Contents

A note to the instructor:

- **Course Manual Content**: This manual contains lecture notes, class work with solutions, possible assignment questions, projects and supplemental material to aid in teaching MATH 2131.

- **Student/Instructor Meeting**: This course will meet for 3 hours per week. You may choose to break up this time into 1.5 hours of front facing instructing and 1.5 hours of student work time/questions.

- **Course Hours**: A typical assignment for this 1 hour lab should only require 3 hours of at home work from the student per week. A typical project should require approximately 9 hours from the student.

- **Course Structure**: This manual will provide you will all the necessary information and review material to teach MATH 2131. All materials MUST be covered in the semester! There is a calendar at the end of this manual which provides a typical outline of covered material, assignment dates and project dates.

- **Student Difficulty Range**: A typical introductory student should be completing roughly 9 assignments and 3 projects in the semester. Given that there is a range of backgrounds for students taking this course, if a student feels the course material is too low of a level, it may be suggested that they complete more projects and no assignments.

- **Other Optional Choices**: You may choose to use Assignment 9 as an opportunity for students to do group presentations in place of a final exam. If this is the case alter Assignment 9 by removing provided data and giving more general instructions. **DO NOT ALLOW THE USE OF THE BUILT IN MATLAB REGRESSION FUNCTION!**

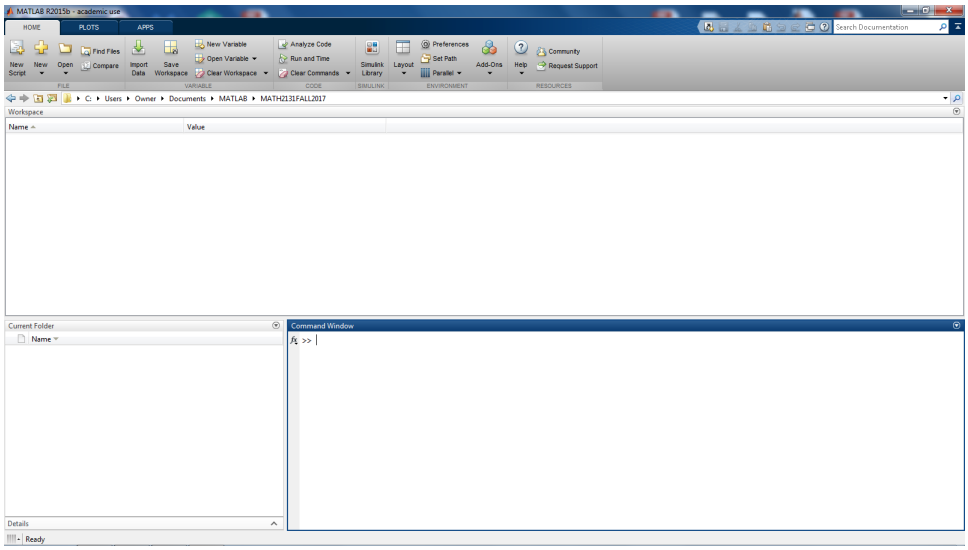
- **Using the Pre-made Questions and Exam Outline**: Since this manual will be used by many people, it is suggested that the instructor use the provided questions and exam as a template for assignments and make minor alterations to ensure academic accuracy (no copying).

Chapter 1

MATLAB Course Introduction

The goal of this course is to develop a basic understanding of scripting in MATLAB while utilizing techniques and applications from Linear Algebra.

1.1 The MATLAB user interface



Below is a list and description of each main section of the MATLAB user interface.

- Command Window: The command window allows you to enter code line-by-line while automatically running each line after the "Enter" button is pressed. (Note: Any code typed in the command window will not be saved.)

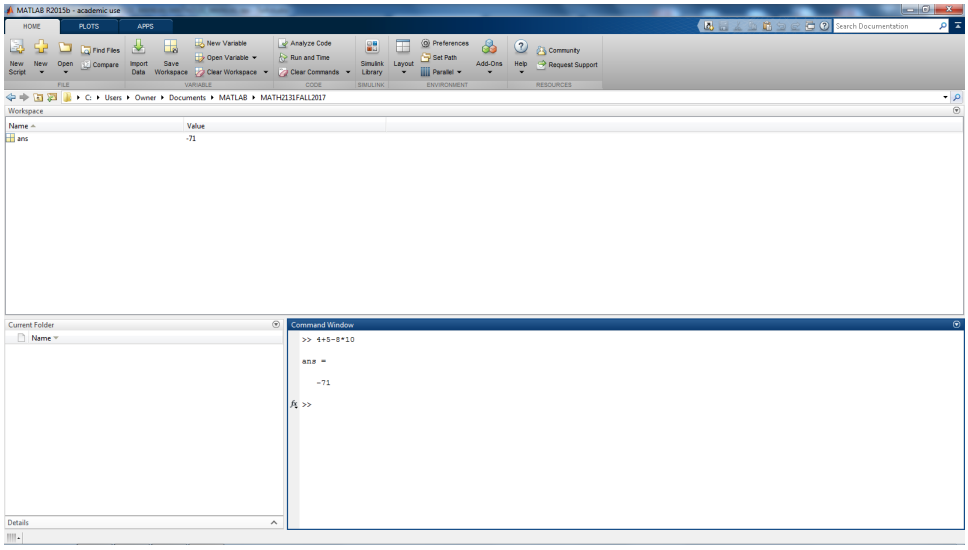
- Workspace: The workspace is where all described variables are temporarily saved. (Note: When the MATLAB application is closed, anything in the workspace is erased.)

- Current Folder: The current folder tells MATLAB where to store all saved data. To set the current folder, click the "browse for current folder" button located to the left of the file pathway (beginning with C:).

- Current Directory: The current directory tells MATLAB where to access previously saved data.

- New Script: The new script button allows you to open a file where you may write and save code to be opened at a later time.

1.2 Running code



To understand how code runs, we will perform a simple example in the command window. Suppose you want to know what the solution is to $4 + 5 - 10 * 8$, type the following code into the command window and hit the "enter" button.

```
>> 4+5-8*10
```

You should notice that the output MATLAB gives has the automatic variable name "ans"; this is because we did not specify what the variable name should be. This topic will be discussed in the next section. You should also note that "ans" has been saved to the workspace. Continue on to the next section without closing out of MATLAB.

1.3 Creating and saving variables

Now, suppose you want to know what the solution is to $7 + 8 - 20$. As before, we type the following code into the command window and press "enter".

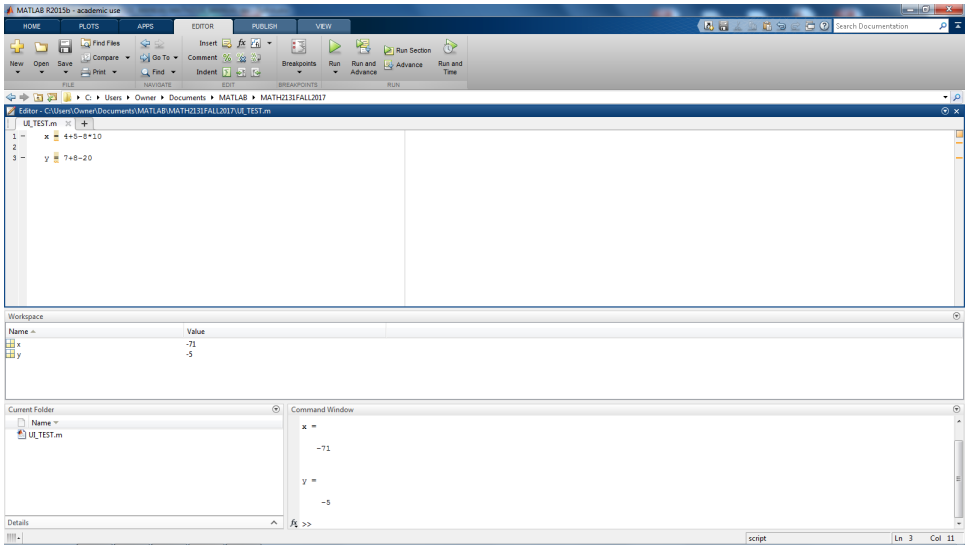
```
>> 7+8-20
```

Pay particular attention to what happens to the variable named "ans" in the command window. The value of "ans" has changed to the last output solution. In order to save our variables for later use we need to give them a label. As an example type the following into the command window.

```
>> x = 4+5-8*10
```

Now there is a variable named "x" in the workspace. This variable will stay in the workspace until you close the MATLAB application or right-click and delete it.

1.4 Creating and saving scripts



If you intend to run code without needing to save it, the command window should be used. In this course we will be saving all of our code. To start, click on the "new script" button at the left hand corner of the MATLAB application.
You may begin typing your code in the new script. Once you have completed typing the code you wish to run, click the green arrow button labeled "run" located under the edit tab. As an example type the following into your script.

```
x = 4+5-8*10
y = 7+8-20
```

Once you hit the "run" button, MATLAB will prompt you to save the file to your current folder. After you have saved the file, you will notice the workspace will have variables "x" and "y" and the command window will output the values of "x" and "y". If you do not wish to see the output in the command window each time you run, add a semicolon at the end of each line in your script.

```
x = 4+5-8*10;
y = 7+8-20;
```

This script is now saved and can be opened at anytime. To open, double click on the file from your MATLAB application. It should be in your current folder.

Chapter 2

MATLAB Coding Basics

2.1 Entering Matrices and Vectors

Here we will learn how to enter matrices and vectors into MATLAB through examples.

To construct $x = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, type the following into MATLAB.

```
x = [1,2,3];
```

You may continue in the same way to construct any 1xn vector.

To construct $y = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, type the following into MATLAB.

```
y = [1;2;3];
```

You may continue in the same way to construct any nx1 vector.

You should notice that instead of using "," to separate the values we use ";" in the second example. The symbols "[" and "]" tell MATLAB that you are creating an array, the symbol "," if used between the two array symbols tells MATLAB to stay on the same line while ";" if used between the two array symbols tells MATLAB to go to the next line. Using this information, construct the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

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

2.2 Slicing Matrices and Vectors

Indexing is a fundamental concept in coding. In MATLAB everything is indexed in variable(row, column) format. For example, suppose we have created the following matrix in

MATLAB,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

and would like to know the value in the 2nd row and 3rd column of A. To do this, we type,

```
value = A(2,3)
```

This will return `value = 6`. We can utilize the indexing of previously defined matrices to create new vectors and matrices. This technique is called *slicing*. For example, suppose we want to create a column vector from the first column in A. Then we would simply type,

```
x = A(:,1);
```

Notice that we used the ":" in this command. In this context, ":" tells MATLAB "ALL" so that our command reads x is equal to all the rows of A in the first column.

Remark 1. *Consider the dimension of the above vector x.*

Now suppose you would like to create a matrix B made from the first 2 rows of A and the first two columns of A. To do this we would type,

```
B = A(1:2,1:2);
```

Then our code reads, B is equal to rows 1 *to* 2 of A and columns 1 *to* 2 of A.

2.3 Concatenation

Concatenation is the putting together of 2 or more things to create a new thing. In our context, we think of concatenation as the opposite of slicing. This is yet another time saving "trick" used in coding. Suppose we have two vectors,

$$x = \begin{bmatrix} 2 & 1 & 2 & 4 \end{bmatrix}, \quad y = \begin{bmatrix} 3 & 3 & 1 & 2 \end{bmatrix}$$

and we need to create the following matrix in MATLAB,

$$A = \begin{bmatrix} 2 & 1 & 2 & 4 \\ 3 & 3 & 1 & 2 \end{bmatrix}$$

We could type everything over again,

```
x = [2,1,2,4];
y = [3,3,1,2];
A = [2,1,2,4;3,3,1,2]
```

Or we could put x and y together to create A,

```
x = [2,1,2,4];
y = [3,3,1,2];
A = [x;y]
```

2.4 Functions and Syntax

Programming applications speak in their own language. We call this language the "syntax". We need to use the correct syntax in order for our code (or list of instructions) to be understood. The following is an example of inappropriate syntax.

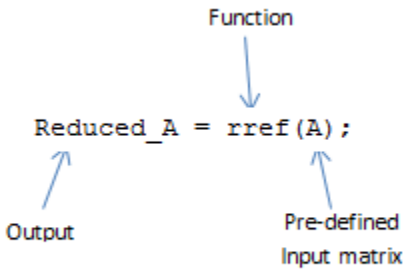
```
y = 2/x;
x = 3;
```

Since x is not defined before y, the application cannot perform the task $y = 2/x$ and will return an error. To fix this we would write,

```
x = 3;
y = 2/x;
```

Programming applications also have built-in *functions* to save us time in coding. For example, if we want to solve $y = 2/x$ as above, we do not need to tell the application how to divide, we simply *call* the function /. Functions can also be used to perform more complicated tasks. To utilize these functions, we need to know how the application recognizes the function inputs and how it returns outputs. We obtain this information from the MATLAB documentation. The documentation can be found here: https://www.mathworks.com/help/matlab/

A simple example of a function we will be using in future assignments is the built-in MATLAB function `rref`. The function `rref` takes in a matrix and outputs the matrix in its row reduced echelon form.



In your script file, type the following lines of code and hit the "run" button.

```
A = [1 4 3; 0 2 6; 3 14 10];
Reduced_A = rref(A);
```

You should notice a new variable named `Reduced_A` has been added to the workspace. Double click on the name and MATLAB will open the variable in an excel-like sheet. You can now check that `Reduced_A` is the reduced row echelon form of the matrix `A`.

More useful functions for the following questions

- `sum`: This function accepts a single matrix or vector input and outputs the sum of entries. If the input is a *vector*, `sum` will return a single value which is the sum of all the entries of the vector. If the input is a *matrix*, `sum` will return a row vector of values which is the sum of all the entries of the column vectors.

  ```
  B = sum(A);
  ```

- `fprintf`: This function requires a minimum of *two* inputs and outputs a string of characters along with the value of a variable in the command window. The first input is a string with a "`%`·" to denote where the variable value should output the second is the name of the variable.

  Input:

  ```
  A = [1 2; 3 4];
  fprintf('The value of the first entry of A is %d', A(1,1));
  ```

  Output:

  ```
  The value of the first entry of A is 1.
  ```

- `%_._f`: This function is used in strings to tell MATLAB how many significant to allow in the output.

  Input:

  ```
  b = 1512.056348;
  fprintf('%4.3f is the value of b, b);
  ```

  Output:

  ```
  1512.056 is the value of b.
  ```

- `%d`: This function is used in strings to tell MATLAB to round output to the nearest integer.

- `randi`: This function takes three values in the input denoting the interval and the size of the requested matrix and outputs a random matrix of integers which fall between the interval and has input size.

  ```
  A = randi([0,5],4,4);
  ```

- `randn`: This function takes two values in the input denoting the size of the requested matrix and outputs a random matrix taking values from a standard normal distribution.

```
A = randn(4,5);
```

- `rand`: This function takes two values in the input denoting the size of the requested matrix and outputs a random matrix taking values from a uniform distribution on [0,1].

```
A = rand(2,2);
```

- `disp`: This function takes a variable input and outputs the value of the variable to the command window.

  Input:

```
A = [1,2;2,3];
disp(A)
```

  Output:

```
1 2
2 3
```

- `magic`: This function returns an n-by-n matrix constructed from the integers 1 through n^2 with equal row and column sums. The order n must be a scalar greater than or equal to 3.
  Input:

```
B = magic(3);
disp(B)
```

  Output:

```
8     1     6
3     5     7
4     9     2
```

- `blkdiag`: This function constructs a block diagonal matrix from input arguments.
  Input:

```
C = [1 1; 1 1];
D = [2 2; 2 2];
BD = blkdiag(C,D);
disp(BD)
```

Output:

```
 1     1     0     0
 1     1     0     0
 0     0     2     2
 0     0     2     2
```

- `exp`: This function takes in a constant, vector or matrix and outputs the base e for each *entry*. `exp(A)`

- `expm`: This function takes in a matrix and outputs the base of the matrix. `expm(A)`

- `log`: This function takes in a constant, vector or matrix and outputs the base log for each *entry*. `log(A)`

- `logm`: This function takes in a matrix and outputs the base log of the matrix. `logm(A)`

- `^n`: This function takes a matrix to the nth power. `A^n`

- `.^n`: This function takes the nth power of the entries of a matrix or vector. `A.^n`

- `abs`: This function takes in a constant, vector or matrix and outputs the absolute value of the input. `abs(A)`

- `diag`: This function removes the diagonal from a matrix and outputs a vector of the diagonal entries or outputs a diagonal matrix from a vector. `diag(A)`

- `'`: This function takes the transpose of a matrix. `A'`

2.5 Classwork

1. Given a vector $v_1 = (1, 2, 3)$. Find a diagonal matrix A_1 such that v_1 is the diagonal of A_1.

```
v1 = [1 2 3];
A1 = diag(v1);
```

2. Compute the square of matrix A_1 and the matrix of squared entries of A_1. Is it the same matrix? Why or why not?

```
A1^2;
A1.^2;
```

3. Compute the matrix exponential of A_1 and the exponential of all the entries of A_1. Is it the same matrix? Why or why not?

```
expm(A1);
exp(A1);
```

4. Find a 6×6 block diagonal matrix B such that the first block is A_1 and second block is a magic matrix.

```
B = blkdiag(A1,magic(3));
```

5. Compute the square of matrix B and the matrix of squared entries of B. Is it the same matrix? Why or why not?

```
B^2;
B.^2;
```

6. Using the command `pascal()` to create a 5×5 Pascal matrix P.

```
P = pascal(5);
```

7. Using the command `toeplitz()` to create a 5×5 Toeplitz matrix C by using vector `[1 2 3 4 5]`.

```
C = toeplitz([1 2 3 4 5]);
```

8. Find the matrix polynomial $P^2 - 2P + I$. The scalar version is $x^2 - 2x + 1$.

```
P^2+2*P+eye(5);
```

9. Find the matrix polynomial $(P - I)^2$. The scalar version is $(x - 1)^2$. Is it the same matrix as the matrix from question 8?

```
(P-eye(5))^2;
```

10. Given a matrix $A = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}$. Find the matrix polynomial $A^2 - 3A - 28I$. The scalar version is $x^2 - 3x - 28$.

```
A = [1 6; 5 2];
A^2-3*A-28*eye(2);
```

11. Given vectors $v_1 = (1, 1, 1)$ and $v_2 = (3, 1, 2)$. State the difference between the results of command `v1*v2'` and `v1.*v2`.

```
One multiplies the two vectors together the other multiplies the entries of
the two vectors together.
```

12. Given two 2×2 matrices A, B. State the difference between the results of command `A*B` and `A.*B`.

```
One multiplies the two matrices together the other multiplies the entries of
the two matrices together.
```

13. Given an matrix C. Find the command for computing the matrix exponential of C.

```
expm(C)
```

14. Given an matrix D. Find the command for computing the scalar logarithm of D.

```
logm(D)
```

2.6 Questions

(1.)
a. Create a random integer matrix A of size 4x4 taking values from 5 to 15 and display it using the MATLAB function `disp`.

b. Slice row 1 of matrix A and store it in a variable `row1` then display it using `disp`.

c. Slice row 4 and store it in a variable `row4` then display it using `disp`.

d. Add `row1` and `row2` and store this sum in a new variable `sumrow` then display it using `disp`.

e. Create a new variable called `sumrow1` such that it contains the sum of the entries of `row1`. Display a message using the MATLAB function `fprintf` that reads

The sum of entries in row1 is **

where in place of ** you will have a number that is equal to the sum.

f. Slice column 2 of matrix A and store it in a variable `col2` then display it using `disp`.

g. Slice column 3 of matrix A and store it in a variable `col3` then display it using `disp`.

h. Add `col2` and `col3` and store this sum in a new variable `sumcol` then display it using `disp`.

i. Create a variable called `sumcol2` such that it contains the sum of the entries of `col2`. Display a message using the MATLAB function `fprintf` that reads

The sum of entries in col2 is **

where in place of ** you will have a number that is equal to the sum.

(2.)
a. Create a diagonal matrix from the diagonal entries of matrix A and display it in the command window.

b. Take the transpose of matrix A and display it in the command window.

c. Create an upper triangular matrix from the values of matrix A by using the MATLAB function `triu`. Display this in the command window.

d. Create a lower triangular matrix from the values of matrix A by using the MATLAB

function `tril`. Display this in the command window.

e. Remove the diagonal entries from the matrix you created in question. Output this new matrix to the command window.

f. Create a block matrix where,

- A is the upper left
- 4 x 4 matrix of ones is in the upper right.
- 4 x 4 matrix of zeros is in the lower left.
- 4 x 4 identity matrix is in the lower right.

Be sure to suppress output!

g. Create a 2 x 2 matrix that consists of the following entries from the output matrix in question f.

$$\begin{bmatrix} a_{2,4} & a_{2,5} \\ a_{3,4} & a_{3,5} \end{bmatrix}$$

Display this matrix in the command window.

h. Create a block diagonal matrix consisting of the following matrices,

- A random 3 x 3 matrix of integers.
- A 2 x 2 matrix of all entries equal to 8.
- The matrix $B = \begin{bmatrix} -5 & -6 & -9 \\ -4 & -4 & -2 \end{bmatrix}$

Display this matrix in the command window.

i.(**) Create a random 3x3 matrix or integers, A, and show by computation that A is equal to the direct sum of its symmetric and skew parts.

(3.)

a. A 3 by 3 matrix A0 with only one as entries.

b. A 3 by 3 matrix A1 with only 2 as entries.

c. A 3 by 3 matrix A2 with only 3 as entries.

d. Create a 9 by 9 matrix with the 3 above matrices,A0, A1, and A2, on the diagonal.

e. Compute the matrix of the squared entries of A.

f. Compute the square of the matrix A. Is it the same matrix?

(4.)

a. Using the MATLAB command 'pascal' construct a lower triangular Pascal matrix B of order 5. Hint:Check the MATLAB's help.

b. Compute the exponential of all the entries of B.

c. Compute the matrix exponential of B.

d. Take the absolute value of the matrix B and matrix of the logarithm of the entries of B.

e. Take the absolute value of the matrix B and compute the matrix logarithm B.

f. Compute the matrix of the identity matrix plus the matrix of logarithm of the absolute values of the entries of B.

g. Compute the matrix logarithm of the identity matrix plus the absolute value of the matrix B.

(5.)(**)

a. Construct a Toeplitz matrix C using the vector [2 1 0 -1].

b. Compute the inverse of this matrix C.

c. Compute the square root of this matrix C.

d. Compute the matrix polynomial of order 1. The scalar version is $1 + x$.

e. Compute the matrix polynomial of order 2. The scalar version is $1 + x + x^2$.

f. Compute the matrix polynomial of order 5. The scalar version is $1+x+x^2+x^3+x^4+x^5$.

g. Recompute the above polynomial of degree 2 and 5 using `.^` instead of `^`? Display the two matrices.

h. Are they the same than in question 4., 5. ? Explain why they are different and which between `.^` instead of `^` should be used to create a matrix polynomial.

Chapter 3

Systems of Linear Equations

3.1 Linear Equation

Definition 1. *Let $a_1, ..., a_n$ be constants and $x_1, ..., x_n$ be variables in $\mathbb{R}$ then*

$$a_1x_1 + ... + a_nx_n = b$$

is a linear equation where $b \in \mathbb{R}$ is either a constant or the product of a constant and the first power of a single variable.

3.1.1 Examples (Linear)

Ex. $4x_1 - 5x_2 + 2 = x_1 \rightarrow$ rearranged $\rightarrow 3x_1 - 5x_2 = -2$

Ex. $x_2 = 2(\sqrt{6} - x_1) + x_3 \rightarrow$ rearranged $\rightarrow 2x_1 + x_2 - x_3 = 2\sqrt{6}$

3.1.2 Examples (Non-Linear)

Ex. $4x_1 - 6x_2 = x_1x_2$

Ex. $x_2 = 2\sqrt{x_1} - 7$

3.2 Linear System

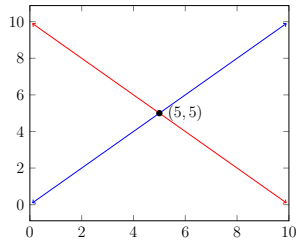
Definition 2. *A list $(s_1, s_2, ..., s_n)$ of numbers that makes each equation in the system true when the values $s_1, s_2, ..., s_n$ are substituted for $x_1, x_2, ..., x_n$ respectively is a solution to a system of linear equations. Solutions to linear systems fall into one of three categories:*

- Exactly one solution (*consistent*)
- Infinitely many solutions (*consistent*)
- No solution (*inconsistent*)

3.2.1 Examples of Linear Systems

Ex. Two Equations in two variables (each equation determines a line in 2-space.)

$$\textbf{(a)} \quad \begin{aligned} x_1 + x_2 &= 10 \\ -x_1 + x_2 &= 0 \end{aligned} \qquad \textbf{(b)} \quad \begin{aligned} x_1 - 2x_2 &= 3 \\ 2x_1 - 4x_2 &= 8 \end{aligned}$$



(a)

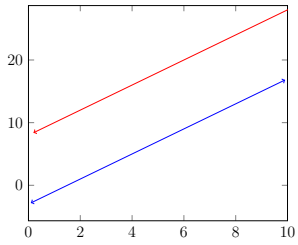

(b)

Figure 3.1: (a) exactly one solution and (b) no solution.

Ex. Three Equations in three variables (each equation determines a plane in 3-space.)

$$\textbf{(a)} \quad \begin{aligned} 2x_1 + x_2 + 3x_3 &= 1 \\ 4x_1 + 2x_2 + 2x_3 &= 2 \\ 4x_1 + 3x_2 + 5x_3 &= 3 \end{aligned} \qquad \textbf{(b)} \quad \begin{aligned} 2x_1 + x_2 + 3x_3 &= 7 \\ 4x_1 + 2x_2 + 2x_3 &= 2 \\ 2x_1 + x_2 + x_3 &= 5 \end{aligned}$$

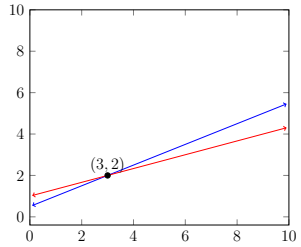
3.3 Equivalent Systems

Definition 3. *The solution set of a linear system is the set of all possible solutions of a linear system.*

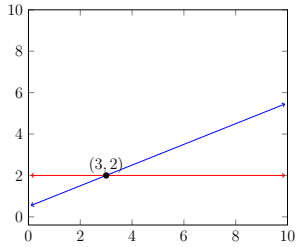
Definition 4. *Two linear systems are equivalent systems if they have the same solution set.*

3.3.1 Examples of Equivalent Systems

$$\textbf{(a)} \quad \begin{aligned} x_1 - 2x_2 &= -1 \\ -x_1 + 3x_2 &= 3 \end{aligned} \qquad \textbf{(b)} \quad \begin{aligned} x_1 - 2x_2 &= -1 \\ x_2 &= 2 \end{aligned}$$



(a)



(b)

3.4 Solving a Linear System in Matrix Form

Definition 5. *Given a system of n equations where x and b are vectors in* $\mathbb{R}^n$*,*

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2$$

$$...$$

$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n$$

the coefficient matrix form of this system is given by

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix}$$

and the augmented matrix form is given by

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} & b_1 \\ a_{21} & a_{22} & ... & a_{2n} & b_2 \\ ... & ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} & b_n \end{bmatrix}$$

We may solve a system in matrix form by performing *elementary row operations* on the augmented matrix. An elementary row operation is given by one of the following,

- Replacement: add one row to a multiple of another row.
- Interchange: interchange two rows.
- Scaling: multiply all entries in a row by a nonzero constant.

Row operations are complete when we either reach a triangular form for the matrix (solve for at least one variable) or reach a diagonal form for the matrix (solve for all variables).

In the following chapters, we will learn to use MATLAB to answer two fundamental questions,

1. Does there exist a solution to the system of equations?
2. What is the solution?

Chapter 4

Using MATLAB to Solve Systems of Linear Equations

4.1 Finding the Solution

In order to use any programming language to solve systems of linear equations, we must be able to write them in a matrix/vector format. To do this, we utilize linear algebra and the equivalent matrix form of a linear system. Suppose we want to solve the following linear system.

$$\begin{aligned} x_1 + 2x_2 + x_3 &= 1 \\ -x_1 + 2x_2 + 2x_3 &= 2 \\ x_1 - x_2 + 4x_3 &= 4 \end{aligned}$$

Our equivalent definition tells us that this is the same as the augmented matrix form,

$$\begin{pmatrix} 1 & 2 & 1 & 1 \\ -1 & 2 & 2 & 2 \\ 1 & -1 & 4 & 4 \end{pmatrix}$$

We may solve this system by performing step-by-step elementary row operations on our augmented matrix, however there is a simpler way in the form of a MATLAB built-in function. Instead of taking time to perform each row operation step-by-step, we can use `rref` to reduce our augmented matrix to its row reduced echelon form.

```
A = [1,2,1,1;-1,2,2,2;1,-1,4,4];
B = rref(A);
disp(B)
```

The output of this code with give us this following,

```
1 0 0 0
0 1 0 0
0 0 1 1
```

and we may conclude that the solution to this system is $x = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Remark 2. *Use your knowledge from the previous sections to determine why this is the output.*

4.2 Existence of a Solution

Suppose we would like to determine whether a solution exists (consistency) for the following system.

$$\begin{aligned} 3x_2 - 6x_3 &= 8 \\ x_1 - 2x_2 + 3x_3 &= -1 \\ 5x_1 - 7x_2 + 9x_3 &= 0 \end{aligned}$$

We may proceed in the same way as the previous section. The following code,

```
A = [0 3 -6 8; 1 -2 3 -1; 5 -7 9 0];
B = rref(A);
disp(B)
```

returns the output,

```
1 0 -1 0
0 1 -2 0
0 0  0 1
```

and we conclude that there is no solution to the above system of linear equations.

Remark 3. *Use your knowledge of linear algebra to determine why the system is inconsistent (there is no solution).*

4.3 More useful functions for the following questions

- `fprintf`: This function (introduced in Chapter 2) can output multiple variables.

  Input:

  ```
  x = 1;
  y = 2;
  fprintf('Our values are %d and %d',x,y)
  ```

  Output:

  ```
  Our values are 1 and 2.
  ```

- `\n`: This notation can be used in the `fprintf` function to "go to the next line".

  Input:

  ```
  x = 1;
  y = 2;
  fprintf('Our value for x is %d \n Our value for y is %d', x,y)
  ```

  Output:

  ```
  Our value for x is 1
  Our value for y is 2
  ```

4.4 Questions

1. Create the augmented matrix for the following system of equations,

$$\begin{aligned} 4x_1 + 7x_2 + x_3 &= 9 \\ 2x_2 - x_3 &= 3 \\ x_1 - 5x_2 + 4x_3 &= 2 \end{aligned}$$

2. Write code to determine whether the solution exists. If it does exist, create a vector representing the solution.

3. Output your answer from 2. to the command window in sentence format using the `fprintf` function. If the solution exists, make sure your solution vector is contained in the output.

4. Repeat 1-3 for the following system of equations.

$$\begin{aligned} 3x_1 + 7x_2 + x_3 &= 9 \\ 2x_2 + 10x_3 &= 0 \\ 4x_2 + 20x_3 &= 6 \end{aligned}$$

5. Write code to determine whether the following systems of equations are equivalent. Output your answer to the command window using `fprintf`.

$$\begin{aligned} (1)\quad 4x_1 + 2x_2 &= 1 \\ 2x_1 - x_2 &= 2 \end{aligned} \qquad \begin{aligned} (2)\quad 8x_1 + 4x_2 &= 2 \\ 4x_2 - 2x_3 &= 4 \end{aligned}$$

Chapter 5

Vector and Matrix Equations

In this chapter, we will discuss some key results from linear algebra on vector and matrix equations. We will use these results to understand the fundamental concept $Ax = b$.

5.1 Vector Equation

The geometric description of a vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ in $\mathbb{R}^2$ is the point (x_1, x_2) in the plane, where $\mathbb{R}^2$ is the set of all points in the plane.

It is useful to consider geometric interpretations when dealing with spans of a set of vectors, linear combinations of vectors, linear transformations, change of basis and many more concepts in linear algebra.

Proposition 1. *Parallelogram Rule: If u and v in $\mathbb{R}^2$ are represented as points in the place, then $u + v$ corresponds to the fourth vertex of the parallelogram whose other vertices are 0, u and v. (Note that $0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$)*

Definition 6. *Given vectors $v_1, v_2, ..., v_p$ in $\mathbb{R}^n$ and scalars $c_1, c_2, ..., c_p$, the vector y defined by*

$$y = c_1v_1 + c_2v_2 + ... + c_pv_p$$

is called a linear combination of $v_1, v_1, ..., v_p$ using weights $c_1, c_2, ..., c_p$.

Proposition 2. *A vector equation*

$$x_1a_1 + x_2a_2 + ... + x_na_n = b$$

has the same solution set as the linear system whose augmented matrix is

$$[\ a_1\ a_2\ ...\ a_n\ b\ ]$$

Moreover, b can be generated by a linear combination of $a_1, a_2, ..., a_n$ if and only if there is a solution to the linear system corresponding to the augmented matrix.

5.2 Matrix Equation

Definition 7. *If A is an mxn matrix, with columns $a_1, ..., a_n$ and if x is in $\mathbb{R}^n$ then the product of A and x, denoted by Ax is the linear combination of columns of A using the corresponding entries in x as weights, i.e.,*

$$Ax = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} = x_1a_1 + x_2a_2 + ... + x_na_n$$

Theorem 3. *If A is a mxn matrix, with columns $a_1, ..., a_n$ and if b is in $\mathbb{R}^n$ then the matrix equation*

$$Ax = b$$

has the same solution set as the vector equation

$$x_1a_1 + x_2a_2 + ... + x_na_n = b$$

which, in turn, has the same solution set as the system of linear equations whose augmented matrix is

$$\begin{bmatrix} a_1 & a_2 & ... & a_n & b \end{bmatrix}$$

Theorem 4. *Suppose $Ax = b$ is consistent for some given b, and let p be a solution. Then the solution set of $Ax = b$ is the set of all vectors of the form $w = p + v_h$ where v_h is any solution of the homogeneous equation $Ax = 0$.*