How to use this Textbook

A brief guide as to how to interpret and make best use of this book, follows.

Fonts and highlighting

Throughout ... but be aware: probably not particularly consistently, the following formatting in the text is used to distinguish the specific context of the word:

- **Bold** – indicates program/software names (e.g. **MATLAB**).
- *Italics* – indicates technical/jargon words, particularly specific to **MATLAB** (but not command words or functions themselves) or programming (concepts), e.g. *loop*.
- Sans-serif font family typeface – indicates keyboard keys (e.g. F5), program menu items (e.g. Save as ...), program window names, and filenames (except where used in **MATLAB**).
- `Typewriter font family typeface` – indicates **MATLAB** commands and *functions*, and lines of code (see examples below).
- Color highlights in the text are used to reflect the colors employed by **MATLAB** at the command line, or in the code editor.

Help(!) and keyword definitions

MATLAB `help` is not always especially helpful. In the course text, for each *function* that MATLAB provides a comprehensive help on, such as `help`, a simple summary version will be displayed in the right hand margin in a grey box. For example – the box headed `FUNCTION`.

Also appearing in grey boxes in the margin are overviews and summaries of **MATLAB** commands or functions as well as ways to do things in **MATLAB**. For example – the box headed ***loops***.

`FUNCTION`
A simple and/or summary usage of particular **MATLAB** commands and *functions* is provided in a grey-background box in the margin.
...
...

loops
There are a number of different ways of constructing *loops* in **MATLAB** ...
...
...

What and when to type

Examples of **MATLAB** code/commands are indicated by text in a 'Typwriter' font, e.g.

```
A = [1 2 3 4]
```

When the given examples additionally illustrate how they are typed in plus the 'result' OR, requires you to type in the lines at the command line, the text is again highlighted text in a 'Typwriter' font but in addition, the command line prompt (») is shown at the start of a line (you do not type in the prompt ...), e.g.

```
» hello
```

is asking you to type in `hello` at the command line, and

```
» hello
Undefined function or variable 'hello'.
```

is then showing you what happens!

When you see a string or variable name in all CAPITAL LETTERS – this is a 'placeholder' and is indicating that you should substitute in an appropriate string or variable name in its place, e.g.

```
load('FILENAME','-ascii');
```

is indicating that you substitute the name of your actual file in place of `FILENAME`. Alternatively:

```
plot(MYARRAY(:,1),MYARRAY(:,2));
```

would indicate that you should substitute your actual variable name (holding the data to plot in this example) in place of `MYARRAY`.

Code structure

A visual guide to the structure of your programs is given by schematic figures in the page margin. For example, a generic *script* (yellow box) is shown by Figure 1, and a generic *function* (green box) by Figure 2.

In these schematics, the flow (sequence) of the code is indicated by the red arrow.

For the *function*, that information is passed into the *function*, and then returned back to where the function was called from, is indicated by the red arrows entering the top of the box and leaving the bottom of the box, respectively. (But note that there is no line of code at the end that tells the model to return values ... this is simply to

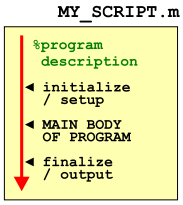

Figure 1: Schematic for a generic *script*.

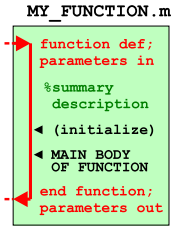

Figure 2: Schematic for a generic *function*.

illustrate the flow of the program, particularly when things get more complicated and there are multiple *scripts* and *functions* involved.)[7]

For the *script*, the code file starts with a comment (`%program description`) summarizing what the *script* does, although after the *function* definition header line, so to should the *function* (somewhere have comment lines describing what it does).

The black left-pointing filled triangles and associated text to the right, indicate categories of code content, and occurring in what order, that the programs might contain.

The purpose of these cartoons is to help you when faced with a blank page and the question: 'Where do I start' or 'What do I write' appears prominently in your mind[8]. It is to give you some sort of idea what bits might go where, and what general content is required in the file. The cartoons do not (and are not intended) to show the exact details of the code content. Nor do they necessarily indicate all the different sections needed. Conversely, not all the sections illustrated may be strictly necessary and in come examples there may be nothing to 'initalize' and there may be no constants of local parameters to define the values of at the program start.

So please – use the cartoons as a simple visual guide to the approximate structure of your program and do not over-interpret them.

[7] Don't worry for now ... it should hopefully all become apparent later.

[8] Also surrounded by flashing neon lights.