MATLAB Workshop 13 - Linear Systems of Equations

Objectives: Create a script to solve a commonly occurring problem in engineering: linear systems of equations.

MATLAB Features:

colon (`:`) operator - a special vector matrix operator

Notation	Action	Result
`a:c`	creates vector with elements **`1`** apart	**`[a  a+1  a+2  ...  c]`**
`a:b:c`	creates vector with elements **`b`** apart	**`[a  a+b  a+2b ...  (≤c)]`**
`A(m,:)`	selects mth row of **`A`**	
`A(:,j)`	selects jth column of **`A`**	
`A(m:n,:)`	selects mth through nth rows of **`A`**	
`A(:,j:k)`	selects jth through kth columns of **`A`**	
`A(m:n,j:k)`	selects mth through nth rows of **`A`** and jth through kth columns of **`A`**	

matrix functions

number of rows and columns	**`[rows, cols] = size(x)`**

workspace command

Command	Action
`varname = load(filename)`	retrieves the contents of a file as ASCII text and assigns the contents to **`varname`**. **`filename`** is a string that specifies the name and location of the file. if the file is not in the current directory, the entire path must be specified.

- **Linear systems of equations and matrix algebra**

Matrix algebra provides a convenient shorthand notation for linear systems of equations. Consider the system of equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

where the a_{ij} represent the coefficients of the unknown variables x_i. This system of equations can be written in matrix algebra notation as

$$[A]_{N\ x\ N}(X)_{N\ x\ 1} = (B)_{N\ x\ 1}$$

where **[A]$_{\mathbf{NxN}}$** is a square (same number of columns as rows) matrix of coefficients, **(X)$_{\mathbf{Nx1}}$** is a column vector of unknowns, and **(B)$_{\mathbf{Nx1}}$** is the "forcing" vector (also a column vector). Note that the multiplication is defined within the rules of matrix algebra., i.e., an (NxN) matrix times an (Nx1) vector yields an (Nx1) vector.

In expanded notation, the matrix algebra equation can be written as

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

You can verify that applying the rules of matrix multiplication to the above matrix algebra equation reproduces the original set of equations. Note that the coefficient matrix is obtained by taking the coefficients of each equation and placing them in the corresponding row.

The following example illustrates the use of matrix algebra for linear systems of equations. The set of linear equations

$$\begin{aligned} 32x + 47y - 2z &= 125 \\ 15x + 12z &= 346 \\ 6x - 16y + 15z &= 225 \end{aligned}$$

can be written in matrix algebra form as

$$\begin{bmatrix} 32 & 47 & -2 \\ 15 & 0 & 12 \\ 6 & -16 & 15 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 125 \\ 346 \\ 225 \end{pmatrix}$$

Note that one of the coefficients in the second equation is zero. This needs to be noted explicitly in the coefficient matrix.

- **A Civil Engineering problem**

Linear systems of equations arise frequently in every engineering discipline. For example, consider the following problem that might arise in a road construction project.

A civil engineer needs to purchase 85 tons of road material consisting of 25 tons of gravel, 40 tons of pebbles, and 20 tons of stones. She contacts three local quarries and obtains the following specifications on their product mix.

Product Mix	**Quarry, wt%**		
	1	**2**	**3**
Gravel	40	20	0
Pebbles	50	25	50
Stones	10	55	50

None of the quarries individually has product available in the percentages that are required for the project. So the next question she asks is whether she can meet her requirements through buying some product from each quarry. To do this, she sets up material balances for the amounts of gravel, pebbles and stones she requires. If she purchases Q_1 tons from Quarry 1, Q_2 tons from Quarry 2, and Q_3 tons from Quarry 3, then a material balance for gravel, obtained by multiplying the fraction of gravel from a quarry by the total tons to be purchased from the quarry, becomes

$$(0.40)Q_1 + (0.20)Q_2 + (0.00)Q_3 = 25$$

where the total amount of gravel purchased from the three quarries must be equal to the amount needed for the project. Likewise, material balances for pebbles and stones yield the equations

$$(0.50)Q_1 + (0.25)Q_2 + (0.50)Q_3 = 40$$
$$(0.10)Q_1 + (0.55)Q_2 + (0.50)Q_3 = 20$$

These equations can be put into matrix algebra form as

$$\begin{bmatrix} 0.40 & 0.20 & 0.00 \\ 0.50 & 0.25 & 0.50 \\ 0.10 & 0.55 & 0.50 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \begin{bmatrix} 25 \\ 40 \\ 20 \end{bmatrix}$$

MATLAB can be used to solve this system of equations for Q_1, Q_2, and Q_3.

- **Solution of linear equations in MATLAB**

Solution of a set of linear equations in MATLAB is rather straightforward. In matrix algebra terminology the solution to the set of linear equations

$$[A](x) = (b)$$

where *A* is the coefficient matrix, *x* is the vector of unknowns, and *b* is the right hand side vector is given by (in MATLAB terminology)

```
» x = A\b
```

That is, the **`backslash`** operator, \, is used to find the solution.

<u>(1) Solving a system of linear equations in MATLAB.</u>

To solve a system of linear equations, one needs the coefficient matrix, the right hand side vector, and the **backslash** operator, ****. To find the solution to the quarry problem, enter the following at the command prompt:

```
» Coeff = [0.40, 0.20, 0.00; 0.50, 0.25, 0.50; 0.10, 0.55, 0.50];
» amount = [25; 40; 20];
» Q = Coeff\amount
Q =
   57.5000
   10.0000
   17.5000
```

The first line entered the coefficient matrix, **Coeff**, by rows. The right hand side vector, **amount**, was entered as a column vector. This is very important. What happens if a row vector is used instead? Try it! The solution vector, **Q**, was obtained by using the **backslash** operator. Note that **Q** is a column vector, as would be expected (why?).

- **Creating an augmented coefficient matrix file**

While solution of linear systems of equations is rather straightforward in MATLAB, doing so in the Command Window has a couple of drawbacks. The first drawback is in entering the coefficient matrix. This is rather easy for a system of three equations. However, systems of linear equations frequently consist of hundreds of equations. Entering the coefficient matrix in the Command Window for such a large system is not only tedious, but is fraught with the potential for error. What would happen if one coefficient in the 10,000 required coefficients for a system of 100 equations is entered incorrectly? Would you even catch such an error? Similar issues are encountered with entering the right hand side vector in the Command Window.

To avoid these problems, engineers (and others) typically create text files with the coefficients. Rather than have two text files for a problem (one containing the coefficients and the other containing the right hand side vector), the two sets of numbers are generally combined into a single file called the ***augmented coefficient matrix*** file. The augmented coefficient matrix file contains the coefficient matrix in rows and columns with the right hand side vector appended as the $(n+1)^{th}$ column, where n is the number of equations in the system. For example, the augmented coefficient matrix for the quarry problem would look like

0.40	0.20	0.00	25
0.50	0.25	0.50	40
0.10	0.55	0.50	20

where the first three rows and columns are the coefficient matrix and the fourth (last) column is the right hand side vector. A file containing these numbers has all of the information necessary to get find a solution to the set of equations.

(2) Creating a text file with the augmented coefficient matrix.

There are several methods to create an augmented coefficient matrix file. You could use your favorite text editor such as Notepad or Wordpad (these will automatically save as an ASCII text file). If you use a word processor, such as Word, you will need to specify that the file is to be saved as ASCII text when you save it. You could also use a spreadsheet, such as Excel, to create the file. Using a spreadsheet has the advantage that each coefficient can be loaded into a separate cell. You could then *export* the file as ASCII text.

Pick a method and create an ASCII file with the quarry problem coefficients. Save it as **`quarry.dat`** in your active MATLAB directory. The **`.dat`** extension for a file is frequently used to denote an ASCII file that contains data.

- **Loading an augmented file into the Workspace**

An augmented coefficient matrix is easily loaded into the MATLAB Workspace by using a variation of the load command described in Workshop 4. The particular form of the command to load a data file is

```
varname = load(filename);
```

where **`varname`** is the name assigned to hold the augmented coefficient matrix values and **`filename`** is a string that identifies the name and location of the file.

(3) Loading a text file containing an augmented coefficient matrix.

Load the augmented coefficient matrix values into the Workspace by entering

```
» augmat = load('quarry.dat')
augmat =
    0.4000    0.2000         0   25.0000
    0.5000    0.2500    0.5000   40.0000
    0.1000    0.5500    0.5000   20.0000
```

Why did the values display? How would you suppress display?

Notes on **`filename`**

- **`filename`** can be *hardwired* as shown above (not very useful).
- **`filename`** can be *a variable name* that contains a string as shown here:

```
fname = 'quarry.dat';
augmat = load(fname);
```

or, better yet,

```
fname = input('Please enter augmented matrix file name ==> ','s');
augmat = load(fname);
```

- If the required file is located in the current directory, only the file name (with extension) is required.
- If the required file is not in the current directory, the entire path must be specified, i.e., **`c:\temp\quarry.dat`**.

- **Extracting the coefficient and right hand side vector from the augmented matrix**

Once the augmented coefficient matrix is in the Workspace, we need to ***extract*** the coefficient matrix and the right hand side vector. Although we have been working with a system of three equations and three unknowns, in general, a system of linear equations may be of any size. Fortunately, MATLAB provides the tools to handle this.

(4) Extracting the coefficient matrix and right hand side vector.

Determine the size of the system of equations by entering

```
» [nrows, ncols] = size(augmat)
nrows =
     3
ncols =
     4
```

The number of rows, **nrows**, tells the number of equations in the augmented coefficient matrix. **ncols** should be one greater than **nrows**.

Complete the following MATLAB commands to extract the coefficient matrix and right hand side vector from the augmented matrix

```
        » Coeff = augmat(????)
        Coeff =
            0.4000    0.2000         0
            0.5000    0.2500    0.5000
            0.1000    0.5500    0.5000
        » rhs = augmat(????)
        rhs =
            25
            40
            20
```

Refer to Workshop 12 if necessary.

- **Create a script for solving systems of equations**

Systems of linear equations occur so frequently in engineering that having a user friendly script for their solution would be useful. Remember the general paradigm for script development:

(1) get information
(2) do computations
(3) report results.

Because MATLAB can solve a system of linear equations with a single assignment statement, designing a function to get the necessary information (i.e., a problem name, coefficient matrix and right hand side vector) is the most complex part of designing the script. The output function design is also rather straightforward.

(5) Design an information input function for a linear equation solver script.

Following the design methodology of the previous workshops, design a function that will
1) ask the user for the name of the problem to be solved
2) ask the user for the name/location of the augmented coefficient matrix file
3) load the augmented coefficient matrix file
4) extract the coefficient matrix
5) extract the rhs vector
The function should return the problem name, coefficient matrix, and rhs vector

(6) Solve linear system of equations.

Because MATLAB solves a system of linear equations with a single assignment statement, there is no need to design a separate function for this task.

(7) Display the results.

Design a function that will display the results. To be user friendly, the results from solving the system of linear equations should have a label prior to display. From where might an appropriate label come? How might it be passed to this function for use?

(8) Create the script **`lineqsolver`**.

You now have the requisite functions to create a script that puts it all together. Create a script, **`lineqsolver.m`**, that does the following

- displays script header information (call to **`lineqsolver_header`**)
- gets the necessary input information (call to **`lineqsolver_info`**)
- solves the system of equations (simple assignment statement)
- displays the results (call to **`lineqsolver_results`**)

Be sure your script has an appropriate header section and variable dictionary.

Bonus: Add a **`while`** loop to your script so that the script asks the user whether to repeat for another system of linear equations and, if the answer is yes, repeats the input, calculation, display portion of the script.

Exercises: Use your script, **`lineqsolver`**, to solve the following problems.

1. Six people, Alice, Barb, Carol, Dean, Eric, and Fred, each have a bag of money. Find the amount in each bag if:
   - The total of all six bags is $33.56;
   - Carol has twice as much money as Alice;
   - Barb and Carol together have as much as Dean;
   - Alice and Barb together have as much as Eric;
   - Eric's amount subtracted from Fred's amount is twice Alice's amount; and
   - $1.15 is left after subtracting Barb's amount and Eric's amount from Fred's amount.

2. Find the five-digit number that satisfies the following properties:
   - The sum of all the digits is 18;
   - The third digit is the sum of the first and second digits;
   - Subtracting the third digit from the fourth digit yields 4;
   - The first and third digits added together give the fifth digit; and
   - The first digit is twice the second digit.

3. Find the six-digit number that satisfies the following properties:
   - The first and third digit sum to 10;
   - The sum of the second, third, and fifth digits equals the fourth digit;
   - The third and last digits are the same;
   - The sum of the second and third digit equals the fifth digit;
   - The fourth digit is the same as the last digit minus the first digit; and
   - The sum of all the digits is 22.

4. Five people, Alice, Ben, Cindy, Dean, and Evan, decide to invest in the stock market. What is each person's profit after one year if:

- The sum of all their profits is $299.25;
- Evan’s profit is three times as large as Dean’s profit;
- Ben’s and Cindy’s profits together total to Dean’s profit;
- Alice and Cindy together made $89.20 profit; and
- Alice’s profit was $101.55 more than Dean’s profit.

Recap: You should have learned

- What an *augmented coefficient matrix* is and its relation to a system of linear equations.
- How to use the **`load`** command to retrieve an augmented coefficient file.
- How to use the **`size`** command to determine the number of equations.
- How to extract the coefficient matrix and rhs vector from the augmented coefficient matrix.
- How to use the **`backslash`** operator, **`\`**, to solve a system of equations.