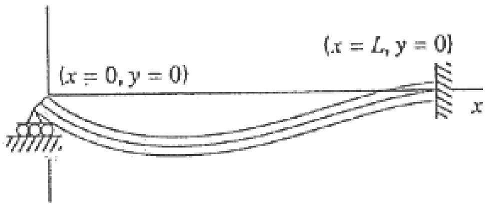
$$g'(c) = \frac{10(20 - 5c^2 - 2c^3)}{(c+5)^2(c^2+4)^2}$$

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

TASK 5

[4 MARKS – L07H]

Consider a beam of length L subject to a distributed load w_0.



The equation for the resulting elastic curve is

$$y = \frac{w_0}{120EIL}(-x^5 + 2L^2x^3 - L^4x)$$

where y is the deflection, E is the modulus of elasticity of the material and I is the moment of inertial.

Use the Newton-Raphson method to determine the point of maximum deflection to a precision of 10^{-4} and the value of the maximum deflection by finding the root of the derivative of the function given $L = 600$ cm, $E = 50{,}000$kN/cm^2, $I = 30{,}000$cm^4 and $w_0 = 2.5$kN/cm.

Plot the following three graphs in one figure to verify that your answer is correct. Ensure you label your plot.

- Plot y against x, where x ranges from 0 to 600 cm.
- Plot the derivative of y against x for the same range above.
- Plot the maximum deflection point and observe that the derivative of the gradient is 0 at the point

Write a statement that includes the maximum deflection value and the corresponding x value. An example output is shown below.

```
        The maximum deflection of -x.xxcm occurs at xxx.xxcm
```

2 marks deducted for poor programming practices (missing comments, unnecessary outputs, no axis labels, inefficient coding, etc.)

END OF ASSESSED QUESTIONS

The remainder of this document contains supplementary and exam-type questions for extended learning. Use your allocated lab time wisely!

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

Lab 7 – Supplementary questions

These questions are provided for your additional learning and are not assessed in any way. You may find some of these questions challenging and may need to seek and examine functions that are not taught in this unit. Remember to use the help documentation. Coded solutions will not be provided on Moodle. Ask your demonstrators or use the discussion board to discuss any issues you are encountering.

TASK 1S

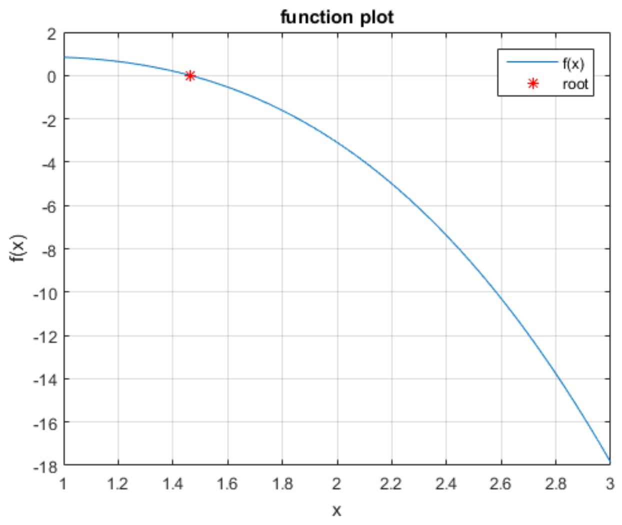
Use the bisection method to determine the roots of the function

$$f(x) = x^2 - x^3 + \sin(x)$$

between *x*=1 and 3. Use a precision of 0.001. Plot this function and mark the zero point with a red asterisk. Use the fzero() function to check your answer.

SOLUTION

```
root_fzero =     1.4639450223
root_bisection =     1.4638671875
percentage error =     0.0053167873%
```



Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

TASK 2S

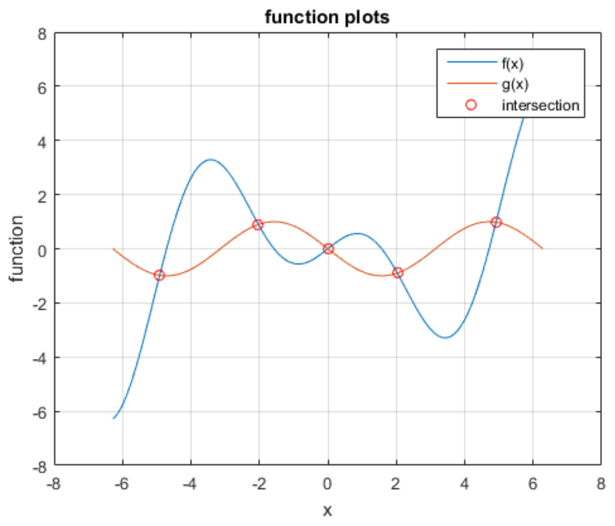
Find the intersecting points of the following functions

$$f(x) = x\cos(x)$$
$$g(x) = -\sin(x)$$

between -2π ≤ x ≤ 2π using the false-position method. Choose appropriate arguments for the false-position method. Plot the functions to identify by eye where the intersecting points are located, then plot the intersections as red circles on the same plot.

SOLUTION

```
xr = -4.9131804394 for bracket [-6 -4]
xr = -2.0287578381 for bracket [-3 -1]
xr = 0.0000000000 for bracket [-1 1]
xr = 2.0287578381 for bracket [1 3]
xr = 4.9131804394 for bracket [4 6]
```



Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

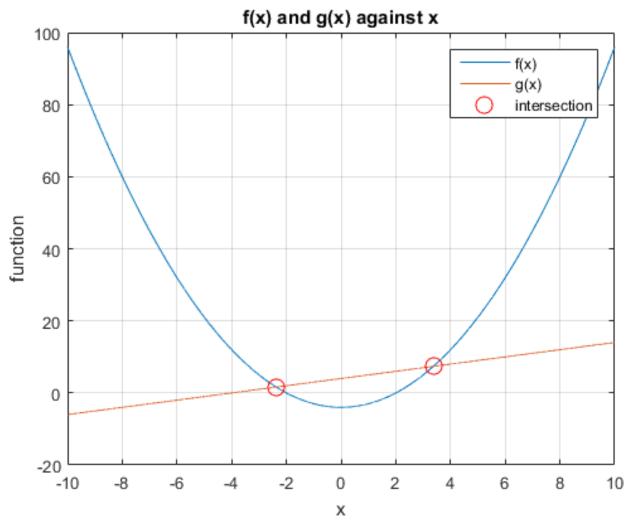
TASK 3S

Consider these two functions

$$f(x) = x^2 - 4$$
$$g(x) = x + 4$$

Plot the two functions, remembering to label your plot and include a legend. Use the Newton-Raphson method to determine all intersecting points of these two functions. Use a precision of 0.001. Mark the intersecting points with markers of size 10. Use the fzero function to check your answers.

SOLUTION



```
       root_NR       root_fzero            error
 -2.3724137931  -2.3722813233   0.0055840694%
  3.3722813742   3.3722813233   0.0000015104%
```

10 **Important:** If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

TASK 4S

For the function

$$f(x) = x^{10} - 2$$

determine whether the bisection method or the false-position method converges faster using a lower limit of x=0 and an upper limit of x=2. Use a precision of 0.001. Use the fzero function to check your root answer.

SOLUTION

```
    Method     N_iter                  xr     error
 bisection         12          1.07177734     0.0004%
  falsepos        821          1.07172075     0.0049%
     fzero                     1.07177346
```

TASK 5S

If an amount of money *P* is invested for *k* years at a nominal annual interest rate *r* (expressed as a decimal fraction), the value *V* of the investment after *k* years is given by

$$V = P\left(1 + \frac{r}{n}\right)^{nk}$$

where *n* is the number of compounding periods per year. Take *P*=1000 and *r*=4%.

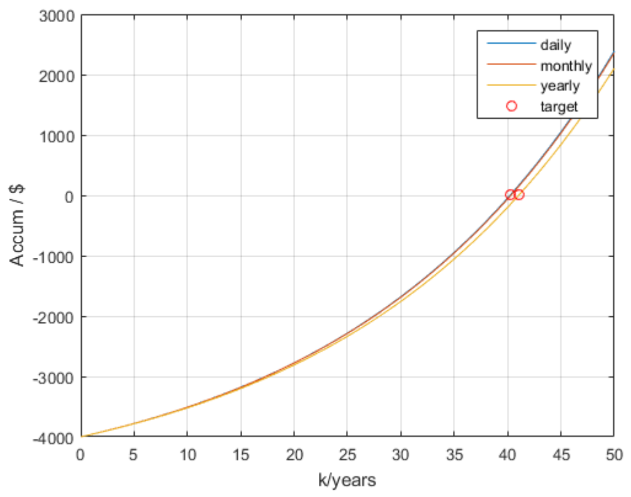
Determine how long it takes (to the nearest year) to accumulate $5,000 using daily (365 days in a year), monthly (12 months in a year) and yearly compounding periods. Use any root-finding method of your choice with a precision of $100.

SOLUTION

```
365 compounding periods take 40 years to
save the given amount.

12 compounding periods take 40 years to
save the given amount.

1 compounding periods take 41 years to save
the given amount.
```



Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

Lab 7 – Exam-type questions

These questions are provided for your additional learning and are not assessed in any way. You may find these type of questions on ENG1060 exams. Solutions will not be provided on Moodle. Ask your demonstrators or use the discussion board to discuss any issues you are encountering. Additionally, you may use the exam collaboration document on Moodle (under the exam section) to share your answers.

1. Which of the following root-finding methods are bracketed methods?
    a. False-secant
    b. False-position
    c. Newton-Raphson
    d. Modified-secant
    e. Secant

2. Which of the following statements are true for open methods?
    a. Requires the original, first and second derivative functions.
    b. Will always find a root when given an initial guess
    c. Requires 2 inputs instead of the first derivative
    d. Calculates the gradient (estimated or with a derivative function) to find the root
    e. Initial guess is required to be negative

3. What is the purpose of the precision parameter for root-finding functions?
    a. Defines the global truncation error
    b. Tolerance to check if the function evaluated at the root is the approximate value
    c. Defines how many decimal places the initial guesses have
    d. Changes the variable type from ‘single’ to ‘double’ precision
    e. The sum of all the errors from every root-finding iteration

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

4. Root-finding methods can be used to approximate the intersections of 2 different functions, $f(x)$ and $g(x)$. Given:

$$f(x) = x^2 - 5x - 6$$

$$g(x) = 2x - 10$$

a) Use an initial guess of $x_L = 5$, and $x_U = 10$ and the false-position method to determine the first 2 iterations to estimate one of the two roots.

Iteration:	x_L	x_U	x_R
0	5	10	
1			
2			

b) Provide the syntax for fzero to determine the intersections of the two functions, with an initial guess of 7. The help fzero documentation is provided as a reference

```
fzero  Single-variable nonlinear zero finding.

    X = fzero(FUN,X0) tries to find a zero of the function FUN near X0,     if X0 is a
scalar.  It first finds an interval containing X0 where the function values of the
interval endpoints differ in sign, then searches that interval for a zero.  FUN is a
function handle.  FUN accepts real scalar input X and returns a real scalar function
value F, evaluated at X. The value X returned by fzero is near a point where FUN changes
sign (if FUN is continuous), or NaN if the search fails.
```

```
% Define h(x) where f = g
hfunc = @(x1) x1.^2 -7*x1 +4;
% Use fzero to find root
Xr1 =
```

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.

c) Plot the f(x) and g(x) functions. Mark the intersection points (xr1, xr2) on the plot with blue circles. Add a legend, positioned north.

```
% define F(x1) and G(x1) as anonymous functions
F =
G =
% Define “X” domain as 100 equally spaced points from 10 to 10
X =


% Plot the F and G functions and the intersection points at Xr1 and Xr2
figure;





% label axes, add title and legend positioned at the north of the plot
```

d) One of your colleagues asks for help on defining the while condition wihin their root-finding function. Provide the while condition for the root guess “Xr” and “precision” variable

```
% while condition to iterate and find root
while
```

Important: If you are struggling with a task, ensure that you have performed hand-written work (e.g. hand calculations, pseudocode, flow charts) to better understand the processes involved. Do this before asking demonstrators for help and use it to assist with your illustration of the problem.