This lab explores some of MATLAB’s capabilities for working with matrices and their application. Pay attention to the various methods for **creating**, **addressing**, **editing** and **applying** matrices.

The name MATLAB actually stands for MATrix LABoratory because this was the original focus of the software. Today it can do much more, but is still one of the best tools available for working with matrices. Each student must try out their own commands. However, asking for help and comparing your results with other students is encouraged.

**I. The two-loop circuit – Variations on the solution**

The two-loop circuit problem covered in class and studio results in equations 1 and 2 as a starting point

(1)

(2)

The basic matrix equation for a linear system is: ***AX = b*** Where A is the coefficient matrix, x is the variable column vector, and b is the constant column vector. Therefore, the matrix setup would be

1. **Creating:** Create the coefficient matrix (A) and the column vector of constant terms (b) in MATLAB.   
   Reminders: • Use square brackets [ ] to indicate the values are in a vector or matrix

• A space (or comma) is used to separate different numbers in the same row

• A semicolon, when inside square brackets, indicates that a new row is being started.

1. **Applying** - Solution Option 1: Using the matrix inverse

In ENGR 127 we solved linear systems analytically by determining the inverse matrix *A-1*. So the solution would be ***x = A-1 b***. The command to do this in MATLAB is:

>> x = inv(A)\*b or >> x =A(-1)\*b (Try one of these and write the results below)

***I1  =*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***I2*** = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

N.B., to see more significant figures for *I2* try: >> x(2)

(Note: you could also use the MATLAB command linsolve(A, b) and get the same result)

1. **Applying** - Solution Option 2: Left Division (i.e. a back slash).   
   In MATLAB you can use the backslash (i.e., ***\*** which is typically on the key above the ***Enter*** key) to solve this matrix equation. It is important to note that this is another case where what is legal and helpful in the computer syntax would not be a correct mathematical expression (where matrix division must be done with an inverse). However, this approach is considered the preferred approach n MATLAB because it uses a more accurate background calculation. The command for doing this in MATLAB is:

>> x = A\b (Try this one and compare to the results in 2 above)

1. Addressing, Editing and Applying - Solution Option 3: Cramer’s Rule  
   Applying Cramer’s Rule to this problem gives us a chance to try out some matrix calculations and manipulations. Remember that in Cramer’s rule a variable’s column in the coefficient matrix is replaced by the constant vector. The ratio of this new matrix’s determinant over the determinate of the original coefficient matrix gives the variables value. Try this in MATLAB using the following steps:
   1. Create two copies of the coefficient matrix (A in this handout), one for each variable.   
        
      For example in MATLAB: >> A1 = A; A2 =A;
   2. Now we can replace the first column in one of the copies (e.g., A1) with the constant vector

**Addressing** matrices is always done by giving the number of the row and then the number of the column. For example A(1, 2) means the value in the first row and the second column. In MATLAB to address an entire row or column instead of a particular number you use a colon (:). For example A(:, 2) means all rows in the second column. This addressing can be used on the RHS (Right Hand Side) of an expression to select a specific value from the matrix or on the LHS (Left Hand Side) as the address of the target for an assignment (i.e., the exact location where the answer is to be saved).

**Editing:** To replace the first column in A1 with the b vector the command would be:

>> A1(:,1) = b

* 1. **Editing:** Replace the second column in A2 with b following a similar procedure.
  2. **Applying** Cramer’s Rule:  
     Now all that is left is to take the determinates of these matrices and form the necessary ratios. For this problem and. Straight lines around a matrix indicate a determinate. In MATLAB the function for a determinant is ***det(A)*** – calculate the solution in MATLAB using this approach.

**II. Problems in Paradise – when all this does not work (Applying):** There are problems that do not work out well by this or any other solution method because they are “singular”. Singular means that the matrix does not have one unique solution. Here we examine a couple of these cases.

1. Problem Problem
   1. To look at this, we will adjust our original coefficient matrix, A. Use array addressing to replace Matrix A row 2, column 2 value with 600.   
      Write the command you used here \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. Solve this new problem by the method of your choice.   
      What are the resulting values of the two currents?
   3. So what happened?   
      To look at this: rewrite equations 1 and 2 with the second coefficient in equation 2 set to 600 instead of 3400. Solve these equations for *I1* as a function of *I2* (i.e., so each equation is of the form *I1 = mI 2 + b* ). Write the solved equations below.

*I1* = f (*I2*) = (1)

*I1* = f (*I2*) = (2)

Notice the slope and intercept of the two equations. What do they tell you about the nature of this system?

What do you observe this time? **Show your results so far to your instructor or TA**

**An Important Aside:** Transposing of a matrix or vector

Transposing a matrix or vector means reversing its rows and columns. In other words, the first column of the original matrix becomes the first row of the transposed matrix.

1. To transpose a matrix in MATLAB you simply add an apostrophe (‘) after the matrix. Transpose one of your modified matrices and look at the arrangement before and after (e.g. use the command >> AT = A1’).
2. Transposing a vector will turn a row vector into a column vector or a column vector into a row vector. The constant vector (b) could be defined for the problem by >> b = [-10 -7]’ (Try it).

**III. Forces in a Truss – A large array problem.** (Adapted from Cleve Moler, *Experiments with MATLAB*, MathWorks, Inc., <http://www.mathworks.com/moler>, accessed 1/20/2015)

Figure 1 depicts a plane truss having 13 members (the numbered lines) connecting 8 joints (the numbered circles). The indicated loads, in tons, are applied at joints 2, 5, and 6, and we want to determine the resulting force on each member of the truss.

For the truss to be in static equilibrium, there must be no net force, horizontally or vertically, at any joint. Thus, we can determine the member forces by equating the horizontal forces to the left and right at each joint, and similarly equating the vertical forces upward and downward at each joint. For the eight joints, this would give 16 equations, which is more than the 13 unknown factors to be determined. For the truss to be statically determinate, that is, for there to be a unique solution, we assume that joint 1 is rigidly fixed both horizontally and vertically and that joint 8 is fixed vertically. Resolving the member forces into horizontal and vertical components and defining *α* = 1*/√*2, we obtain the following system of equations for the member forces *fi*:

**Figure 1:** Diagram of a truss. Circles are the connection nodes. The bottom numbers represent the applied loads in tons.

Horizontal (Σ left = Σ right) Vertical (Σ up = Σ down)   
 i.e., the x – direction i.e., the y-direction

Joint 2: *f*2 = *f*6*, f*3 = 10

Joint 3: *αf*1 = *f*4 + *αf*5*,* 0= *αf*1 + *f*3 + *αf*5

Joint 4: *f*4 = *f*8*,* 0= *f*7

Joint 5: *αf*5 + *f*6 = *αf*9 + *f*10*, αf*5 + *f*7 + *αf*9 = 15;

Joint 6: *f*10 = *f*13*, f*11 = 20;

Joint 7: *f*8 + *αf*9 = *αf*12*,* 0 = *αf*9 + *f*11 + *αf*12

Joint 8: *f*13 + *αf*12 = 0*.*

🡺 Review the above balances to see how the forces balance at each node.

To solve this system of equations in MATLAB, use the following steps:

1. Set up the equations so you can clearly determine the coefficient matrix and the constant vector. Moving all the constants to RHS of each equation and the variables to the LHS the result is:

Horizontal (Σ left = Σ right) Vertical (Σ up = Σ down)   
 i.e., the x – direction i.e., the y-direction

Joint 2: *f*2 - *f*6 *= 0 f*3 = 10

Joint 3: *αf*1 - *f*4 - *αf*5*, = 0 αf*1 + *f*3 + *αf*5=0

Joint 4: *f*4 - *f*8 *= 0 f*7= 0

Joint 5: *αf*5 + *f*6 - *αf*9 - *f*10 *= 0 αf*5 + *f*7 + *αf*9 = 15

Joint 6: *f*10 - *f*13 *= 0 f*11 = 20;

Joint 7: *f*8 + *αf*9 - *αf*12 *= 0 αf*9 + *f*11 + *αf*12 = 0

Joint 8: *αf*12 + *f*13= 0

1. Fill in Table 1 (on the last page of this handout) based on these rearranged equations. The first two lines in the table have been filled in as an example (showing how the two joint 2 equations are entered into the table. Continue filling in the table for the other equations.
2. **Creating/Editing:** Create the coefficient matrix in MATLAB. For such a large (and sparse matrix) it will be easiest to do this using MATLAB’s interactive variable editor. Working in the Command Window complete the following steps.
   1. Setup the value of α in MATLAB (e.g., ).
   2. Create an empty variable for the coefficient matrix by assigning square brackets with nothing in them to your desired variable name (e.g. >> M = [ ]).
   3. Open the variable editor window by double clicking on the variable you created in the ***Workspace*** window.
   4. Filling in the array: The variable editor window looks a bit like a spreadsheet. Fill in your coefficient matrix terms as follows.
      * Fill in the first line: Start by putting a zero in column 13; this will fill the row with zeros. For the columns where the first balance equation has a term replace the zero with the term.
      * Fill in the Second line: Follow a similar procedure for successive lines; Filling in the non-zero terms for each row.
      * Continue filling in the Rows: You can use the variable created earlier when filling in terms that have alpha in their coefficient (i.e., use "a" for alpha where needed, MATLAB will fill in the value)
      * When coefficient array is complete, you can close variable editor.
   5. Double Check: Look at the Workspace window. It should show that your coefficient matrix is 13 x 13. You should also review all the terms for errors. One way to do this is to maximize command window and then enter the variable name in the command window so that the array is echo printed.
3. Create the constant vector (b). You choose how. When complete make sure it is a column vector with 13 elements.
4. Solve this system using any of the four approaches we have covered (linsolve, left division, matrix inverse or Cramer’s Rule).

**Complex Numbers & Calculations in MATLAB**

Overview: You can create complex variables using i or j to represent (both are internal constants in MATLAB). Use the regular operators (+ - \* /) to carry out calculations with complex numbers.

Rectangular (Cartesian) Form: >> x = 1 + 2j The default format in MATLAB

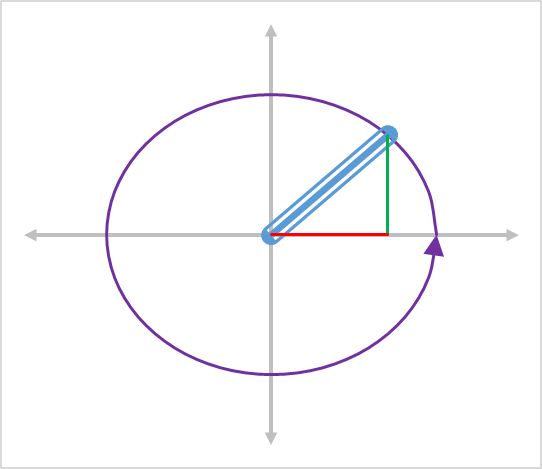
Eulerian (Exponential) form: >> x = 2.2\*exp(1.11i)

(1) explicitly include all operators except for a number times i or j (ex: 3i and b\*i)

(2) use the function exp(c) for ec.

(3) i must follow the number (N.B., i3 will be interpreted as a variable)

**For: x = a + bi or x = M eiθ**



Real Axis

Imaginary Axis

M

θ

b

a

|  |  |  |
| --- | --- | --- |
| **Converting Forms: x = a + bi 🡺 x = M eiθ** | | |
| **Item** | **Math. Formula** | **MATLAB Function** |
| Magnitude(M) |  | >> abs(x) |
| Angle (θ) | arctan(b/a)  (+ quadrant) | >> angle(x) |

|  |  |  |
| --- | --- | --- |
| **Converting Forms: x = M eiθ 🡺 x = a + bi**  (MATLAB converts automatically) | | |
| **Item** | **Math. Formula** | **MATLAB Function** |
| Real component (a) | M cos(θ) | >> real(x) |
| Imaginary component(b) | M sin(θ) | >> imag(x) |

bb

**Practice Exercises (try these out to see how commands work, not for turn in)**

**Command Line Exercises – try the following expressions in MATLAB in the command window**

Enter these two complex: >> x = 4+3i >> y = 5 – 5j;

Add and Subtract >> x + y >> x - y

Multiply and Divide >> x\*y >> 1/x >> x/y

Values for Eulerian Form >> M = abs(x) >> theta = angle(x)

Entering Eulerian Form >> z = 5 \* exp(0.6435j)

**Assignment: Due:** Before the beginning of the next lab

1. **Truss Matrix Problem** (the “Forces in a Truss” problem)

Deliverable - Present the solution to this problem including:

1. Table 1 filled in for this problem (Excel).
2. The MATLAB commands to solve the matrices clearly showing which matrix solution technique was used (it is not necessary to show how the matrices where created).
3. The resulting command window answer. (13 forces – the solution)
4. A well formatted table (Excel) of the above forces, the member number for each and the units.
5. **Complex Number Script Exercise:**

Write a script to calculate the magnitude and angle of a complex number.

* **Input:** One input: a single complex number in cartesian form (i.e., of the form a + b i )

Use interactive input (i.e., the input function) to have user input the complex number

* **Output:** Two outputs: magnitude (M) and angle(θ) – in degrees  
   Clearly label the output using the disp command.
* **Comments:** Introduction: Program name & purpose; your name & the date   
   Variables: Define the name, nature and units of all script variables.   
   Organize these variables as input, output and intermediate variables  
   Program Logic: Clearly comment on steps including input, calculations and output;   
   Label output with the ***disp*** function
* **Validation:** test with 1 + 2i from above (Eulerian form: M ~ 2.236 θ ~ 63.4o)

**Deliverable:** A copy of the script and the command window execution session (including script call, interactive input and output result for the test case)

**Rubrics Available on the Lab website   
(do look at them)**

**Table 1:** Coefficient Matrix (A) and Constant Vector (b) for the Forces in a Truss Problem.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Joint** | **Direction** | **A, The Coefficient Matrix** | | | | | | | | | | | | | | **Y, The Constant Vector** | |
| **f1** | **f2** | **f3** | **f4** | **f5** | **f6** | **f7** | **f8** | **f9** | **f10** | **f11** | **f12** | **f13** |  | |
| **1** | **2** | X | 0 | **1** | 0 | 0 | 0 | **-1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **2** | Y | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **10** | |
| **3** | **3** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **4** | Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **5** | **4** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **6** | Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **7** | **5** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **8** | Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **9** | **6** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **10** | Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **11** | **7** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **12** | Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **13** | **8** | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |

**Turn in the Filled in Table with the Problem Solution**