Armstrong State University
Engineering Studies
MATLAB Marina – Linear Algebraic Equations Primer

Prerequisites

The Linear Algebraic Equations Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, arrays, logic expressions, conditional structures, iteration, functions, and debugging. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, MATLAB Marina Arrays module, MATLAB Marina Logic Expressions module, MATLAB Marina Conditional Structures module, MATLAB Marina Iteration module, MATLAB Marina Functions module, and MATLAB Marina debugging module.

Learning Objectives

1. Be able to create matrices using MATLAB.
2. Be able to index matrices using MATLAB.
3. Be able to perform arithmetic operations on matrices using MATLAB including: addition, subtraction, multiplication, transpose, and inverse.
4. Be able to determine when to use scalar, matrix, and element by element operations.
5. Be able to solve systems of linear equations using the matrix form of the equations and MATLAB.

Terms

scalar, matrix, identity matrix, matrix addition, matrix subtraction, matrix multiplication, matrix transpose, matrix inverse, system of linear equations, nonsingular matrix, determinant, Gaussian elimination (back division)

MATLAB Functions, Keywords, and Operators

+, -, *, ‘, \, size, inv, det, cond, eye, ones, zeros

Matrices

Recall that an array is a multi-dimensional collection of data of the same data type. A matrix is also a 1D or 2D collection of data of the same type. One dimensional matrices are typically referred to as vectors. A 1 x m matrix is a row vector and an m x 1 matrix is a column vector.

$$vRow = \begin{bmatrix} v_1 & v_2 & \cdots & v_m \end{bmatrix}$$

$$vCol = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

The two-dimensional matrix V below has n rows and m columns (m times n total elements). The element in row p and column q is referred to as the pq element of the matrix.

$$V = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & & v_{2m} \\ \vdots & & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{pmatrix}$$

Matrices are treated the same as arrays in MATLAB. Matrices are created, accessed, and modified in the same way that arrays are. Generally when we are performing matrix operations, such as matrix multiplication or matrix inverse, on an array we will refer to the array as a matrix, otherwise we will refer to it as an array.

The Identity matrix, denoted by I_n, is a square matrix with ones along its diagonal and zeros everywhere else. The MATLAB function `eye` will create an identity matrix. The statement `eye(M)` creates a M by M identity matrix and `eye(N,M)` creates a N by M matrix with 1s on the diagonal and zeros everywhere else. The N by M matrix generated by the `eye` function is not an identity matrix as it is not square.

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
>> eye(2,3)
ans =
     1     0     0
     0     1     0
```

Figure 1, MATLAB `eye` Function

Operations on Matrices

Most built in MATLAB functions will accept matrices as arguments since matrices are arrays. Generally MATLAB functions perform their operation on each element in the argument and return a result the same size as the argument.

The MATLAB `length` and `size` functions work for matrices just as for arrays; the `length` function returns the largest dimension of the matrix and the `size` function returns the dimensions of the matrix.

Logic operations can be performed on elements, portions, or the entire matrix just as with arrays. Logic operations can be performed on two matrices of the same size or a matrix and a scalar, just as for arrays. Generally when performing logical operations, we would refer to the data collection as an array not a matrix.

Two matrices A and B are equal (A = B) if the matrices are the same size and all their corresponding entries are equal. The two matrices A and B below are equal if $a_{11} = b_{11}$, $a_{12} = b_{12}$, $a_{21} = b_{21}$, and $a_{22} = b_{22}$.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

The MATLAB equality operator == performs element by element comparison for equality and so it is not a matrix equality operator. The equality operator compares all of the corresponding elements of the two matrices (or a matrix to a scalar) and returns an array of Booleans of the same size as the matrix.

```
>> F = [0, 1, 2 ; 3, 4, 5];
>> G = [2, 3 ; 5, 6];
>> F == G
??? Error using ==> eq
Matrix dimensions must agree.
>> G = [2, 2, 2; 5, 6, 7];
>> F == G
ans =
     0     1     0
     0     0     0
>> F == 3
ans =
     0     0     1
     0     0     0
```

Figure 2, Logic Equality Operator Applied to Matrices

Arithmetic Operations on Matrices

Array (element by element) arithmetic operations (addition, subtraction, element by element multiplication, element by element division, element by element power) can be performed on all the elements of an matrix as long as the operation involves two matrices of the same size or a matrix and a scalar. MATLAB also supports matrix addition, matrix subtraction, matrix multiplication, and matrix power. Generally when performing element by element operations, we would refer to the data collection as an array not a matrix.

The sum of two matrices A and B is written A + B and the difference is written A - B. Matrix addition is commutative but matrix subtraction is not. The sum (or difference) of two matrices is a matrix of the same size whose entries consist of the sum (difference) of the two corresponding entries of the two matrices. Matrix addition and subtraction can only be performed on matrices of the same size. For the 2 x 2 matrices A and B below

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

The sum and difference of the matrices are

$$A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix} \text{ and } A-B=\begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} \\ a_{21}-b_{21} & a_{22}-b_{22} \end{pmatrix}$$

Matrix addition and subtraction are defined the same way as element by element addition and subtraction so there is not a separate dot operator for the element by element versions of these arithmetic operators.

The product or multiplication of two matrices A and B is written AB or A*B. Matrix multiplication is not commutative. Matrix multiplication is defined for matrices where the size of the inner dimensions match, i.e. the number of columns of first matrix equals the numbers of rows of the second matrix. An $m\times p$ matrix A can be multiplied by a $q\times n$ matrix B if $p=q$. The result of multiplying an $m\times p$ matrix A by a $p\times n$ matrix B is a $m\times n$ matrix AB whose ij entry is the ith row of the matrix A multiplied by the jth column of matrix B.

The product of a $1\times n$ row vector and a $n\times 1$ column vector is a 1×1 matrix (a scalar)

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}=a_1b_1+a_2b_2+\cdots+a_nb_n=\sum_{k=1}^{n}a_kb_k$$

The product of two 2×2 matrices is a 2×2 matrix

$$A=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ and } B=\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}, \text{ then } AB=\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

```
>> F = [0, 1, 2 ; 3, 4, 5];
>> G = [2, 3 ; 5, 6];
>> F*G
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> G*F
ans =
     9    14    19
    18    29    40
```

Figure 3, Matrix Multiplication Example

Matrix multiplication is not the same operation as element by element multiplication. The matrix multiplication operator is * and the element by element multiplication operator is the dot multiply .*.
Multiplying a matrix by the appropriate sized identity matrix does not change the matrix, i.e. the result is the original matrix.

Figure 3 shows a MATLAB function for matrix multiplication. It is better to use the MATLAB matrix multiplication operator than your own matrix multiplication function.

```
function result = multiplyMatrix(A, B)
% obtain dimensions of A and B
[nrowA, ncolA] = size(A);
[nrowB, ncolB] = size(B);
result = [];
% check that dimensions compatible for matrix multiplication
if (ncolA == nrowB)
    result = zeros(nrowA, ncolB);
    for k1 = 1 : 1 : nrowA
        for k2 = 1 : 1 : ncolB
            % multiply each element of A in row k1
            % by each element of B in column k2, element k1,k2
            % of result is sum of the multiplications
            for k3 = 1 : 1 : ncolA
                result(k1,k2) = result(k1,k2) + A(k1,k3)*B(k3,k2);
            end
        end
    end
else
    disp('Inner dimensions do not agree');
end

end
```

Figure 3, `multiplyMatrix` Function

MATLAB supports two types of matrix division (\ and /) along with the two types of array division (.\ and ./). Matrix division using \ and / should generally not be used as they do not correspond to legal linear algebraic matrix operations. One exception to this is the case where you have a square matrix A and a column vector b, in which case x = A\b using left matrix divide (matrix back division) gives the solution to Ax = b using Gaussian elimination.

Of the two array division operators, generally one should use the right divided version (./), where A./B divides each entry of A by the corresponding entry of B or divides all elements of A by B when B is a scalar.

Raising a matrix A to a power n is written A^n. Matrix power is only defined for square matrices and the power must be a scalar. The matrix power A^n is equivalent to multiplying the matrix A by itself $n-1$ times. Matrix power is a postfix operator (operator applied at end of the variable to operate on) and is not the same operation as element by element power. The matrix power operator is ^ and the element by element power operator is the dot power .^.

Matrix Transpose and Matrix Inverse

The transpose of matrix changes the rows of a matrix to columns and the columns of a matrix to rows. The conjugate transpose operator is a single quote and is a postfix operator and takes the conjugate of all complex valued elements as well as performing the transpose operation. The non-conjugate transpose operator is a period followed by a single quote and only performs the transpose operation. When the matrix consists of only real valued numbers it does not matter which transpose operator you use but it is good style to use the conjugate transpose only when the conjugate of values is intended.

The transpose of a row vector is a column vector and the transpose of a column vector is a row vector.

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_m \end{pmatrix}' = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}' = \begin{pmatrix} a_1 & a_2 & \cdots & a_m \end{pmatrix}$$

The transpose of a 2×3 matrix $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$ is a 3×2 matrix $A' = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{pmatrix}$

```
>> F = [0  1  2 ; 3  4  5];
>> F.'
ans =
   0   3
   1   4
   2   5
```

Figure 4, Matrix Transpose Example

The inverse of a matrix A written A^{-1} is a matrix such that $AA^{-1} = A^{-1}A = I$, where I is the identity matrix. The inverse of a matrix is only defined for square matrices (number rows equals number of columns) that have full rank (determinant is non-zero). MATLAB has built in functions `inv` and `det` for computing the inverse and determinant of a matrix.

MATLAB allows matrix division, but this is not a legal linear algebraic operation so avoid using this. Multiply by the matrix inverse on the appropriate side instead of performing matrix division.

```
>> A = [1 0 4; -3 1 -1; 0 7 1];
>> det(A)
ans = -76
>> inv(A)
ans =
   -0.1053   -0.3684    0.0526
   -0.0395   -0.0132    0.1447
    0.2763    0.0921   -0.0132
>> A*inv(A)
ans =
    1.0000   -0.0000    0.0000
   -0.0000    1.0000   -0.0000
    0.0000    0.0000    1.0000
```

Figure 5, Matrix Inverse Example

Solving Systems of Linear Equations

A system of n linear equations in n unknowns can be written in the matrix form $Ax = b$, where A is a n by n square matrix of coefficients, x is an n by 1 column vector of unknowns, and b is a n by 1 column vector of constants. Systems of linear equations will have a unique solution when the number of independent equations and unknowns are equal. For example the system

$3x - y = 4$

$x + y = 0$

has as its unique solution, x = 1, y = -1. The MATLAB code of Figure 6 determines the solution to this system of linear equations.

```
A = [ 3   -1; 1   1];
b = [ 4 ; 0];
x = inv(A)*b;
```

Figure 6, Solving System of Linear Equations

The coefficient matrix A of the matrix form of the system of linear equations is square when the number of equations and unknowns are equal. If the coefficient matrix A is square and has full rank (determinant is nonzero) then the matrix A has an inverse and the solution to the matrix form of the system of linear equations $Ax = b$ can be found via $x = A^{-1}b$. If the number of independent equations is greater than the number of unknowns the system is said to be over constrained and there may not be a solution. If the number of independent equations is less than the number of unknowns the system is said to be under constrained and there are an infinite number of solutions. The coefficient matrix A of the matrix form of the system of linear equations is non-square in these cases and the coefficient matrix is thus not invertible.

The MATLAB code of Figure 7a determines the solution to the system of linear equations

$2x - 3y + 4z = 9$

$x + 2y + 3z = 1.5$

$$-7x - y + 2z = -12$$

The matrix form of the equations is:

$$\begin{pmatrix} 2 & -3 & 4 \\ 1 & 2 & 3 \\ -7 & -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 9 \\ 1.5 \\ -12 \end{pmatrix}$$

The solution to this system of linear equations is: x = 2.0, y = -1.0, and z = 0.5.

```
A = [ 2   -3   4 ; 1   2   3 ; -7   -1   2 ];
b = [ 9 ; 1.5 ; -12 ];
x = inv(A)*b;
```

Figure 7a, Solving System of Linear Equations using Matrix Inverse

Systems of linear equations can also be solved using Gaussian elimination. The MATLAB back division operator \ can be used to perform Gaussian elimination to solve a system of linear equations. For an n by n square matrix A and an n by 1 column vector b, the equation $Ax = b$ can be solved using Gaussian elimination via $x = A \backslash b$. The MATLAB code of Figure 7b determines the solution of the system of linear equations using Gaussian elimination.

```
A = [ 2   -3   4 ; 1   2   3 ; -7   -1   2 ];
b = [ 9 ; 1.5 ; -12 ];
x = A\b;
```

Figure 7b, Solving System of Linear Equations using Gaussian Elimination

When solving systems of linear equations numerically, one must be careful that the problem is well posed. The condition number of a matrix is a measure of the sensitivity of the solution of a system of linear to errors in the equation coefficients. The condition number provides an indication of whether the inverse of the A matrix and the solution of the system of linear equations will be accurate. Condition numbers near one indicate a well condition matrix (the condition number of the identity matrix is one) and large condition numbers indicate that the matrix is near to being singular (and the numerical matrix inverse may not be accurate). See MATLAB's help on the cond function for more information.

Last modified Thursday, November 13, 2014