EE202 Lab#2

These problems are for the most part derived from the examples in the text (Higham & Higham) from Chapter 5 “Matrices.”

Creating Matrices

1. [5.1] Enter the following examples.
   a. Spaces or commas separate entries in a row. *Try both ways of creating a matrix*
      ```
      >>  a = [5.1 3.5 2.7]  or a = [5.1, 3.5, 2.7]
      ```
   b. Semicolons or carriage returns ↲ separate rows (i.e., new column).*Try both ways of creating a matrix.*
      ```
      >> b = [4;5;6] or >> b = [4
                                5
                                6]
      ```
   c. Use Space and Semicolon to create two-dimensional matrices.
      ```
      >> b = [-3 0 -1;2 5 -7;-1 4 8]
      ```
   d. You can continue entries by using a comma and an ellipse.
      ```
      >> F=[1 4 5 ...
      55 56]
      ```

[5.1, 5.2] Use eye(m,n), speye(m,n) *s = sparse*[1], zeros(m,n), ones(m,n), rand(m,n), randn(m,n) *n = negative*, to quickly create useful matrices. Eye generates an identity matrix. If only one value is specified a square matrix is generated.

2. Enter the three matrices (zeros, ones, eye) found on page 45.

[5.1, 5.2] Rand function generates random entries between 0 and 1. To set the seed (i.e., initial state) of the random number generator type `>> rand('state',j)`. Without any arguments `rand()` returns a scalar value (i.e, one number). To generate random number over a given range use the formula `min + round(max*(rand(m,n))`.

3. Use the rand function to initialize the state of the random number generator. Then show that you can generate the same scalar random number twice in one Matlab session.

4. Create a matrix with 2 rows and 5 columns containing random numbers between 1 and 52 (inclusive).

[5.2] Use a **colon** to generate an array of equally spaced values m:s:n, where m is start value, s is step (or stride) size (increment or decrement), and n is “less than or equal to.” Note that you do not need to explicitly declare an array using brackets [], although it also works with them. If only one colon is used then s =1. For example `>> 1:3 ans=1 2 3`.

5. To practice using colon notation, enter the three examples shown on pages 50 and 51.

[1] A sparse matrix is a MATLAB sparse storage organization where zero entries are removed thus saving memory.

[5.2, 5.4] Another useful trick when creating an array is to use colon notation to generate a linear array containing the elements you want and then the reshape function in conjunction with the transpose operator to shape the matrix.

6. Using a colon and reshape function, create both matrices A on page 54 and B on page 56.
   a. A =
```
        1       2
        3       4
```
   b. B =
```
        1       2       3
        4       5       6
        7       8       9
```

[5.2, 5.4] Two other useful functions for creating arrays are block diagonal blkdiag, replicate and tile array repmat, and the upper and lower triangular part triu, tril.

7. Applying the functions you have learned so far (ones, zeros, eye, reshape, …) construct the following matrices out of smaller ones using bracket notation (pages 5 and 47*). Do not simply generate these matrices by entering them into Matlab (use functions).*
   a. A =
```
    1       2       0       0
    3       4       0       0
    1       1       1       0
    1       1       0       1
```
   b. B =
```
        1       1       0       0
        1       1       0       0
        0       0       2       0
        0       0       0       2
```
   c. C =
```
        2       0       2       0       2       0
        0       2       0       2       0       2
        2       0       2       0       2       0
        0       2       0       2       0       2
```
   d. D =
```
        1       2       3       0
        4       5       6       0
        7       8       9       0
        1       1       1       1
```
   Tip: To create this matrix you will need the transpose function or operator.

8. Starting with the matrix created at the end of the last problem (Matrix D) generate the following two matrices (Hint: read page 59)
```
a. A =
        1       0       0       0
        4       5       0       0
        7       8       9       0
        1       1       1       1
b. B =
        1         2         3         0
        0         5         6         0
        0         0         9         0
        0         0         0         1
```

[5.2] If you want to generate a vector of equally spaced numbers between a and b, use the linspace(a,b,n) function, where n is the number of points rather than the step size. Notice Matlab also moved it to the last argument versus the second just to confuse you. If you want to generate a vector of logarithmically spaced values between 10^a and 10^b, use the logspace(a,b,n) function , where n is again the number of points rather than the step size.

9. a. Use the linspace function to generate a linear array **d** with three evenly spaced numbers between 1 and 10 (inclusive).

   b. Use the logspace function to generate the logarithmic array **d** where `d = [10 100 1000 10000]`.

Assigning Values to an Matrix

[5.2, 5.4] When assigning a value(s) to an array, the size of the array on the left and right side of the assignment operator (=) must be the same. One exception is placing a single element on the right-hand side, in which case the specified elements (i.e., submatrix or the whole matrix) on the left-hand side will be assigned this value.
Assign an empty matrix [] to the row or column of an array to delete it.

10. Create the following frame matrix by first creating a 4x4 matrix of ones and then assigning zero to the interior values of the matrix.

    a. Ans =

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

    b. Modify matrix A to generate the following matrix.

```
    Ans =
          1         1         1         1
          1         0         0         1
          1         1         1         1
```

Accessing element(s) of an Array

[5.2, 5.4] Here is a simple example of how to access an element of an array `b(2,3) ans = -7`. Note that arrays are One-based.

You can also access elements of an array, creating a submatrix, by finding the intersection of rows p to q and columns r to s as denoted by A(p:q,r:s). A special case is a lone colon in the row or column specifying all rows or all columns, thus A(:,j) is the jth column of A. Use end keyword to specify the last index of the specified dimension, thus A(end,:[2]) gets the last row of A. To get the first row you could type A(1,:). *A little reflection*. In "Creating Arrays" we discovered that the colon was used to

[2] A colon without arguments is shorthand for 1:end

generate an array – it is doing the same thing here. For example the expression A(p:q,r:s) is equivalent to A([p p+1 … q],[r r+1 … s]). This can help you understand some confusing example in the book, notably this one on page 52.

11. a Generate a matrix of primes using the `primes` function and the `reshape` function plus any other techniques you have learned up to this point. Do not generate the matrix as shown at the top of page 47.

   b. to h. Starting with your 3x3 matrix of primes, type in the 7 examples provided on page 52.

Just as the reshape function allows us to generate an `m-by-n` matrix from a liner array; to linearize a, `m-by-n` matrix you can use this special case A(:) of the colon operator.

   i. Generate the following linear array from the matrix of primes. Tip Use the transposition operator or function to get a single row.

```
I =
     2     7    17     3    11    19     5    13    23
```

   h. On one line (3 statements separated by a semicolon), generate the following 4x4 matrix of primes by creating a 4x4 matrix of zeros; filling it A(:) with prime numbers; and finally taking the transpose (‵) of the resulting array (Hint: see page 53).

```
A =
     2     3     5     7
    11    13    17    19
    23    29    31    37
    41    43    47    53
```

12. Create the following matrices, and use them in the exercises that follow:

$$a = \begin{matrix} 15 & 3 & 22 \\ 3 & 8 & 5 \\ 14 & 3 & 82 \end{matrix} \qquad b = \begin{matrix} 1 \\ 5 \\ 6 \end{matrix} \qquad c = [12 \;\; 18 \;\; 5 \;\; 2]$$

   a. Create a matrix called **d** from the third column of matrix a.
   b. Combine matrix b and matrix d to create matrix **e**, a two-dimensional matrix with three rows and two columns
   c. Combine matrix b and matrix d to create matrix **f**, a one-dimensional matrix with six rows and one column.
   d. Create a matrix **g** from matrix a and the first three elements of matrix c, with four rows and three columns.
   e. Create a matrix **h** with the first element equal to a[1,3], the second element equal to c[1,2], and the third element equal to b[2,1]