

**APM 153 LECTURE FIVE** – Calculating the Determinant, Cramer’s Rule, Methods of Checking Your Solutions, Exact vs. Approximate Solutions, Algorithm Development.

**Calculating the Determinant of a Matrix**

(1) The **determinant** of a matrix is an expression of the **relationship** between all the elements in a square matrix. Like inverses, only **square** matrices have determinants.

(2) This relationship between the elements of an matrix is **conserved**. It is an **intrinsic characteristic** of the matrix. For example, the determinant of the product of two matrices is equal to the product of the two determinants of the two original matrices.

(3) The determinant of a matrix is calculated as a **series of comparisons** of subsets of the matrix to determine the relationship between the elements.

(4) The easiest determinant to calculate is that of a 2 X 2 matrix as shown below.

Given the matrix A,                      The determinant |A|, is calculated as,..

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad |A| = (a * d) - (b * c)$$

(5) The determinant is particularly useful for calculating the inverse of a matrix.

Again, given the matrix A,                      The inverse of A is calculated as,..

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad A^{-1} = (1/ |A|) * \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

(6) In order to calculate the inverse you must multiply the rearranged matrix by 1 over the determinant. Can you see why there can be no inverse if the determinant is equal to zero?

(7) If the calculated value of the determinant is equal to zero, we say the matrix is **singular**, and does not have an inverse.

(8) In Assignment Two, Part Two, the matrices A and C were **singular**. What **error message** did Matlab give when you tried to calculate the inverse of A and C ?

## Cramer's Rule

(9) As explained in Assignment Two, Cramer's Rule is based upon the relationship between the coefficients in a matrix and the results. We can take advantage of this relationship to solve for the unknowns by calculating determinants.

(10) To use Cramer's Rule we first **replace** one of the columns in the matrix of coefficients with the column vector from the **results**.

(11) Given the system of coefficients from Assignment Two, Part One, Number 3,...

$$\begin{array}{rcl} 17x + 13y + 21z & = & 3 \\ 18x + 14y + 22z & = & 2 \\ 19x + 15y + 24z & = & 1 \end{array} \quad \begin{array}{l} \text{the matrix of the coefficients and the column} \\ \text{vector of the results are shown below.} \end{array}$$

$$A = \begin{bmatrix} 17 & 13 & 21 \\ 18 & 14 & 22 \\ 19 & 15 & 24 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

(12) The first column in the matrix A represents the coefficients of **x**. If we replace that first column with the values from the column vector b, we create what is known as an **augmented matrix**, "with respect to x", which we represent as  $A_{aug\_x}$ .

$$\text{if } A = \begin{bmatrix} 17 & 13 & 21 \\ 18 & 14 & 22 \\ 19 & 15 & 24 \end{bmatrix} \quad \text{then} \quad A_{aug\_x} = \begin{bmatrix} 3 & 13 & 21 \\ 2 & 14 & 22 \\ 1 & 15 & 24 \end{bmatrix}$$

(13) Finally, we can calculate the value of x as the ratio of the determinants of the two matrices.

$$x = |A_{aug\_x}| / |A| \quad (\text{In Matlab } x = \det(A_{aug\_x}) / \det(A) )$$

$$(14) \text{ Likewise, we calculate } \begin{array}{l} y = |A_{aug\_y}| / |A| \\ z = |A_{aug\_z}| / |A| \end{array} \quad \text{and}$$

## Methods of Checking Your Results in Matlab.

(15) Certainly the slowest method to check your results would be to solve each series of equations by hand. What is the name of the method where we solve for one unknown at a time using “elimination”?

---

(16) A much faster way of checking your results in Matlab would be to store values of unknowns in a vector and then multiply the unknowns by the original matrix A as in the lines of Matlab code below. The **vector c** is set up to store the unknowns.

```
>> c = inv(A) * b
```

```
>> A * c
```

The result of **A\*c** should be the same as the values in the **vector b**.

(17) A third method to check your results would be to plug the values of the unknowns back into one of the original equations and see if you get the expected result.

(18) This method does work but, sometimes Matlab gives a result that at first may seem incorrect. Take for instance the series of linear equations in Part One, Problem Two from Assignment Two.

(19) If we put the coefficients from this series of linear equations into a matrix A and the results into a column vector b and then solve for the unknowns,...

$$A = \begin{bmatrix} 3 & 7 & 2 & -9 \\ 0 & 1 & 1 & 1 \\ -2 & 0 & -3 & 1 \\ 10 & 1 & -2 & -1 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 7 \\ -5 \\ 18 \end{bmatrix} \quad \text{inv}(A) * b = \begin{bmatrix} 2.0967 \\ 2.6598 \\ 1.2867 \\ 3.0535 \end{bmatrix}$$

(20) If we plug the calculated values for the unknowns back into the first equation, do we get the expected result of 0? Remember, the original equation was  $3w + 7x + 2y - 9z = 0$

$$(3 * 2.0967) + (7 * 2.6598) + (2 * 1.2867) - (9 * 3.0535) = \mathbf{6.0000e-004}$$

The answer is close to, but not exactly zero (0.0006). Why?

(21) The reason why Matlab does not return a value of zero is because of **rounding error**.

(22) We could get Matlab to return a more precise result by forcing Matlab to use higher precision.

```
>> format long
```

```
>> inv(A)*b
```

```
ans =
```

```
2.09671848013817  
2.65975820379965  
1.28670120898100  
3.05354058721934
```

(23) This time when we plug in the unknowns Matlab return a value of **zero**. Yeah!

### **Exact vs. Approximate Solutions**

(24) One of the most important concepts to understand in mathematics is that the solution to many if not most equations is an **approximation** and not an exact answer.

(25) Take for example the number 10 divided by 6. The answer is  $1.6\overline{6666}$ , which is a repeating decimal. When we report this number we usually either show it with the line above the last six or we just cut off the extra decimal places and round the last digit up.

(26) **We** decide how precise our answer needs to be. Or, our calculators decide for us.

(27) If you use your calculator regularly you may fall into a false sense of precision. You must remember that many solutions and commonly used numbers such as Pi are approximations.

(28) One important part of **algorithm development** therefore will always be deciding at the beginning of the algorithm what level of precision is acceptable.

## Algorithm Development

(29) We will continue to learn about algorithm development by following Chapter Three in your textbook which starts on page. 85. (Please read from pg 85 to 107 this week.)

(30) Section 3.1 is titled “*Introduction to Top-Down Design Techniques*”. What do you think is meant by “top-down”?

(31) Let’s go back to our previous discussion concerning algorithms from the first week of class. What were some of the steps we listed in an creating an algorithm?

---

---

---

---

(32) Once we have figured out what steps we need to take for the algorithm we can make a formal chart of those steps. What was the name of that type of chart?

(33) Another method for writing out an algorithm is to use **pseudocode**. Pseudocode uses just words instead of flowchart symbols to represent the steps in an algorithm.

(34) The reason why it is called “pseudocode” is because each line of the algorithm will read almost like lines of code from a real program. For example, if at a particular place in the algorithm we need to enter some data, we might write a line of code like,...

*Enter the coefficients a, b, and c.*

(35) Or, if the algorithm reaches a decision point, we might write,..

*If  $a+b$  is greater than 0, then stop.*

(36) Finally, if the algorithm has reached the point where it has found a solution, we might write,..

*Write final value out to screen.*

(37) Your textbook describes pseudocode as “*a hybrid mixture of MATLAB and English*”.

(38) The textbook goes on to say that “*It (ie pseudocode) is structured like MATLAB, with a separate line for each distinct idea or segment of code, but the descriptions on each line are in English*”.

(39) Read Chapter Three carefully to understand what is meant by the following terms.

pseudocode	top-down design	relational operators
unit testing	structured program	beta release
decomposition	stepwise refinement	

(40) Chapter Three does a really nice job of introducing algorithm development, pseudocode and then shows you how to convert an algorithm into Matlab code.

(41) On pages 106 and 107, the textbook gives an example of a program written in Matlab to solve for the roots of a quadratic equation based on the algorithm discussed in the chapter.

(42) We will type this program into the Matlab Editor to get a feel for how Matlab code works and to try out some of the ideas discussed in the Chapter.

(43) To open the Matlab Editor, click on the “white page” icon on the Matlab Toolbar.

(44) Type in the program exactly as it appears on pages 106 and 107, with the exception of **use your own name** under where it says **programmer**.

(45) Save the program as **calc\_roots.m** on your USB drive.

(46) On Wednesday, during lecture, I will review how **calc\_roots.m** is supposed to work.

(47) Also on Wednesday, be prepared to answer questions based Chapters 1, 2, and 3.