Technical computing with Matlab – exercises

Version November 20, 2023

Some of the exercises are based on or from the book *MATLAB: a practical introduction to programming and problem solving* by Stormy Attaway (3rd edition).

Contents

Logbook

Note: more information on your logbook can be found in the lecture notes and the logbook examples on Blackboard.
To help you with your logbook, you can copy-paste the commands entered in MATLAB to a Word document. This is the most straightforward way. Copy-paste is done by selecting the command with your mouse, pressing CTRL+C, switch to your Microsoft Word document, and pressing CTRL+V. Note that you can create an online Word document via `https://onedrive.lincoln.ac.uk`. Don't forget to *logout* at the end of the session.
You can also create screen shots. In windows 10 this is done by pressing WINDOWSKEY+SHIFT+S, then select an area you would like to copy, and subsequently paste it in Word. Make sure the text is large enough to read in the Word document. Alternative is to use the 'Print Screen' button for the entire screen or ALT+'Print Screen' for copying only the application you're in at the moment.
Alternatively, you can do this later on, but record the commands using the `diary` command.
The diary can be started by executing the following command

```
diary session1.txt
```

Also do this at the start of every new session (but with an appropriately named filename). When you finish your session, use `diary off`. Hint: if you type in the command

```
format compact
```

the resulting output does not take up too much space.
Don't forget to also copy-paste scripts, functions, figures, etc. – they are not included in the diary.
Note that the diary file needs to be modified and annotated – a logbook consisting only of diary files will be marked down considerably.
Remember to save a copy of all your created files (Word file if you didn't do it online, diary file, scripts, etc). This can be achieved in several ways:

- Email it to yourself (e.g., as a draft; safest way)
- Save to your OneDrive (login via `onedrive.lincoln.ac.uk`, don't forget to logout at the end of the session otherwise other people may see your files).
- One may want to store it on a USB stick, but they are not very reliable. Hence always make an extra backup! To prevent data loss, use the 'eject' button in Windows at the end of your session instead of just pulling out your USB stick.

Best is to do at least two of the previously mentioned options.
Notes:

1. Sometimes errors that you typed in MATLAB can also be included in your logbook, since this may be part of your learning process.
2. Every student should have their own logbook; copying between students is plagiarism and can have serious consequences.
3. Pages of long repeated output (such as an array of a million elements) do not add much to a logbook; please only include the first few and last few lines of them.

1 Exercises session 1

1.1 Matlab OnRamp course

1. Register online for a Matlab account using your university email address, and complete the online Matlab OnRamp course entirely. See https://matlabacademy.mathworks.com/R2023a/portal.html?course=gettingstarted

   When you finish the entire Matlab Onramp course, include your certificate of completion in your logbook (in contrast to the other exercises it is not necessary to include the solution of each task of the OnRamp course in your logbook, but you are free to do so).

   More help for accessing the OnRamp course can be found on Blackboard.

1.2 Matlab Installation

1. Install Matlab on your own laptop/desktop computer. Instructions are on Blackboard.

2 Exercises session 2

2.1 Calculator

1. Use Matlab to determine

   (a) $123/456$

   (b) $\ln(5)$

   (c) $\log(5)$

   (d) $\tan(6)$

   (e) $7^{1/3}$

   (f) e^8

   (g) $\sin(2\pi/3)$

   (h) $\sin(\cos(9))$

   (i) $e^{2\sqrt{3\sin(4+\log(5))+6}-7}+8$

2. The radius r of a circle is 6. Calculate the area A of the circle using Matlab. Hint: $A = \pi r^2$.

3. Look up the command `acos` using the command `help`, and calculate $\arccos(a)$ for $a = 1/2$. Divide the answer by π to write the result as a fraction of π.

2.2 Variables

1. Set a variable a equal to $7/3$, b equal to $9/6$. Finally, set a variable c equal to $a \cdot b$. What is c?

2. Create a variable `myage` and store your age in it. Subtract two from the value of the variable, and update the variable `myage` with this new value. Add one to the value of the variable, and update the variable again. Observe the Workspace Window and Command History Window as you do this.

3. Set a variable x equal to 10. Increase the content of the variable by 1. Now multiply the content of the variable by a factor 2. What is the value of x?

4. Set a variable r equal to 6. Set another variable A equal to πr^2 (by referring to the variable r), to calculate the area of a circle. Increase the radius by 2 and re-calculate the area. Hint: you can use the up/down arrows on the keyboard to re-use a previously typed command. Afterwards, repeat the calculation for a radius of 1.3×10^{-12}. Does A need to be updated?

2.3 Variable names

1. Test and tell for each of the following names if it is a valid variable name. Hint: see if it appears in your workspace, if you try to assign a value to it.

   (a) `x`

   (b) `x2`

   (c) `2x`

   (d) `xy`

   (e) `x x`

   (f) `exc3.1`

   (g) `tan`

   (h) `if`

   (i) `end`

   (j) `a long variable name`

   (k) `a_long_variable_name2`

   (l) `x$`

   (m) `x%`

   (n) `a_`

   (o) `_a`

2.4 Scripts

1. (a) Create a script with the name `my_script`, by typing in the command line:

   ```
   edit my_script
   ```

   Note that MATLAB will automatically add the extension `.m`, so one does not need to type this.

   (b) In the script set $a = 1$, $b = 2$, $c = 3$, $x = 4$ and then set a variable y to $ax^2 + bx + c$.

   (c) Execute the script by typing in the name of the script on the command line (without the extension `.m`!). What is the value of y?

2.5 Vectors and intrinsic functions

1. Use the colon, :, to create a vector of integers starting from 11 to 20. Hint: Use the help function for : for more information

2. Use the colon operator to

   (a) create a vector of integers starting from 100 to 110 with step 2

   (b) create a vector starting from 100 to 110 with step 0.1.

3. Use `linspace` to create a vector from 11 to 20 with 50 elements.

4. Create a vector of 10 elements between 0 and 3, set the result in a variable **x**, and then calculate $\sin(x)$ for all these values at once.

5. Create a vector **x** containing the values $0.1, 0.2, 0.3, \ldots, 1.0$ and hence find the following functions of the entries

   (a) The hyperbolic sine. Hint: use `lookfor` hyperbolic

   (b) The natural logarithm.

   (c) The base-10 logarithm.

2.6 Complex numbers

1. Calculate

   (a) $(1+i)(1-i)$

   (b) $|2+4i|$

   (c) i^i

   (d) $\sqrt{i}$

   (e) $(3+4i)(5-6i)^{7+8i\cos(9+10i)/11}$

2. Set x to the imaginary part of $\tan(i)$. What is the value of x? Hint: `imag`.

3. Set $x2$ to the real part of $\sqrt{i}$. What is the value of $x2$? Hint: `real`

4. Determine the argument of $z = 1/i$ using MATLAB.

3 Exercises session 3

As a reminder: keep everything in your logbook. One can copy-paste the relevant text (by selecting it with the mouse) to an (online) Word document. However, be sure to edit and remove superfluous and redundant text, as otherwise the logbook becomes long-winded. Hint: using the command `format compact` makes the output more compact. Make sure to include completion date, weekly summary and reflection.

3.1 Plotting

Create the following plots within your own *scripts*. Save all the resulting plots as `.jpg` files with the same name as the exercise number. Include them as well in your logbook.

1. Plot the function $y = \sin(x)$ for $x \in [-2\pi, 2\pi]$ using an interval in x of $\pi/50$. On the same figure plot $y = \cos(x)$ with a dashed, red line. Your x-axis should be in the range -2π to 2π and your plot should include a legend.

2. On one figure plot the four curves $y = x^2 + c$ for $c = 0, 2, 4, 6$, with $x \in [-5, 5]$ (using at least 100 points). Hint: get help on the command `hold`

3. Create a plot of the function $y(x) = e^{-x^2}$ for x in the range -0.5 to 1.5. Your plot should have the following features

   - The curve should be a black broken line.
   - The y-axis should be labelled 'This is the y axis'.
   - The range of x-values in the graph should be -0.5 to 1.5. Hint: look for `axis`
   - The range of y-values should be 0 to 1.
   - The plot should have the title 'Plot of the function y(x). Author: [Your name]'

   Hint: you may have to resort to the `lookfor` and `help` commands to achieve all of this.

3.2 Vector indexing – one element

1. (a) Create a vector of 5 elements ranging from 2 to 4 (equally spaced), and store the result in a variable $\mathbf{v}$.
   (b) Store the first element of this vector in x.
   (c) Store the last element in y. Hint: use MATLAB help for `end` and read the part about the *index* (discard the other meanings).
   (d) Store the third element of this vector in z.
   (e) Calculate $x + y + z$. Check your answer by hand.

3.3 Vector operations

1. Calculate the following quantities using MATLAB vectors

   (a)
$$\sum_{k=1}^{50} k = 1 + 2 + 3 + \ldots + 50$$

   (b)
$$\sum_{k=1}^{20} k^2 = 1 + 2^2 + \ldots + 20^2$$

2. Find the following sums by first creating vectors for the numerators and denominators

   (a)
$$\frac{3}{1} + \frac{5}{2} + \frac{7}{3} + \frac{9}{4}$$

   (b)
$$\frac{2}{1} + \frac{4}{2} + \frac{8}{3} + \frac{16}{4} + \frac{32}{5} + \frac{64}{6} + \frac{128}{7} + \frac{256}{8}$$

3. Write a script that sets a vector x to a range of integers 0, ..., 10. Also include in the script:

   (a) Set a scalar variable `maxx` to the maximum of the vector x. Hint: use the help for `max`
   (b) Set a vector variable `cosx` to the $\cos(x)$ for all values of x.
   (c) Set a scalar variable `meanx` to the avereage of x. Hint: use the help for `mean`.

3.4 Strings

1. (a) Create a string ‘this is a string’ and store it into a variable called `s`
   (b) Create a string named `s1` with contents ‘this is ‘
   (c) Create a string named `s2` with contents ‘a string’.
   (d) Combine the two strings in a new variable called `stot`.
   (e) Change the content of the first element of the string `stot` to ’T’, by using an index to the vector
   (f) Add an exclamation mark to the end of the string `stot` and store it in the same variable again.

3.5 Boolean algebra

1. Determine whether the following expressions are true or false by using MATLAB:
   (a) $1 > 0$
   (b) $3 \geqslant 4$. Hint use the help on >= for more information.
   (c) $(3 > 4)$ or $(3 < 4)$
   (d) $8 \neq 9$. Hint: use `help` $\sim$ and `help` $\sim$=.
   (e) $3 = 3 + 1$
   (f) not false
   (g) true and (true or false)
2. Set a variable n equal to π. Set $b1$ equal to $(n > 3)$. Set variable $b2$ equal to ’not $b1$’. What is the content of variable $b2$?
3. Create a string named s with contents ’hi’. Determine if it is equal to the following using `strcmp` (use MATLAB help for more info):
   (a) ’Hii’
   (b) ’hi’
   (c) ’Hi’
   (d) ’hi ’

4 Exercises session 4

4.1 Output

1. Output the string ’This string has been written to the screen’ using the `disp` command
2. Write a script that sets the variable x to $\sqrt{3}$ and then outputs a *string* from x. Hint: use `num2str`.
3. Write a script that set a number to the variable T and subsequently outputs the text

   ```
   The temperature is ...
   ```

   where the dots have to be replaced by the actual number contained in T. For example, if $T = 37$, the output should be `The temperature is 37`.

4.2 Conditional branching

1. Write a script that sets x to a value. Then use the `if` statement to test if the number is larger than 0. If it is larger than zero, then write to the screen that 'This number is larger than zero'. Run the script with a positive and with a negative number.

2. (a) Write a script that sets the variable `Saturday` to 15, and the variable `Sunday` to 17.
   (b) Add at the end an `if` statement that displays the string 'Saturday is colder' if `Saturday<Sunday`.
   (c) Add another `if` statement, which displays the string 'Sunday is colder', if `Sunday<Saturday`.
   (d) Add another `if` statement, which displays the string 'Both days have the same temperature', if the two variables are equal.
   (e) Change the first line of the script so that `Saturday` equals 18 and make sure the script runs as expected.
   (f) Do the same for `Saturday` equal to 17.

3. Create a script named `testnumber.m`
   (a) At the start of the script, set a variable z to $2 + 3i$.
   (b) Add an `if` statement to test if the imaginary part of z is unequal to 0 (this implies it is a complex number). If this is the case, then multiply z by its complex conjugate (see help of `conj`), take the square root of this and store the result in l. Also set a string named `s` to `this is a complex number`.
   (c) Add another `if` statement, such that if the number z is not complex (imaginary part is 0), then just square the number and take the square root of it, and store the result in l. In this case, also set the string named `s` to `this is a real number`.
   (d) How would you test if the number is a integer (whole) number? Implement this as well in the script, including an appropriate string.
   (e) Test the script by changing the value of z in the beginning of the script to i) a real number and ii) an integer

4.3 Functions

1. Go to https://matlabacademy.mathworks.com, login using the credentials you created during week 1, and open the course MATLAB *Fundamentals*. Complete
   (a) Increasing automation with functions: Creating and Calling Functions (15.1)
   (b) Increasing automation with functions: Function Files (15.2)
   (c) Increasing automation with functions: Workspaces (15.3)

2. In this exercise a user-defined function will be created.
   (a) Start by creating a new file, named `area_circle`, using `edit area_circle` in the command line.
   (b) Add the following code into this file:

```
function A=area_circle(r)
  % Calculates the area of a circle given the radius
  A=pi*r^2;
end
```

Notice that r is the *input argument* and A is the *output argument.*

(c) Calculate the area of a circle of radius 4, by calling the function from the command line by typing

```
>> area_circle(4)
```

(d) Call the function for $r = 2$ (instead of 4)

(e) Within the function definition, change the input argument r to R, and also change the formula to `A=pi*R^2`. Test if the function still works by calling it from the command line.

(f) Change the output argument `A` to `area` and make a similar change in the formula. Test if the function still works by calling it from the command line.

(g) Change the comment in the function and observe this comment by typing `help area_circle` in the command line.

(h) Now call the function for a vector `radii=[3,4,5]`. Does the function work? If not, adjust the function, by replacing `^` with `.^` and test it again.

(i) Finally, change the function so that a diameter is expected as an input argument. Change the symbol R to d for this, and adjust the formula accordingly.

5 Exercises session 5

5.1 Functions and scripts

1. (a) Write a *script* that converts a temperature in Celsius to a temperature in Fahrenheit. The conversion is:

$$F = \frac{9}{5}C + 32$$

where C is the temperature in degrees Celsius and F the temperature in Fahrenheit. Subsequently output

```
The temperature is ... degrees Celsius, which corresponds to ...
   degrees Fahrenheit
```

where the dots should be replaced by appropriate numbers.

(b) Test it with zero degrees Celsius, 25 degrees Celsius and 37.77 degrees Celsius.

(c) Create now a *function* that given a scalar C in Celsius as an input argument, returns the value in Fahrenheit as an output argument. Test the function by trying out known values (i.e., 32 Fahrenheit is 0 degrees and 100 Fahrenheit is about 37.8 degrees).

2. The volume of a cone is given by

$$V = \frac{1}{3}\pi r^2 h$$

where r is the radius of the circular base and h the height of the cone.

(a) Write a *function* with input arguments r and h and output argument the volume. Test it for a radius of 4 and a height of 6.1.

3. This exercise consists of creating a function, and then calling the function from a (separate) script.

(a) The idea of the function is to plot curves of the form $y = c\exp(ax)$, with $x \in [-3, 3]$ within one figure (using at least 100 points). Do this by creating a function with exactly 2 input arguments: c and a. In the function the required curve should be plotted.

(b) Subsequently, create a script that calls the function in the following way. Plot the curve for $c = 1$, $a = 0$, for $c = 1$, $a = 1$, for $c = 1$, $a = -1$, for $c = -1$, $a = -1$, and for $c = -1$, $a = 1$. Plot them all in one figure.

(c) Change the function, so that if either a or c is complex, then nothing should be plotted. Instead, the message

```
Error: both input arguments should be real numbers
```

should be displayed.

(d) Test the modified function in a script, for $c = 2$, $a = 3 - 2i$, for $c = \sqrt{15 - i}$, $a = 12$, and for $c = \sqrt{2}$, $a = -\pi$.

5.2 Loops

Solve all the following exercises by using separate scripts and/or functions.

1. Calculate the follow quantities, but now using `for` loops.

   (a)
   $$\sum_{k=1}^{50} k = 1 + 2 + 3 + \ldots + 50$$

   (b)
   $$\sum_{k=1}^{20} k^2 = 1 + 2^2 + \ldots + 20^2$$

2. Find the following sum by using a 'for' loop using a counter from 1 to 4, calculating the nominator and denominator from this counter, and adding it to an accumulator
   $$\frac{3}{1} + \frac{5}{2} + \frac{7}{3} + \frac{9}{4}$$

3. Calculate
   $$1 + x + x^2 + x^3 + x^4 + \cdots + x^{10}$$
   for $x = 0.4$ using a `for` loop.

4. (a) Write a function with input argument `n` and output argument `S`, which calculates the sum
   $$S = \sum_{k=1}^{n} k^2$$
   using a `for` loop.
   Test the function for $n = 1$, $n = 0$, and $n = 3$.

   (b) Write a function with input argument `n` and `x`, and output argument `S`, which calculates the sum
   $$S = \sum_{k=0}^{n} x^k$$
   using a `for` loop.
   Test the function for $(n = 1, x = 1)$, $(n = 0, x = -1)$, and $(n = 3, x = 2)$.

(c) Write a function with input argument `n` and output arguments `S2` and `S3`, which calculates the sums

$$S2 = \sum_{k=0}^{n} k^2$$

and

$$S3 = \sum_{k=0}^{n} k^3$$

using a `for` loop.

Test the function for $n = 1$ and $n = 2$.

(d) Write a function with input arguments `n` and `m`, and output argument `S`, which calculates the sum

$$S = \sum_{k=0, k\neq m}^{n} \frac{k^2}{k!}$$

using a `for` loop.

Test the function for $n = 1$ and $m = 1$, for $n = 2$ and $m = 0$, and for $n = 2$ and $m = 1$. Check your answers by hand.

5. Create a script. In the script set a variable x to 2. Then calculate the sum

$$S = \sum_{k=0}^{n} x^k$$

using a `for` loop. However, the sum may not exceed 1,000,000. For this, use the `break`. If adding the next term would make the sum exceeds 1,000,000, then break the loop using `break`. Test the script for $n = 10$ and $n = 100$.

6 Exercises session 6

6.1 Nested loops

1. (a) Calculate

$$\sum_{n=1}^{10} \sum_{m=1}^{4} n x^m$$

for $x = 0.4$ using a nested `for` loop.

(b) Now modify the loop such that it exclude the terms with $n = 7$.

6.2 Switch

1. Download the script `switch_test.m` from Blackboard, learning materials.

   (a) Set at the beginning of the script a variable called `number_of_countries` to a number, signifying the number of countries in the UK.

   (b) Change the script so that the correct answer is 4 and not 5.

   (c) Add another switch option, if the answer is 5: 'The answer is slightly lower'

   (d) Add another switch option, if the answer is -1 or -2: 'The answer can not be negative!'. Use the construct with { }

(e) Test the script for all possible pathways of execution.

2. Create a script that

   - Set the variable `name` to a certain value.
   - By using a switch statement, do the following. If it is your own name, display 'Hello, myself'. If it is one of your neighbour's name, display: 'Hello, neighbour!'. If it is any other name, display: 'Hello, Stranger'. If it is either your surname or your neighbour surname, display: 'Hello, you should have entered your first name'.

   Test it for all five cases.

3. (a) Write a function that tests whether an input argument string is a certain letter. If the input string is anything other than a 'Q', it prints an error message; otherwise, the function should not show anything on the screen. Test the function by calling it from a script.

   (b) Create a function with a `switch` statement. In the function test whether the input argument is either 1, 3 or 5, and display the resulting textual number. I.e., if the input argument is 3, display the text 'three'. If the input is something else, show the message 'Invalid value for the input argument. The value should either be 1, 3 or 5.'. Test it by calling the function from a script.

6.3 While loop

1. Given the following loop:

```
while x < 20
   action
end
```

   (a) For what value of the variable `x` would the action be skipped entirely?

   (b) If the variable `x` is initialized to have the value of 5 before the loop, what would the action have to include in order for this to not be an infinite loop?

2. Repeat exercise 5.2.1 but now with a `while` loop. I.e., determine

   (a)
$$\sum_{k=1}^{50} k = 1 + 2 + 3 + \ldots + 50$$

   (b)
$$\sum_{k=1}^{20} k^2 = 1 + 2^2 + \ldots + 20^2$$

3. Repeat exercise 6.1.1a but now with a *nested* `while` loop.

4. (a) From within a script calculate the sum

$$S = \sum_{k=1}^{m} k^2$$

   where m is an integer such that the sum just exceeds 100000. Hint: use a `while` loop. Determine what m is, by using this loop.

(b) Convert the previous script in a function, with input argument the number to exceed and output argument m.

(c) Change the function so that it returns m, such that the sum just not exceeds a certain number (for example, 100000). Hence, if it would be larger than m, the sum would exceed this number.

7 Exercises session 7

7.1 Paths and programs

1. If you do not understand a certain function or command, use the MATLAB `help` function for more information.

   (a) If the user account that you use to log into the windows computer is `User`, then go to the folder `C:\Users\User\Documents\Matlab`, by clicking on the current folder, and replacing the text with `C:\Users\User\Documents\Matlab`. If you have used a different user name, replace User by your user name.

   (b) From within the command window, create a folder called `session7`, using `mkdir`

   (c) Go to this folder within MATLAB using the command `cd` (i.e., type `cd session7`.

   (d) Within this folder, create a function `my_square.m` that squares the input argument number and returns as an output argument this square.

   (e) Call the function from the current folder by typing in the name of the function from the *command window*. Make sure it is working as expected.

   (f) Go to the parent folder, using `cd ..`

   (g) See if you can run the function from this folder. MATLAB should respond with an error message.

   (h) Go back to the `session7` folder, using again `cd`.

   (i) Add the current folder to the path, making use of the functions `addpath` and `pwd`, e.g., use `addpath(pwd)`.

   (j) Go to the parent folder, using the command typed in before.

   (k) See if you can run the function `my_square` from this parent folder (if not, something went wrong; try to fix it).

2. (a) If you haven't done so already, create a folder `session7`. Then go to this folder.

   (b) From here, create the folders `functions` and `scripts`. Add the two folders to the path.

   (c) In the folder `functions`, create a function that calculates $\log_3(x)$, i.e., the logarithm with base 3, and returns the result. Call the function `log3`.

   (d) In the `scripts` folder, create a script that first sets a number to a certain value. Then test if the number is larger than zero. If so, calculate the $\log_{10}(x)$, $\log_3(x)$, and $\log_2(x)$ of the number, otherwise show an error message. Name the script `calc_logs`. For the $\log_3(x)$ calculation, use your previously created function.

   (e) From the folder `session7` call the script `calc_logs`.

3. (a) Create a function `readradius` that asks the user for a radius. The function should return this radius.

(b) Create a function `calcarea`, that has a radius as input argument and returns the area as output argument (see one of the previous exercises).

(c) Create a function `printarea`, that has a radius and area as input and uses `disp` to display

```
The area of a circle of radius ... equals ....
```

(d) Finally, create a script that first calls the `readradius`, then `calcarea` and finally `printarea`, while passing appropriate parameters for all these functions.

Notice that this program is in some sense trivial and could be done within a single script, but it serves to illustrate the idea of a more complex program.

7.2 Test-driven development

1. Write a function `my_nexthour` that receives one integer argument, which is an hour of the day, and returns the next hour. This assumes a 12-hour clock, so for example the next hour after 12 would be 1. Here are two examples of calling this function:

```
>> my_nexthour(3)
ans =
     4
>> my_nexthour(12)
ans =
     1
```

   Test your function for both these values.

2. Download the files `test_nexthour.m` and `nexthour.m` from Blackboard and place them in a convenient folder (which is not necessarily the `Downloads` folder).

   (a) The first `assert` test in the script file `test_nexthour.m` has an appropriate error message. Modify the two other tests, so that they also have an appropriate error message.

   (b) Add appropriate *pass* messages after *each* `assert`

   (c) Modify the function `nexthour`, by editing the file `nexthour.m` and fixing the bugs in this file. The behaviour of the function `nexthour` should be the same as the previous exercise.

   (d) Add an extra line in the test script, that tests if the `nexthour(4)` equals 5.

   (e) Add an extra line in the test script, that tests if the `nexthour(12)` is smaller than 13.

   (f) Now run the script `test_nexthour`, and make sure all `assert` tests pass.

3. (a) Write a function that, given a scalar x, returns $x^2 + 2x + 3$. Test it with $x = 0$ and $x = 1$, and check your result by hand.

   (b) Create a script with name `test_my_polynomial` that tests the function `my_polynomial`. Test that if this function is given the argument 0, it returns 3, and for the argument 1, it returns 6. The tests should use `assert` as in the previous exercise on `nexthour`.

   (c) Write a function that, given a vector x, returns $x^2 + 2x + 3$, element-wise. Test it with the vector [1, 2, 3] and check your result by hand.

   (d) How would you test this function with vector input in the script `test_my_polynomial`?

8 Exercises session 8

8.1 File I/O

Some extra files are on Blackboard in the item *Session 8* and packaged as a `.zip` file. Download this file and *unzip* the files by clicking on the `.zip` file with the right button of your mouse and select `Extract All ...`.
These files are needed for some of the following exercises.

1. (a) Create the folder `session8`, and make this the current MATLAB work directory (you could use `cd`).

   (b) Download both the files `time.txt` and `temp.txt` (they are in the aforementioned `.zip` file) and put them in this folder.

   (c) Create a script, where you load each file in a variable, using `load` (twice). Finally, plot the time of the day in hours vs the temperature of a human in degrees Celsius.

2. (a) Download the file `sample.txt` from Blackboard (this is in the aforementioned `.zip` file) and put it in your current MATLAB work directory.

   (b) Create a script named `reverse_script.m` that does the following

      i. Set a string that signifies a input filename in the variable `inputfilename`

      ii. Set a string that signifies an output filename in the variable `outputfilename`

      iii. Reads the data in the input file in a vector (make use of the `load` command and the variable `inputfilename`).

      iv. Reverse the order of the vector (i.e., [1,4,6] should become [6,4,1]), using the `(end:-1:1)` indexing method on your vector variable.

      v. Writes the vector to the output file in plain text. You can use `save` command for writing the output file with name `outputfilename`. A plain text file in MATLAB is known as ASCII encoding, and can be selected using the option `-ascii` with `save`, see help.

   (c) Run your script with the input file `sample.txt` and output file `reversed.txt`. Check the contents of the output file by opening it in Microsoft Word or by using `type reversed.txt` in MATLAB.

   (d) Run it with input file `reversed.txt` and now output file `double_reversed.txt`. Check that the original file is restored, by opening both `sample.txt` and `double_reversed.txt` in Windows.

   (e) Create a function named `reverse_function.m` that has two input arguments: `inputfilename` and `outputfilename` and has no output arguments. The function should do the same as the `reverse_script.m` (except now the lines with the string variables that appeared in your script are now input parameters for the function). Also test this function by calling it from the MATLAB command line, i.e., `reverse_function('sample.txt', 'sample_reversed.txt')` and check if the output file contains the expected data.

3. Download the script `fileex.m` from Blackboard (this is in the aforementioned `.zip` file). Create a file with name `subjexp.txt` by typing `edit subjexp.txt` at the MATLAB prompt. The file should contain the following

```
5.3 a
2.2 b
```

```
3.3 a
4.4 a
1.1 b
```

Modify the script `fileex.m` such that it displays the sum of the numbers from the file.

4. Write a script that will read from a file x and y data points in the following format:

   ```
   x 0 y 1
   x 1.3 y 2.2
   ```

   The format of every line in the file is the letter ’x’, a space, the x value, space the letter ’y’, space and the y value. First, create the data file with 10 lines in this format. Do this by using the editor, then save the file as `xypts.dat`. The script will attempt to open the data file and error-check to make sure it was opened. If so, it uses a `for` loop (since the file contains exactly 10 points) and `fgetl` to read each line as a string. In the loop, it creates `x` and `y` vectors for the data points. After the loop, it plots these points and attempts to close the file. The script should print whether or not the file was closed successfully.

5. Modify the script from the previous problem. Assume that the data file is in exactly the same format, but do not assume that the number of lines in the data file is known a priori. Hence, instead of using a `for` loop, loop until the end of the file is reached using a `while` loop. The number of points in the file should be mentioned in the plot title.

8.2 Scope

1. (a) If one sets a local variable within a user-defined function to a value. Can one change this value from the command prompt?

   (b) If one sets a local variable within the command prompt, can one change this value from within a user-defined function?

8.3 Debugging

1. (a) Create a new script with the following piece of code.

   ```
   1 S=0;
   2 x=0;
   3 while x<5
   4     x=x+1
   5     S=S+x
   6 end
   ```

   It adds the numbers 1 till 5 together. Don’t add the linenumbers to the code, they will be visible automatically.

   (b) Place a *breakpoint* at line 6, by clicking on the 6 in the editor. A red indicator will appear. Run the code, and when paused press *Continue* or enter `dbcont` three times. What are the values of S and x now? Is this what you expect?

   (c) Now place a *conditional breakpoint* at line 6, by clicking on the 6 in the editor → right-mouse click → and enter the text `x==3`. Then run the program. What is S when the program breaks? Is this what you expected? How many times do you have to press *Continue* or enter `dbcont` to end the program?

Extra material

In case you finished all exercises for all sessions so far, and have your entire logbook up-to-date, then the following exercises can be used to fill up the remaining time of the in-class session.
This material will not be asked during the test and does not have to be part of your logbook, but it is good for your personal development in case you have time left in the session. You can continue your progress at any time.

1. Complete the course MATLAB fundamentals (accessible via `https://matlabacademy.mathworks.com/mycourses`).
2. Complete the course MATLAB Programming Techniques (accessible via `https://matlabacademy.mathworks.com/mycourses`)