MATLAB II

Recall that MATLAB uses the symbol * for scalar multiplication. It uses the same symbol for multiplying a matrix and a vector.

>> M=[1 2; 3 4]; v=[1; -1];
>> M*v

ans =
  -1
  -1

If the matrix and the vector cannot be multiplied (i.e. the sizes don’t match), MATLAB gives an error.

In MATLAB, to retrieve the $i,j$th entry of a matrix, $M$, type $M(i,j)$. For example,

>> M=[1 2 3 4; 5 6 7 8; 9 10 11 12]

\[
M =
\begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12
\end{pmatrix}
\]

>> M(3,2)

ans =
  10

>> M(2,4)

ans =
  8

To retrieve more than one element at a time, give a list of the columns and/or rows that you want. For example, to retrieve the entries in row two, columns two and four, type

>> M(2, [2 4])

ans =
  6    8

If instead we wanted columns two through four, we can use
>> M(2,2:4)

ans =

   6   7   8

The following command extracts the submatrix made from rows 1 and 3 and columns 1,2, and 3.
>> M([1 3], 1:3)

ans =

   1   2   3
   9  10  11

A colon by itself retrieves all columns (or rows).
>> M(2,:)

ans =

   5   6   7   8

Thus, M(i,:) extracts the i
th row of M. Similarly, M(:,j) extracts the j
th column of M.
>> M(:,3)

ans =

   3
   7
  11

The transpose of a matrix is a new matrix whose i,j
th entry equals the j,i
th entry of the original matrix. In MATLAB, the transpose operator is '.
>> M=[1 2; 3 4; 5 6]

M =

   1   2
   3   4
   5   6

>> M'
ans =

    1   3   5
    2   4   6

The transpose operator can be useful when forming a matrix whose rows are certain vectors. For example,

\[ v = [1; 1; 1], \quad w = [-2; 2; 3] \]

\[ v = \]

\[ 1 \]
\[ 1 \]
\[ 1 \]

\[ w = \]

\[ -2 \]
\[ 2 \]
\[ 3 \]

\[ N = [v'; w'] \]

\[ N = \]

\[ 1 \quad 1 \quad 1 \]
\[ -2 \quad 2 \quad 3 \]

Notice that multiplying a vector \( u \) by \( N \) gives a vector with entries \( v \cdot u \) and \( w \cdot u \).

\[ u = [3; 2; 1]; \quad N \cdot u, \quad [\text{dot}(v, u); \text{dot}(w, u)] \]

\[ \text{ans} = \]

\[ 6 \]
\[ 1 \]

\[ \text{ans} = \]

\[ 6 \]
\[ 1 \]
Recall that the \textit{rank} of a matrix is the number of pivots in the corresponding reduced matrix. One way to compute the rank of a matrix is to use the MATLAB command \texttt{rref} and then count pivots. Alternately, MATLAB has a built-in command called \texttt{rank}.

\begin{verbatim}
A =

1 1 2
2 3 5
1 3 4

>> rref(A)

ans =

1 0 1
0 1 1
0 0 0

>> rank(A)

ans =

2
\end{verbatim}

We can use the command \texttt{rank} to determine whether a collection of vectors is linearly independent. \{\textbf{v}_1, \ldots, \textbf{v}_k\} is linearly independent precisely when the matrix with columns \textbf{v}_1, \ldots, \textbf{v}_k has rank \(k\). For example, the following three vectors in \(\mathbb{R}^3\) are linearly independent.

\begin{verbatim}
>> v1=[1; 3; -2; 1]; v2=[2; 2; -1; -1]; v3=[3; 1; 1; 1];

>> rank([v1 v2 v3])

ans =

3
\end{verbatim}

Calculating the rank also gives a quick way of deciding whether a given vector is in the column space of a matrix. Suppose a matrix \(M\) has rank \(r\). Then the rank of \(M\) augmented by \(\textbf{b}\) will either have rank \(r\) or \(r+1\). (Either the new column will have a pivot or it won't.) \(\textbf{b}\) is in the column space of \(M\) if and only if the rank of the augmented matrix is still \(r\). (Why?) For example, with \(A\) as above,

\begin{verbatim}
>> rank([A, [1; 1; 1]])
\end{verbatim}
ans = 
    3

>> rank([A, [1; 5; 7]])
ans =
    2

Therefore, \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix} is in the column space but \begin{bmatrix} 1 \\ 1 \end{bmatrix} is not.

EXERCISE. Suppose that \( A \) is a \( 5 \times 7 \) matrix and \( B \) is a \( 5 \times 5 \) matrix. Suppose that \( \text{rank}(A) = 4 \), \( \text{rank}(B) = 4 \), and \( \text{rank}([A,B]) = 4 \). What can we conclude? Why?

We can use MATLAB to compute bases for the null space and the column space of a matrix. However, there are subtleties to each procedure.

MATLAB has two commands for producing a basis for the null space. The command \texttt{null(M,'r')} uses reduced echelon form to compute the basis, just as we do when computing the basis by hand. The 'r' stands for "rational" but you can remember it as standing for "regular."

>> A=[1 2 2; 2 4 4]

A =

\[
\begin{bmatrix}
1 & 2 & 2 \\
2 & 4 & 4 \\
\end{bmatrix}
\]

>> B=null(A,'r')

B =

\[
\begin{bmatrix}
-2 & -2 \\
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

Notice that what we get is actually a matrix whose columns form the basis for the null space. To get the vectors individually, we use

>> B(:,1), B(:,2)

ans =

\[
\begin{bmatrix}
-2 \\
1 \\
0 \\
\end{bmatrix}
\]
-2
1
0

ans =

-2
0
1

The command `null(M)` (without the 'r') produces what is called an orthonormal basis for the null space. An orthonormal basis is a basis, \( \mathbf{u}_1, \ldots, \mathbf{u}_k \), which also satisfies \( \mathbf{u}_i \cdot \mathbf{u}_j = 0 \) for \( i \neq j \) and \( \mathbf{u}_i \cdot \mathbf{u}_i = 1 \). That is, the vectors in the orthonormal basis are all unit vectors and they are all perpendicular to each other.

>> N=null(A)

N =

-0.9428  0
0.2357  -0.7071
0.2357   0.7071

>> dot(N(:,1), N(:,1))

ans =

1

>> dot(N(:,1), N(:,2))

ans =

0

>> dot(N(:,2), N(:,2))

ans =

1
To get a basis for the column space of a matrix, we use a modification of the \texttt{rref} command. As we know, \texttt{R=rref(M)} produces the reduced echelon form of \( M \). If instead we type \texttt{[R,p]=rref(M)}, we also get a list of the pivot columns.

\[
\text{\texttt{M=[1 \ 2 \ 3; \ 2 \ 4 \ 1; \ 3 \ 6 \ 1]}}
\]

\[
\begin{array}{ccc}
1 & 2 & 3 \\
2 & 4 & 1 \\
3 & 6 & 1
\end{array}
\]

\[
\text{\texttt{[R,p]=rref(M)}}
\]

\[
\begin{array}{ccc}
1 & 2 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{array}
\]

\[
\begin{array}{c}
p = \\
1 & 3
\end{array}
\]

To get a basis for the column space, we extract those columns of \( M \) which correspond to the pivots.

\[
\text{\texttt{M(:,p)}}
\]

\[
\begin{array}{c}
\text{\texttt{ans =}} \\
1 & 3 \\
2 & 1 \\
3 & 1
\end{array}
\]