An Introduction to MATLAB Programming

Center for Interdisciplinary Research and Consulting

Department of Mathematics and Statistics

University of Maryland, Baltimore County

www.umbc.edu/circ

Winter 2008

1 Introduction

In this tutorial we introduce the basics of programming in Matlab. We will start by reviewing the elements of structured programming; next, we will begin our discussion of Matlab programming. The flow of control and the related Matlab keywords will be mentioned along with several examples for the purpose of illustration. Another topic which we will discuss is the use of Matlab functions. At the end, we will point the audience to the appropriate sections of Matlab's documentations for more information on Matlab programming. The outline of the major objectives of this workshop is the following:

- Matrix Manipulation
- Data Analysis
- Basics of Structured Programming
- Basic Input and Output
- Flow of Control in a Program

2 Matrix Manipulation

In addition to performing algebraic operations with matrices, Matlab provides many useful commands for manipulating matrices in other ways.

The **reshape** command is used to change the shape of a matrix: For instance, **reshape**$(\mathbf{A}, \mathbf{m}, \mathbf{n})$ produces an m-by-n matrix whose elements are taken columnwise from A:

```
>> A = [0 1 2; 3 6 9]

A =

    0    1    2
    3    6    9


>> reshape(A,3,2)

ans =

     0    6
     3    2
     1    9
```

The **diag**, **tril**, and **triu** commands can be used to extract diagonal, lower- or upper-triangular portions of a matrix. Examples:

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

    1     2     3
    4     5     6
    7     8     9


>> diag(A)

ans =

     1
     5
     9


>> diag(A,-1)

ans =

     4
     8


>> tril(A)

ans =

     1     0     0
     4     5     0
     7     8     9


>> triu(A)

ans =

     1     2     3
```

```
     0     5     6
     0     0     9


>> triu(A,1)

ans =

     0     2     3
     0     0     6
     0     0     0
```

3 Data Analysis

Matlab also includes a number of useful functions for data analysis of vectors and matrices. The **min** and **max** commands return the minimum and maximum elements of a vector, as well as the index for that element, as in this example:

```
>> x = [1 5 -2 3 4]

x =
    1     5    -2     3     4


>> [min(x) max(x)]

ans =

     -2     5


>> [m,i] = min(x)

m =

    -2

i =

     3
```

```
>> [m,j] = max(x)

m =

     5

j =

     2
```

When operating on matrices, **min** and **max** will return a vector containing the minimum (or maximum) element in each column of the matrix, and an index specifying which element in the column is the minimum (or maximum), while **sum** will return a vector containing the sums of each column:

```
>> A = [0 -1 2; 1 2 -4; 5 -3 -4]

A =

     0    -1     2
     1     2    -4
     5    -3    -4


>> max(A)

ans =

     5     2     2


>> [m,i] = min(A)

m =

     0    -3    -4


i  =

     1     3     2


>> sum(A)
```

```
ans =

     6    -2     6
```

The examples included here cover only a small subset of Matlab commands and their functionality. The user is encouraged to experiment on his own, and consult the Matlab documentation for more information.

4 Structured Programming in Matlab

Matlab provides extensive math and graphics functions along with a powerful high-level programming language. Programmers can choose to program using the classical structured programming approach, but it is also possible to do object-oriented programming in Matlab. Our discussion, however, will be limited to structured programming in Matlab.

4.1 Structured Programming

Structured programming (modular programming) is a software development paradigm in which problems are solved in a top-down fashion. As a program gets larger, it is sub-divided into sub-programs to provide code that is easier to read, debug, and maintain; also this approach facilitates reuse of commonly used procedures (functions). A natural starting point in our discussion of structured programming in Matlab will be a quick review of the elements of structured programming.

4.2 Basic Constructs of Structured Programming

Recall that an algorithm is a finite set of instructions for accomplishing a given task which given an initial state will terminate in a defined end state. And a computer program is an implementation of a given algorithm in a programming language. In the discussion that follows, we will use Matlab programming language as our implementation tool.

There are three main programming constructs:

1. Sequence
2. Branch (Decision)
3. Loop

The Sequence construct refers to writing a group of programming statements in a sequence. The Branch construct enables us to change the flow of control *if* a given condition is satisfied, and finally the Loop construct enables the program to run a statement (or a group of statements) a number of times.

5 Some Technical Details

Before we go deeper in our discussion of Matlab programming, we take care of some technical details in this section. We will first mention naming conventions for Matlab program files; next, we will talk about Matlab's program development environment, and then have a brief discussion of input and output.

5.1 File Names in Matlab

Matlab programming codes are saved in files with extension `.m`. This gives rise to the so-called Matlab M-files. For example, a program that computes the roots of a quadratic polynomial may be saved in a file named `quadRoots.m`. An M-file may contain a Matlab script or a Matlab function; the distinction between the two types will become clear in the course of our discussion.

5.2 Development Environment

Matlab programs can be created using any text editor. In a Unix environment, one can use editors such as vi, emacs, nedit, etc. In Windows environment, notepad and alike can be used. In addition, Matlab provides a user-friendly editor where programmers can create their programs (and also debug them). To access Matlab's editor, type the following in Matlab command prompt,

```
edit
```

this opens Matlab's editor (Figure 1).

5.3 A Word on Input and Output

Before doing any programming we recall that a typical computer program (in structured programming) receives data as input, does some processing on the given data and provides output. Hence, input and output constitute an important part of any computer program. The programming examples in this tutorial do not involve interactive (or batch) data input. However, in every example, we will create some output. The commands used for providing output are `disp` and `fprintf`. We will discuss these commands at appropriate places throughout this tutorial. Although our discussion of input and output will be limited in this tutorial, we point out that Matlab provides flexible file I/O options as well; for more information, the reader can refer to Matlab's documentations.

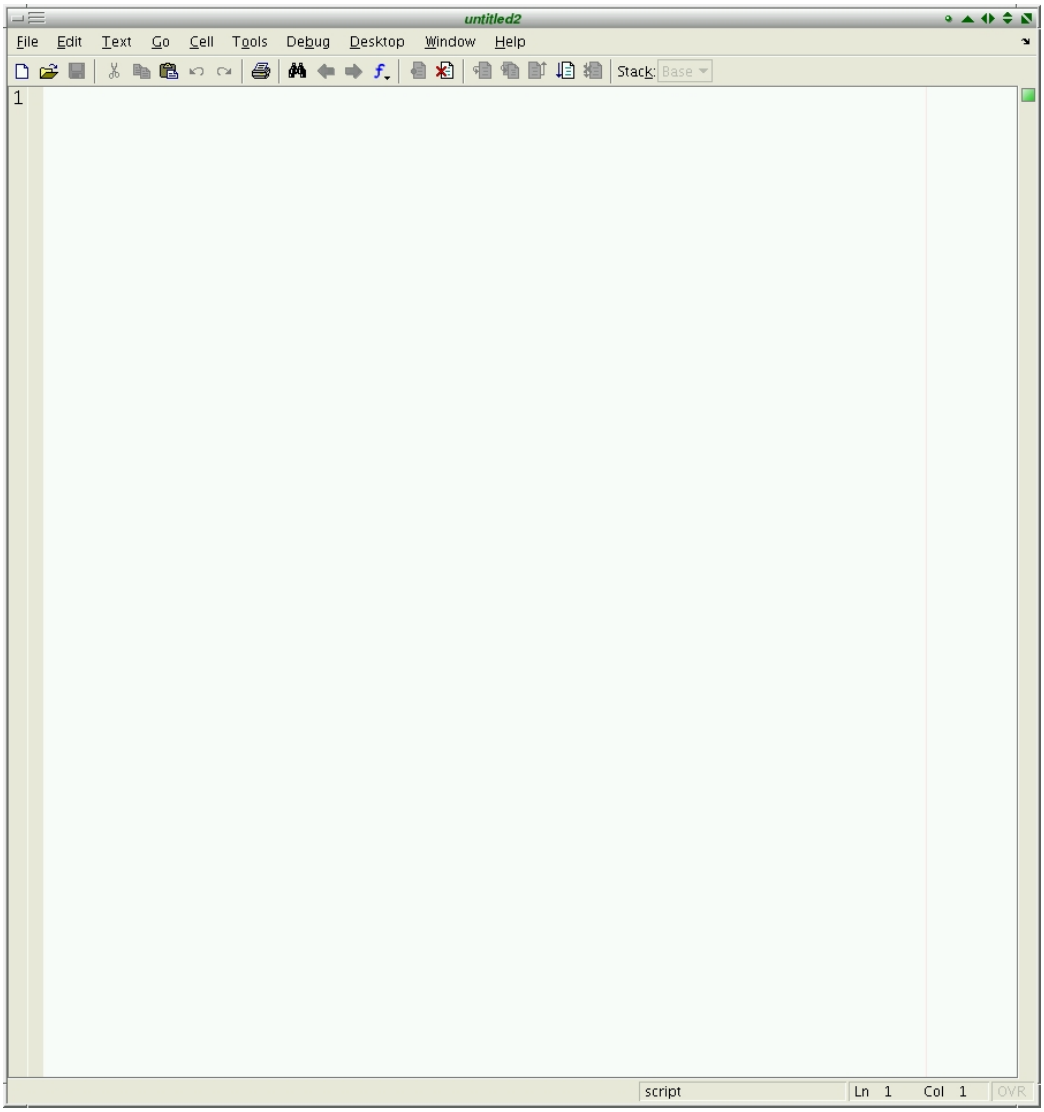

Figure 1: Matlab's program development environment.