Lesson 2: Introduction to Matlab for marine disease modelling

Learning objective: The learning goal of this lesson is to use the software MATLAB for mathematical calculations and formulations, to create script files and make plots, and read external files.

Self-learning steps: The lesson introduces new users to MATLAB. The student will learn to make mathematical calculations, reading and analyzing data, and making graphics. First, the learner will learn to getting access to matlab and use different windows. Second, the student will make easy calculations using basic operators and defining variables. The student will also learn how to use the help window and matlab learning online resources. Finally, the learner will create scripts, read files and make and save plots in different formats. The student will need to spend 3 hours to learn how to use matlab basics for the following modelling exercises including creating scripts, running models and making plots.

Additional resources for the lecture: Additional learning resources for modelling and programming with MATLAB or the alternative open source software GNU Octave can be found on links 5 and 6 in section "Readings and other resources".

2.1. Getting acces to matlab

EHU/UPV has a site license for MATLAB. You can download the package onto your personal computer.

https://es.mathworks.com/academia/tah-portal/universidad-del-pais-vasco-31427936.html#get

This software works with Windows, Mac OS X and linux/UNIX operating systems. Contact us for information about downloading and installing this software.

2.2. The Matlab window

Opening MATLAB, here is a large command window in the center of the workspace or window. This place is where you will type commands. There are a number of action buttons (Home, Plots, Apps, Editor) along the top of the window which have pop-up menus for these actions.

Smaller windows on either side of the central command window have information on variables that have been created (right), folders and scripts that exist in the current directory (left), lists of previously used commands or history of typed commands (right). These windows can be showed by clicking on the LAYOUT button on the menu bar or removed by clicking on the ”x” in the upper right corner of the window.

New scripts can be created by clicking NEW/SCRIPT on the top-left of the window, on the menu bar.

2.3. Formulas and calculations in Matlab

MATLAB responds to typed commands. Some commands are available on pull-down menus in various places. The effects of commands are cumulative. That is, results are saved so later commands can work with results created in earlier commands. Commands are typed into the workspace. These might be assigning values to vari-

ables, evaluating mathematical functions, creating graphs and so forth.
Variables are defined by giving them a value:

- Typing x=5; will create a variable called x and give it the value 5. The semicolon at the end of the line prevents MATLAB from echo-printing the variable name and its value.
- Typing x will cause MATLAB to repeat the variable name and print its value.
- Typing x=1:100; will create a variable called x and assign it a set of values from 1 to 100. In this case, x is a vector (or list). You do not have to tell MATLAB ahead of time that you want to create a vector.
- Typing y=x+3; will create a variable called y and give it the values of x increased by 3. If x is a vector, then every element of the vector will be increased.
- Typing x2=cos(x); will create a variable called x2 and give it the values of the cosine of x. [Note that sin assumes the argument is in radians, not degrees.]
- x=zeros(5,10); creates a table with 5 rows and 10 columns all of which have the value 0 (zero).
- A similar creator ones(10,20) creates a table filled with the value 1.
- These creators are based on matrices so the command x=zeros(10); creates a table with 10 rows and 10 columns.
- To create a list (vector), use x=zeros(10,1); which creates a variable with 10 rows and one column (a column vector) or x=zeros(1,10); which creates a variable with 1 row and 10 columns (a row vector).

The arithmetic operations are +, -, *, and /, for add, subtract, multiply and divide. So, typing y3 = y + x; will create another variable and give it the values of the sum for y and x, assuming the variables have the same ”shape”–that is that they are both single numbers or both vectors or matrices with the same number of rows and columns.
Because MATLAB is based on vectors and matrices, the multiply operation is a

"vector product" which will multiply the corresponding elements of the vector and add the results (a vector inner product).
In order to simply have the corresponding elements of the two vectors multiplied, the operator is .*.
Similarly, the divide operator (/) is a matrix inverse operator. To divide corresponding elements of two vectors, use the operator ./.

2.4. Predefined Variables

Some variables with known values are pre-defined. So, if you use the variable pi then you get the expected value of the ratio of the circumference of a circle to its diameter. Similar common variables are e (the base of natural logarithms), i and j (the square root of -1), and others.

2.5. Online resources and other documents

There are two general ways to get help. If you know the command name and just want a reminder of the options, then type help COMMAND in the command window, where COMMAND is the name of the command you are interested in. This action will produce a few lines to a few pages of details about the given command.
Try help sin to get information about the trigonometric function "sine".
A more comprehensive (web based) help system is available in two ways. Typing doc COMMAND will open the help browser (which is a part of MATLAB) and show you the general help for the command. If you just type doc then the general help window is shown in the browser and you can find the command by various means. Finally, there is a tab called "help" at the top of the MATLAB window which will open the general help browser system.

MATLAB has diverse capabilities and many online sources to learn how to use them.

The documentation 'getting started with MATLAB' on the web page of Mathworks - makers of MATLAB and SIMULINK- is lalso an interesting starting point. The link is:

`https://www.mathworks.com/help/matlab/index.html`

2.6. Script files and the built-in editor

Comands into MATLAB need to be entered one at a time. MATLAB allows you to enter all of the MATLAB commands into a file (the file must end in ".m" which identifies it as a MATLAB script file). Then typing the file name (without the ".m") will cause MATLAB to read the file and execute the commands in sequence. When MATLAB gets to the end of the file, it will stop, leaving all of the results that were created and wait for your next command.

The script file can be created with any text editor that makes plain text files (nano, vi, emacs, textedit, etc.). But there is a built-in editor in MATLAB which can be started by clicking on the "Desktop" tab at the top of the MATLAB window and choose "Editor". A new window will open with the editor. A new file (script) can be created easily by clicking on New/Script. The editor commands are somewhat intuitive with arrow keys moving the cursor or using the mouse to move the cursor by clicking on a location. Typing adds text. Delete removes text. The script can be saved by clicking Save/Save as...
Under the "help" tab on the editor window, there is an entry "Using the MATLAB editor" which will get you started.

2.7. Reading data files

MATLAB can read text and excel files, with variables as headers (first row ofeach

column) and a large variety of files, but the details of this capability can be daunting. You can import files using Matlab windows. For this, It helps to put your files to read on the folder you are working with scripts (see MATLAB left panel, 'current folder'). Position your mouse pointer on the file to import at the 'current folder' window and click on the right button on the mouse to show the secondary menu up. Then, click on 'import data'. A matlab window is poping up showing the imported file (.txt or .xlsx). Select one option for the output type. If you want to have each column imported as a variable, then you need to select the option 'as columns'. FInally, click on 'import selection' on the right of the window and each column is imported with its' data associated to the variable name on the original file.

Another option could be this other one. If your data are in a plain text file and in columns, then there is an easy way to read the data. Note that there cannot be any header line or any text in the file (for this simple method to read data)– only numbers. There cannot be any missing data items (no blank entries in the columns).

If the file name is "data.dat" and it has columns of numbers (for example: cases, temperature and salinity), then the command 'data=load('data.dat')' will read the numbers into the variable called data. The character string in the parentheses must be the correct file name. The variable to the left of the = symbol can be anything you want–I use data out of habit. There are no conventions for the file name–any name will work. I use .dat to indicate data files, but any chose (or none at all) will work.

The variable data is a table of numbers with the same shape (number of rows and columns) as the numbers in your file. Continuing the example above, the first column would be the depth, the second and third columns would be the temperature and salinity measured at that depth, respectively.

It is useful to make the variable names more meaningful. So, the following will take the columns of data and put them into new variables. References to specific items in a variable is accomplished with data(r,c) where r is the row number and c is the column number that is desired. For example, data(3,2) would get the value in the third row and second column of the variable. It is possible to get a range of values using the "colon" notation. So, data(1,1:3) would get the first row and columns 1 through 3.

Finally, to get all values along a dimension of the table, just put a colon in that location. So the first command below extracts all rows of the first column of the variable data.

```
cases=data(:,1);
temp=data(:,2);
sal=data(:,3);
clear data
```

So, there are now two variables with columns of numbers giving the , temperature and salinity from the data file. The last command above removes the variable called data since we no longer need it.

2.8. Making simple graphs

Simple line plots can be made with the plot command and associated labeling commands. A short example is:

```
x=0:0.1:1;
y=cos(x);
plot(x,y)
```

```
title('Cosine')
xlabel('x')
ylabel('cos(x)')
```

The first command creates a vector (a list) having the values from 0 to 1 in steps of 0.1. The next line calculates the cosine of all values of x and stores the results in the variable y. The plot(x,y) command opens a graphics window, makes choices for the axes and plots the line. The final three commands put a title on the plot and label the x and y axes. Note the use of a single quotes (') to surround text to indicate a character string. The plot command can create a wide variety of line plots. Some options are given below:

```
plot(x,y,x,z) % plot two curves
plot(x,y,'--') % plot using a dashed line
plot(x,y,'r-.') % plot using a red dash-dotted line
plot(x,y,'b',x,z,'g') % plot two curves using blue and green lines
```

To get more information about options for this command type help plot. Note that the % symbol indicates a comment. MATLAB ignores any typing after % to the end of the line.

2.9. Saving and printing plots

There are two ways to save the plot or make paper copies. The most direct is to use the "File" menu on the graphics window to send the plot to the printer. If you want to save an electronic version, you can print to a file which will save a postscript version of the plot (by default, other choices are available on the save window).
A more versatile option is to use the print command in the command window. This command allows control over the image format and other features of the file that is

created. It also will automatically save a figure when it is created so you don't have to remember to do this every time. The command needs two strings as input: device details and file name.
print('-djpeg','sin.jpg') print('-dtiff','sin.tif') print('-dpng','sin.png') print('-dpsc2','sin.ps') print('-depsc2','sin.eps') The device information is in a string that begins with "-d" and is followed by information on the image type. Use help print to find other options for the file/image format. The endings on the file names are traditional, but have no effect on the format. Other commands or software may care about the ending.

2.10. Saving results for later use

The simplest way to save variables and values for later use is with save FILENAME. All of the currently defined variables (names and values) will be saved to a file named "FILENAME.mat", where you can use any string that you choose in place of "FILENAME". The ending ".mat" is required by MATLAB so it will know that the file is a MATLAB formatted file.
If you exit MATLAB and then start it again at a later time, the command load FILENAME will cause MATLAB to read the file and restore all variables and values that you saved.
Sometimes you only want to save certain variables for later use. For example, if you have read a data file and created variables depth, temp and sal, then you can save just these variables with the command save ctd001 depth temp sal which will create a file called "ctd001.mat" which will contain the three variables.