**Math 425 Applied & Comput. Lin. Algebra   Fall 2024 Handout**

# A MATLAB Primer I

**What is MATLAB ?**

- a software package used for computation in engineering, science, and applied mathematics that comes with a programming language, graphics capability, and a large standard library;
- symbolic expressions and manipulations are not possible (unless one has Symbolic Toolbox); all variables must have values and all results are (potentially inexact) numerical results;
- compared to other low-level programming languages such as C++, it is slower (but not that much slower), so it is not for high-performance computing

**MATLAB Work Environment**

- After starting MATLAB you get a multipaneled work environment.
- The *Command Window* is located in the middle. MATLAB accepts commands typed at the *prompt* `>>`.
- MATLAB is not a compiled computer language, so after you give a command, it tries to execute it immediately and then awaits another one.
- At the left is the *Current Directory Window.* MATLAB is only aware only of the files in the current directory (folder) and other directories on its *path.* You can add files to a directory on the path.
- The *Workspace* is located at the top right. It shows the current variable names and information about their contents. At the start it is empty.

**Getting Help**

- Type `help <command>` at the prompt to get a quick help on the syntax of a command. For instance try `help plot`. Typing `help` by itself gives you a hyperlink to the help browser.
- Type `doc <command>` to get more extensive help in a separate window. Try `doc plot`.
- There is a function browser in the Command Window, sitting to the left of the prompt. Click on it to see a searchable list of available functions.
- *Getting Started* manual is a great place to start (located on top of the Command Window). Click on it to see the contents. Find your way into Matrices and Arrays and spend two minutes to see what is offered.

**Basic Commands and Syntax** Type in a valid expression and press `Enter`. MATLAB will execute it and return the result. A variable must have a value before it can be used. When an expression returns a single result not assigned to a variable the result is assigned to `ans` which can be used as any other variable. Try the following commands in sequence;

```
>> 5 + 8
>> sin(pi/2)
>> 1/0
>> x = sqrt(3)
>> atan(x)
>> pi/ans
```

The built-in number `eps` (epsilon) is the machine precision which bounds the maximum relative error in representing real numbers and doing arithmetic. Because of the machine precision computations which you expect to return zero are not always zero. For instance, $e^{\ln 10} - 10 = 0$, but try

```
>> exp(log(10)) - 10
```

Furthermore

```
>> % Anything typed after the % sign is a comment.
>> x = rand(10,10); % the semicolon ; is used to suppress output
```

In order to save all defined variables in a file type `save myfile` and they will be saved to a filed called `myfile.mat` in the current directory. Later you can type `load myfile` and the saved variables will be returned to the workspace.

## Nice Things to Know

- Use the up-arrow key to go back to previous command. Try it.
- If you are waiting too long for a computation to finish use `Ctrl-C` to interrupt the computation.
- MATLAB will display only 4-5 digits of a result but it is storing about 15 significant digits. So do not copy or retype a displayed result since you will be losing a lot of information. Wait until the end of all your computations to round of results. If you want to see numbers displayed in the long form output type
  ```
  >> format long
  ```
  And then check out what you get after you compute
  ```
  >> sin(pi/4)
  ```
  To return to the short form output type
  ```
  >> format short
  ```

## Matrices and Vectors

- To enter a small matrix list the elements in between square brackets and use semicolons to separate rows. Try
  ```
  >> A = [1 2 3; 4 5 6; 7 8 9]
  >> b = [0 ; 1 ; 0]
  ```
- The `size` command returns the number of rows and columns:
  ```
  >> size(A)
  >> size(b)
  ```
- You can concatenate matrices as long as dimensions match:
  ```
  >> [A b]
  >> B = [ [ 1 2 ; 3 4] [5 ; 6]]
  ```
- The empty matrix is `[]`. The identity matrix, say the $3 \times 3$ identity matrix, is created by `eye(3)`. A diagonal matrix is created like this: `diag([1 2 3])`. A zero matrix and a matrix of all ones are created as follows, respectively: `zeros(2,3)`, `ones(2,4)`. Try all of these.
- Here are bunch of ways of accessing elements of a matrix:
  ```
  >> A(2,3) % single element
  >> b(2) % one index suffices for a vector
  >> b([1 3]) % multiple elements
  >> A(1:2, 2:3) % the submatrix consisting of the first two rows and last two columns
  >> B(1, 2:end) % first row, all but first column
  ```

```
    >> B(:  , 3) % all rows of column 3
```

**Matrix Operations** The operators `+, -, *, ^` are used in a matrix sense.

```
>> A = [ 1 2 3 ; 4 5 6; 7 8 9]
>> D= [0 0 1; 0 0 1; 0 0 1]
>> A - 3*D
>> Asquare = A^2
>> b = [0; 1; -1]
>> B = [1 -1 1; 1 1 1 ]
>> Atimesb = A*b
>> AB = A*B % What do you observe?  Why?
>> BA = B*A % How about now?
```

**Solving Systems** There are canned commands in MATLAB for solving systems of linear equations. But before we get you should learn how to compute the row echelon form of a matrix. The relevant command is `rref` which stands for *reduced row echelon form.* We know row echelon form of a matrix. To compute the reduced row echelon form we compute the row echelon form and then execute two more simplifications: First we make the pivots equal to 1 by dividing each row (the entire row!) by the value of the pivot in that row. Second we use row operations of type I to make each entry *above* each pivot equal to zero.

**Example 1.** If we want to compute the reduced row echelon form of the augmented matrix

$$\left(\begin{array}{rrr|r} 2 & 3 & -1 & 1 \\ 4 & 7 & 2 & 8 \\ -2 & -5 & -7 & -13 \end{array}\right)$$

we first compute the row echelon form

$$\left(\begin{array}{rrr|r} 2 & 3 & -1 & 1 \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Then we divide the first row by 2, and finally we do $-\frac{3}{2} * R2 + R1 \to R1$ to get the reduced row echelon form

$$\left(\begin{array}{rrr|r} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 0 & 0 \end{array}\right) \longrightarrow \left(\begin{array}{rrr|r} 1 & 0 & -\frac{13}{2} & -\frac{19}{2} \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 0 & 0 \end{array}\right).$$

Enter

```
>> C = [2 3 -1 1 ; 4 7 2 8 ; -2 -5 -7 -13]
>> rref(C)
```

If the coefficient matrix of the system is nonsingular you can use the \operator to compute the solution. You can use the `rank` command to see whether the matrix is nonsingular.

```
>> A = [5 3 -1 ; 3 2 -1 ; 1 1 1]
>> b = [9 ; 5 ; -1]
>> rank(A) % matrix is nonsingular
>> A \ b
```

*In order to prepare this document I used Learning MATLAB by Tobin A. Driscoll heavily.*