ENG1060: COMPUTING FOR ENGINEERS

Lab 6 – Week 7

2020 OCT NOV

Welcome to lab 6. This lab is designed to consolidate your learning in Part A to date. 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 loops and continued application of debugging skills
2. To identify programming structures and in-built functions appropriate for the task
3. To formulate code involving loops and IF statements to solve complex problems

Background:
Engineers do not want to perform repetitive tasks! Leave that to the computers. Loops are used to repeat blocks of code either for a specified number of times or until a specified condition is met. The implementation of loops greatly increases the number of calculations being performed and therefore, increases the potential for bugs to be introduced. Debugging tools are used to help resolve buggy code.

Primary workshops involved:

- Workshop 4: Input, output and IF statements
- Workshop 5: Loops and debugging
- Workshop 6: Loops, advanced functions and MATLAB limitations

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.

There is no team task for this lab. Instead, there is a consolidation quiz for Part A on Moodle worth 2.5%. Check Moodle for details.

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 6") 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 6 – Assessed questions

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

TASK 1

[2 MARKS – L06L]

You should now be familiar with MATLAB's built-in "max" function. Without using the built-in max(), min() and sort() functions, write a function that takes vector as an input and returns the maximum value as well as the index at which the maximum value occurs. Your function should work with input vectors of any length. Your function header should be similar to the following:

```
function [max_value, index] = mymax(x)
```

1. Test your function for $f(x)=x^{(1/x)}$ for x values between 0 and 50 with a step size of 0.001.
2. Check the maximum value and index using MATLAB's built-in max() function.
3. Plot $f(x)$ and mark the maximum point. Remember to include a legend.
4. Use fprintf to print a statement containing the maximum value and the index value.

*Although not necessary for this task, think about how you would code this for a two-dimensional matrix by creating a function which finds the maximum value and its corresponding row and column.

```
function [max_value, row, column] = mymax2(x)
```

TASK 2

[2 MARKS – L06P]

The **numbers.txt** file contains 10,000 randomised integers. You have chosen the 10 lucky numbers as follows:
[2, 220, 389, 725, 768, 241, 36, 953, 531, 161].

You would like to see how many times the lucky numbers appear in the numbers.txt file.

Write an m-file that

A. Imports the integers from the numbers.txt file.
B. Determine the number of times each lucky number appears in the numbers.txt file. Then calculate the total number of times lucky numbers appear.
C. Use fprintf to print the total number of times the lucky numbers appear in the numbers.txt file and the lucky number that appears most frequently.

Example output:
Lucky numbers appear X times and number Y appears most frequently

 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

[2 MARKS – L06Q]

In neural network application, activation function f is adopted in the hidden layer to perform nonlinear transformation on input x and generate output y as

$$y = f(x)$$

Subsequently, any elements of y which are less than 0.5 would be assigned as 0 value. For instance, if $y = [0.1\ 0.4\ 0.8\ 0.7\ 0.2\ \dots]$, then $y_{drop} = [0\ 0\ 0.8\ 0.7\ 0\ \dots]$.

a) Create a function file that produces the $ydrop$ of hidden layer. The function file should take inputs of a function handle and the x. Use the function declaration as shown below:

```
function [ydrop] = activ_drop(fhandle, x)
```

b) Create function handles for the 4 activation functions below and store them in a cell array using fset = {f1;f2;f3;f4}. Then use a for loop to plot the y_{drop} against x in a 2x2 subplot arrangement by using $-3 \leq x \leq 3$ with increment of 0.01 in these cases and set the axes limits to be 'equal'. The titles can be generated in the for loop using title(char(fset{<index>}).

i. $f1 = \frac{1}{1+e^{-x}}$

ii. $f2 = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

iii. $f3 = ln(1 + e^{x})$

iv. $f4 = e^{-x^2}$

TASK 4

[2 MARKS – L06M]

A square wave of period $2T$ may be defined by the following function:

$$f(t) = \begin{cases} 1\ (0 \leq t \leq T) \\ -1\ (-T \leq t < 0) \end{cases}$$

A. Plot the square wave function $f(t)$ as a red line assuming $T = 1$ with increments of 0.0001.

The Fourier series for the periodic square wave function $f(t)$ is given by

$$F(t) = \frac{4}{\pi}\sum_{k=0}^{\infty}\frac{1}{2k+1}\sin\left[\frac{(2k+1)\pi t}{T}\right]$$

From Wolfram: A Fourier series is an expansion of a periodic function f(x) in terms of an infinite sum of sines and cosines. Fourier series make use of the orthogonality relationships of the sine and cosine functions.

See here for an animation: http://www.jezzamon.com/fourier/

B. Prompt the user for the maximum k value (since we cannot compute for infinite k terms). Then calculate and plot the Fourier series $F(t)$ between $t = -1$ and 1 with a step size of 0.0001 as a blue line on the same figure from step A. Remember to include a legend located in the northwest. Also, include a title which states how many terms were used for the Fourier plot.

 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

[2 MARKS – L06I]

An engineer is looking to minimise the growth of a pollutant that contains α and β protein chains. The growth of the pollutant is described by

$$p = \sum_{n=1}^{n_{\max}} \left(\alpha^{-n} + n \cdot |\cos(\beta)|^{\sin(\alpha)}\right)$$

Notice that the $|X|$ here represents the absolute value of *X*. Consider the case where $n_{max} = 50$, α ranges from 10 to 25 (increments of 1) and β ranges from 5 to 30 (increments of 1). Determine the smallest *p* value and the corresponding α and β. Use fprintf to print the relevant values.

Hint: You may need to use two min functions (i.e. min(min(X))) to find the minimum value within a matrix or use the find() function.

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 6 – 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

The product of three consecutive integers is 778596. Determine what these three integers are.

SOLUTION

```
Product = 778596 for consecutive integers 91, 92 and 93.
```

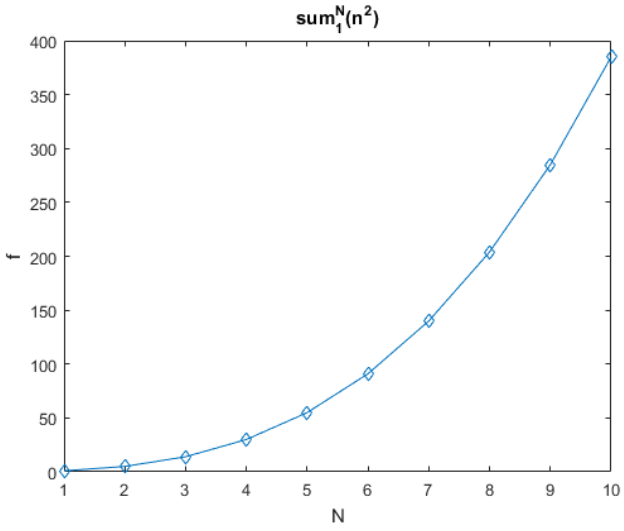
TASK 2S

Calculate the following sum using a for loop with *N*=1 to *N*=10:

$$\sum_{n=1}^{N} n^2$$

Plot the sums as a function of *N*.

SOLUTION



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

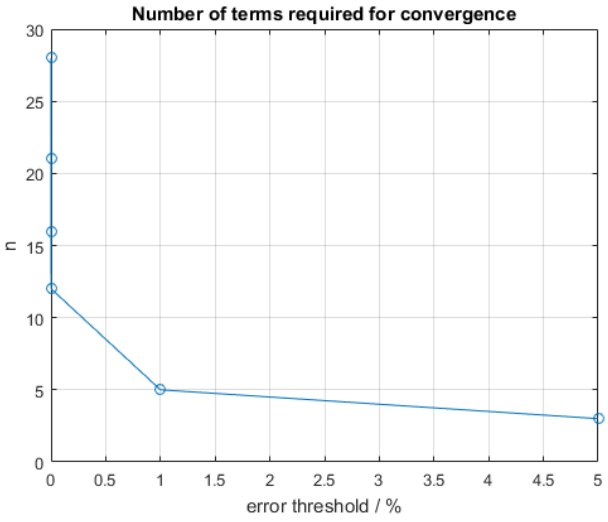
Determine how many terms are required such that the absolute percentage error between one iteration and the next is less than 5%, 1%, 1e-3%, 1e-5%, 1e-7% and 1e-10%.

$$\sum_{n=0}^{\infty} e^{-n}$$

Plot the number of iterations required against the error threshold.

SOLUTION

err_threshold	N	rel error	f_val
5.0E+00%	3	3.21E+00%	1.55300179278
1.0E+00%	5	4.27E-01%	1.57805537866
1.0E-03%	12	3.88E-04%	1.58197313108
1.0E-05%	16	7.11E-06%	1.58197664138
1.0E-07%	21	4.79E-08%	1.58197670643
1.0E-10%	28	4.37E-11%	1.58197670687



 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

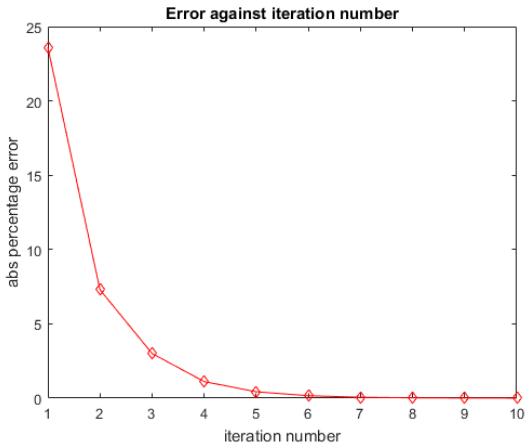
The ratio $(1+\sqrt{5})/2$ can be approximated using the following steps:

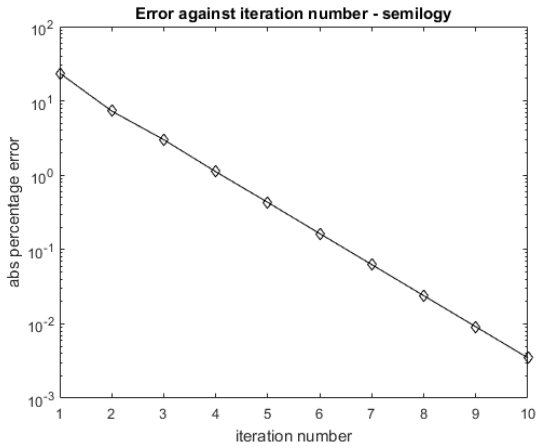
- Start with 1 plus 1
- Take 1 divided by that result and add 1 (e.g. 1+ 1/(1+1))
- Continue to take 1 divided by the previous result (e.g. 1+ 1/(1+ 1/(1+1))

Write a code to calculate this approximation after 10 iterations, taking 1+1 to be the first iteration. For each iteration, calculate the absolute percentage error. Plot the absolute percentage error against iteration number as red diamonds. On a new figure, plot the absolute percentage against the iteration number again, but with black diamonds on a logarithmically scaled y-axis (look up the semilogy function).

SOLUTION

n	f_approx	f_analytical	percentage error
1	2.0000000000	1.6180339887	23.6067977500%
2	1.5000000000	1.6180339887	7.2949016875%
3	1.6666666667	1.6180339887	3.0056647916%
4	1.6000000000	1.6180339887	1.1145618000%
5	1.6250000000	1.6180339887	0.4305231719%
6	1.6153846154	1.6180339887	0.1637402789%
7	1.6190476190	1.6180339887	0.0626457976%
8	1.6176470588	1.6180339887	0.0239135846%
9	1.6181818182	1.6180339887	0.0091363613%
10	1.6179775281	1.6180339887	0.0034894607%





 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 5S

Write a script using loops to produce the following output. Ask the user for a maximum number of iterations (*n*). The script should iterate from 1 to *n* to produce the expressions on the left, perform the specified operation to get the results shown on the right, and print using the format shown (for *n*=9). Use fprintf to produce the statements.

1 x 8 + 1 = 9
12 x 8 + 2 = 98
123 x 8 + 3 = 987
1234 x 8 + 4 = 9876
12345 x 8 + 5 = 98765
123456 x 8 + 6 = 987654
1234567 x 8 + 7 = 9876543
12345678 x 8 + 8 = 98765432
123456789 x 8 + 9 = 987654321

SOLUTION

```
Enter maximum number of iterations : 20
1 x 8 + 1 = 9
12 x 8 + 2 = 98
123 x 8 + 3 = 987
1234 x 8 + 4 = 9876
12345 x 8 + 5 = 98765
123456 x 8 + 6 = 987654
1234567 x 8 + 7 = 9876543
12345678 x 8 + 8 = 98765432
123456789 x 8 + 9 = 987654321
1234567900 x 8 + 10 = 9876543210
12345679011 x 8 + 11 = 98765432099
123456790122 x 8 + 12 = 987654320988
1234567901233 x 8 + 13 = 9876543209877
12345679012344 x 8 + 14 = 98765432098766
123456790123455 x 8 + 15 = 987654320987655
1234567901234566 x 8 + 16 = 9876543209876544
12345679012345676 x 8 + 17 = 98765432098765424
123456790123456800 x 8 + 18 = 987654320987654400
1234567901234568192 x 8 + 19 = 9876543209876545536
12345679012345679872 x 8 + 20 = 98765432098765438976
```

 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 6 – 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 statements is true regarding the code execution speed when using loops?
    a. Increasing the value of the iteration counter increases the execution speed
    b. Pre-allocating known variable sizes prior to the loop increases the execution speed
    c. Appending values in each iteration increases the execution speed
    d. Adding if statements within the loops increases the execution speed
    e. Using nested loop systems instead of vector operations increases the execution speed

2. Which of the following statements describe the function of elseif statements?
    a. Begins a new if statement loop
    b. Runs if all other conditions are not satisfied
    c. Returns a logical value to the command window if the loop is completed
    d. Performs alternative logic check if the preceding conditions have failed
    e. Transforms logical operators into short-circuit operators

3. Which of the following statements is **true** about function files?
    a. Cannot run nested loops within it
    b. Function files cannot call other function files
    c. Can directly call a sub-function from outside the main function
    d. Has its own internally referenced workspace for variables
    e. Has a maximum of five inputs/outputs

For questions 4 and 5, the following code is provided to you:

```
for ii=1:40
    m(ii) = ii/2;
   for jj = 1:3:5
        n(ii,jj) = m(ii)+jj;
        for xx = 5:-3:-10
            p = xx*n(ii,jj).*m;
        end
    end
end
```

 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. What is the value of n(20,5) ?
    a. NaN
    b. 15
    c. 10
    d. Error: Index exceeds matrix dimensions.
    e. 11

5. What is the size of p when ii=30 and jj=1?
    a. 1 x 1
    b. 1 x 4
    c. 1 x 40
    d. 1 x 30
    e. 1 x 5

6. As an acoustics engineer, you want to check the sound pressure levels (SPLs) at specific frequencies from within a speaker enclosure, ENG1060 Industries' latest product. A colleague has provided you with experimental data at particular frequencies, which range from 20 to 20 kHz. For the sound levels at these frequencies, you want to assign a status of:
    - 0 (to represent the sound is damped) if the SPL is less than 30 dB
    - 0.01*sound level (a tuning constant) if SPL is between 30 and 50 dB
    - 1 (for undamped) if SPL is equal to or greater than 50 dB

    You are given data for 2 different speaker enclosures. The resolution (spacing between each frequency data point) from experiments is not always consistent, and integer frequency values are NOT guaranteed.

    The experimental data is provided to you in two files. The first contains the frequencies that were measured. The second contains the SPL magnitudes for each of those frequencies (in two columns, corresponding to each speaker). There are no headers in each of the file (numeric data only).

    Frequencies.txt - column vector of the frequencies sampled, in *Hertz*

19.99
20.08
20.20
…
19 999.80
19 999.89
20 000.01

    SoundlevelData.txt - matrix of SPL measured in *decibels* (dB) for each enclosure (separate columns)

11	13
34	35
62	59

    …

    e.g. Speaker 1 has a sound level of 11 dB at a frequency of 19.99 Hz.

11 **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.

A colleague has informed you of four primary frequencies that are known to have resonance issues, at 25, 50, 100 and 1000 Hz. You have decided to pre-assign these frequencies into a vector below.

```
FreqChecks = [25 50 100 1000]; % frequencies to check in Hertz
```

Provide the missing code required for each of the following steps of the commented code.

```
% load experimental data using importdata() function and append them into one matrix consisting of two columns
F  = importdata('Frequencies.txt')        % load frequencies

ExpSL  = importdata('SoundlevelData.txt')  % load SPL data
```

```
% pre-allocate an indexing storage matrix named "FreqInd" that is the same size as FreqChecks
FreqInd = zeros(1,length(FreqChecks));
```

```
noSpeakers = 2; % no. of speaker enclosures to examine

% pre-allocate a status matrix called "statusSpeaker" that has the following size:
% rows = number of speakers
% columns = number of elements in FreqChecks
statusSpeaker = ones(noSpeakers,length(FreqChecks)
```

```
% for loop statement and iteration counter for each frequency of interest in FreqChecks
for ii = 1:length(FreqChecks)
```

```
% for loop statement and iteration counter for elements in noSpeakers
  for nn = 1:noSpeakers
```

```
% use find function to extract a single frequency entry that is larger than the current frequency of interest, then store its index into "FreqInd". i.e. to get 20 Hz, the next closest value is 20.02 Hz
```

Portion of the "help find" documentation

```
I = find(X,K) returns at most the first K indices corresponding to the nonzero entries of the array X.  K
must be a positive integer, but can be of any numeric type.

I = find(X,K,'first') is the same as I = find(X,K).

I = find(X,K,'last') returns at most the last K indices corresponding to the nonzero entries of the array
X.
```

```
% find the index for the current frequency of interest

FreqInd(ii) = find(F>FreqChecks(ii),1);
```

 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.

```
% Assign a status based on the SPL magnitude, using the conditions provided

        % if statement to check sound level is less than 30 dB
    if ExpSL(FreqInd(ii),nn) < 30


        % assign damped status to a variable called statusTemp
       statusTemp = 0;

% secondary condition to check if sound level is less than 50 dB
    elseif ExpSL(FreqInd(ii),nn) <= 50

% calculate tuning constant, and assign to statusTemp
        statusTemp =

% all other powers, status = 1;
    else

            statusTemp = 1    % assign status to 1, undamped
        end

% assign statusTemp value to matrix statusSpeaker, with corresponding index
    statusSpeaker(   ,   ) =

  end
end
```

13 **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.

7. The procedure to calculate the approximate value of the square root of a number, N, is equivalent to constructing a square with the area N. This is referred to as the Babylonian method.

   You start with an approximate guess of the square root, $x0$. For simplicity, we will assume $x0 = N/1.5$ (this would take much longer for large numbers). This corresponds to a rectangle with sides x0 and N/x0.

   However, we are looking to converge on a square. If the equation x0 = N/x0 is true, the length of the sides of the rectangle would become equal. i.e. a square.

   We can improve the estimate of x0, by averaging value of the two sides where:

$$x1 = \frac{(x0 + N/x0)}{2}$$

   By repeating this procedure, it is possible to get a value which converges towards the true value of $\sqrt{N}$. The incomplete program shown below will give an approximate value for the square root of N, to an accuracy of 1×10^{-5}. The function MySquareRoot.m is shown below.

```
1  function x1 = MySquareRoot(N)
2  x0=N/1.5;            %initial guess
3  tol=input('Provide an accuracy: ');
4  dx= x0-N          %absolute error
5                       %iteration counter
6
7  while                %condition to check absolute value of error is larger than tolerance
8                       %formula to calculate x1, new guess of sqrt(N)
9                       %calc. absolute error between old (x0) and new value (x1)
10 m=m+1;              %increase counter
11                      %reassign newly calculated estimate as old estimate for next iteration
12 end
```

a) Complete the code missing at the specified lines:

Line 5	m = 0;
Line 7	while abs(dx) >tol
Line 8	x1 = (x0+N/x0) /2;
Line 9	x1-x0
Line 11	x0=x1;

 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.

b) To track the convergence of the code, you have decided to print the outputs for each iteration and the current guess of √N to 5 decimal places at line 12. Provide the syntax below:

```
fprintf('Estimate of sqrt(N): %0.5f, Iteration: %g',x0,m)
```

c) You decide to store the values for all the revised guesses of $\sqrt{N}$ and realise lines 8 to 11 will need to be changed, to use the counter variable **m** as the index. Provide the code for each of these lines below.

Line 5	x1(m+1) = (x0+N/x0) /2;
Line 7	dx = x1(m+1)-x0;
Line 8	m=m+1;
Line 11	x0 = x1(m);

d) Provide the syntax that calls the previous function for the number of elements in "Numbers", a vector with values you wish to calculate the square root.

```
clear all; close all; clc
% Calculate square root of numbers from 100 to 500 in increments of 40
% Create Numbers vector – provided to you
Numbers = 100:40:500;

% pre-allocate square root results matrix
sqrtNumbers = zeros(size(Numbers));

% use a for loop to call the function and store the estimated square root values into a vector named: sqrtNumbers
for ii=1:length(Numbers)
sqrtNumbers(ii) = MySquareRoot(Numbers(ii))
end
```

 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.