Matrix Algebra Tutorial
With Examples in Matlab

by

Klaus Moeltner
Department of Agricultural and Applied Economics
Virginia Tech
email: moeltner@vt.edu
web: http://faculty.agecon.vt.edu/moeltner/

Specifically designed as a 1/2 day "bare-bones" introduction to Matrix Algebra using Matlab for beginning Graduate Econometrics students

Comments & suggestions are always welcome!

Notes on Notation

Following standard convention (as used in econometrics) we will index the rows of a matrix by "*i*" (*i = 1..n*), and the columns by "*j*"(*j = 1:k)*. Thus, the total number of rows and columns are given by "*n*" and "*k*", respectively. For both row and column vectors elements will be indexed by "*i*" (*i = 1..n*).

Bold-faced notation will be used for vectors and matrices.

Additional notation will be introduced on a case-by-case basis.

Module I. Basic Vector and Matrix Operations

(see Matlab script script *ma_mod1*)

Vector & Matrix definitions and the Transpose operator

Vectors: n or $k = 1$

Special case: Scalar ($n = k = 1$)

Matrices: $n, k > 1$

Column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Row vector

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$$

Transpose operator

$$\mathbf{x}' = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\mathbf{y}' = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Matrix examples

n = 2, k = 2:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

n = 3, k = 2:

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{32} \end{bmatrix}$$

Transpose operator:

$$\mathbf{X}' = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{bmatrix} \qquad \mathbf{M}' = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \end{bmatrix}$$

Matrix addition / subtraction:

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \qquad \mathbf{B}=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

$$\mathbf{A}+\mathbf{B}=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \end{bmatrix}$$

$$\mathbf{A}-\mathbf{B}=\begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & a_{13}-b_{13} \\ a_{21}-b_{21} & a_{22}-b_{22} & a_{23}-b_{23} \end{bmatrix}$$

Note: For this to work the two matrices must have **identical dimensions**, i.e. $n_A = n_B$, $k_A = k_B$

Special Matrices:
Square matrix: n = k (ex: matrix **X** above)
Diagonal matrix: A square matrix with non-zero elements on its diagonal, and zeros everywhere else. A special case is the **identity matrix I**, which has a diagonal of "1's".
In Matlab, simply use `eye(n)` to create an identity matrix of size *n*. (see script)
Symmetric matrix: A square matrix for which $m_{ij} = m_{ji} \quad \forall i, j$, i.e. elements above the diagonal are mirror images of elements below the diagonal. Example:

$$M = \begin{bmatrix} 1 & 2 & 5 & 0 \\ 2 & 3 & 1 & 44 \\ 5 & 1 & 1.8 & 8 \\ 0 & 44 & 8 & 11 \end{bmatrix}$$

For any symmetric matrix: **M' = M**

Vector and Matrix multiplication:
The simplest case is the multiplication of a row vector by a column vector:

$$\mathbf{a}=\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \qquad \mathbf{b}=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathbf{a}\times\mathbf{b}=\begin{bmatrix} a_1b_1+a_2b_2+a_3b_3 \end{bmatrix}$$

This is called the ***inner product*** of two vectors.
Rule: For this to work the *column dimension* of the first vector must be equal to the *row dimension* of the second, i.e. $k_a = n_b = n$.

Note: Whenever you multiply a row vector by a column vector, the result will be a **scalar** (= 1 x 1 matrix). In general, for row vector **a** and column vector **b** (where $k_a = n_b$), we get:

$$\mathbf{a}\times\mathbf{b}=\sum_{i=1}^{n} a_i b_i$$

Knowing this result allows us to perform any multiplication of *conformable* vectors and matrices.

Some general rules:

1. Matrices must be *conformable* for multiplication, i.e. the *column dimension* of the first matrix must be equal to the *row dimension* of the second, i.e. $k_A = n_B$
2. The resulting new matrix will be of dimension n_A by k_B
3. Denoting an individual cell in the resulting matrix for row *i*, column *j* as $m_{ij},$ use the following rule to fill these cells: *mij* = (row *i* of first matrix) * (column *j* of second matrix)

Examples:

Using **a** and **b** from above:

$$\mathbf{b}\times\mathbf{a}=\begin{bmatrix} b_1a_1 & b_1a_2 & b_1a_3 \\ b_2a_1 & b_2a_2 & b_2a_3 \\ b_3a_1 & b_3a_2 & b_3a_3 \end{bmatrix}$$

Using **X** and **M** from above:

$$\mathbf{M}\times\mathbf{X}=\begin{bmatrix} m_{11}x_{11}+m_{12}x_{21} & m_{11}x_{12}+m_{12}x_{22} \\ m_{21}x_{11}+m_{22}x_{21} & m_{21}x_{12}+m_{22}x_{22} \\ m_{31}x_{11}+m_{32}x_{21} & m_{31}x_{12}+m_{32}x_{22} \end{bmatrix}=$$

$$\begin{bmatrix} \text{first row of }\mathbf{M}\text{ x first column of }\mathbf{X} & \text{first row of }\mathbf{M}\text{ x second column of }\mathbf{X} \\ \text{second row of }\mathbf{M}\text{ x first column of }\mathbf{X} & \text{second row of }\mathbf{M}\text{ x second column of }\mathbf{X} \\ \text{third row of }\mathbf{M}\text{ x first column of }\mathbf{X} & \text{third row of }\mathbf{M}\text{ x second column of }\mathbf{X} \end{bmatrix}$$

Note: Any vector or matrix can be pre- or post-multiplied by a scalar. Multiplication is then performed element-by-element. Using again **M** from above, and scalar *s*:

$$s\mathbf{M}=\begin{bmatrix} sm_{11} & sm_{12} \\ sm_{21} & sm_{22} \\ sm_{31} & sm_{32} \end{bmatrix}$$

Also, for any identity matrix **I** and conformable matrix **M** we have: **MxI** = **M** and / or **IxM** = **M** (thus the name "identity matrix").

Some important special cases:

For column vector **x** with *n* rows (or elements), we get $\mathbf{x}'\times\mathbf{x}=\sum_{i=1}^{n}x_i^2$.

For any two column vectors **a** and **b** of equal length, we can thus derive results such as:

$$(\mathbf{a}-\mathbf{b})'(\mathbf{a}-\mathbf{b})=\sum_{i=1}^{n}(a_i-b_i)^2$$

$$(\mathbf{a}+\mathbf{b})'(\mathbf{a}+\mathbf{b})=\sum_{i=1}^{n}(a_i+b_i)^2$$

The **i**-vector:
Consider a column vector of length *n* with all "1'"s. Such a vector is often denoted as **i**. Some useful results:

$$\mathbf{i}'\mathbf{i}=n$$

$$\mathbf{i}\mathbf{i}'=\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & 1 & \cdots & 1\\ \vdots & \vdots & \ddots & \vdots\\ 1 & 1 & \cdots & 1\end{bmatrix}$$ (*n* by *n* matrix of "1'"s)

For scalar s:

$$\mathbf{i}s=\begin{bmatrix}s\\ s\\ \vdots\\ s\end{bmatrix}$$

For **i**, and some column vector **x** of equal length *n*:

$$\mathbf{i}'\mathbf{x}=\sum_{i=1}^{n}x_i\ .$$

In Matlab, use `ones(n,1)` to create an **i**-(column) vector of length *n*.

Combining some of these results:
The summed deviations of the elements of column vector **x** from its mean can be expressed as:

$$\sum_{i=1}^{n}(x_i-\overline{x})^2=(\mathbf{x}-\mathbf{i}\overline{x})'(\mathbf{x}-\mathbf{i}\overline{x})$$

Useful Rules:

For conformable matrices **A**, **B**, **C:**

$$(\mathbf{A}\times\mathbf{B})'=\mathbf{B}'\times\mathbf{A}'$$

$$(\mathbf{A}\times\mathbf{B})\times\mathbf{C}=\mathbf{A}\times(\mathbf{B}\times\mathbf{C})$$

$$\mathbf{A}\times(\mathbf{B}+\mathbf{C})=\mathbf{A}\times\mathbf{B}+\mathbf{A}\times\mathbf{C}$$

Bending the Rules: The "Dot" operator in Matlab

Matlab's "dot" operator performs element-by-element multiplication and divisions for matrices that have identical dimensions. Strictly speaking, this is not a permissible "textbook procedure", but it can be very useful in programming. Example:

```
A =

     1     2     8
     1     2     8
     1     2     8
     1     2     8

B =

     1     1     1
     2     2     2
     4     4     4
     8     8     8


A./B =    1.0000    2.0000    8.0000
          0.5000    1.0000    4.0000
          0.2500    0.5000    2.0000
          0.1250    0.2500    1.0000

A.*B =      1     2     8
            2     4    16
            4     8    32
            8    16    64
```

Practice:

Q1: Let

$$A = \begin{bmatrix} 2 & 3 & 1 \\ 0 & -1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & -1 \\ 4 & -1 & 2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad E = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Compute by hand:

a. A+B
b. C*A
c. (C*A)’
d. A’C’
e. A’C
f. C*D
g. D*C
h. 3*B
i. B’
j. E’*C’
k. E’*E
l. E*E’

Check your work using Matlab (see script *ma_mod1* for solutions)

Q2: Using vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, ones-vector $\mathbf{i}$ (n x 1), and/or identity matrix **I** (n x n), derive the following expressions:

$$\frac{\sum_{i=1}^{n} x_i}{n}, \qquad \begin{bmatrix} 1-x_1^2 & -x_1x_2 & \cdots & -x_1x_n \\ -x_2x_1 & 1-x_2^2 & \cdots & -x_2x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_nx_1 & x_nx_2 & \cdots & 1-x_n^2 \end{bmatrix}$$

Q.3: Consider the 2x1vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Find **A** s.t. $\mathbf{Ax} = \begin{bmatrix} x_1 & x_1 & x_2 & x_2 & x_2 \end{bmatrix}'$ (note the transpose at the end).

Find **B** s.t. $\mathbf{Bx} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x} \end{bmatrix}$.

Combine manipulations of **x** and vector $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ to obtain $\begin{bmatrix} a_1x_1^2 + a_2x_1x_2 \\ a_2x_2^2 + a_1x_1x_2 \end{bmatrix}$.

Q.4: Using the "dot" operator in combination with *nx1* unit vector **i** and *nxn* identity matrix **I**, compute (by hand) the following derivations of vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$:

$$\left(\sum_{i=1}^{n} x_i^3 \right) / n$$

$$\underbrace{\begin{bmatrix} x_1^2 & 0 & \cdots & 0 \\ 0 & x_2^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_n^2 \end{bmatrix}}_{nxn}$$

$$\underbrace{\begin{bmatrix} \sum x_i^4 - \bar{x} \\ \sum x_i^4 - \bar{x} \\ \vdots \\ \sum x_i^4 - \bar{x} \end{bmatrix}}_{nx1}$$ where $\bar{x}$ is the mean of the elements of **x.**

Module II. Working with Partitioned Matrices

(see Matlab script *ma_mod2*)

More notes on notation (or: "a confusing convention"):

When performing matrix manipulations, including partitioning, we often want to refer to individual rows or columns. To follow Greene and some other popular econometrics textbooks, we will generically index columns by "$\mathbf{c_j}$" (*j = 1..k*, as before), and rows by their transposes, e.g. $\mathbf{x_i'}$ (i =1...n, as before). So we think of each row as a transposed column with dimension *k x 1*. (However, in some cases, we will deviate from this convention, just to keep you on your toes...)

Any matrix **X** can be visualized as being composed by sub-matrices or vectors. Fragmenting a matrix into sub-elements is called "partitioning" a matrix. Consider the following matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

There are many different ways to partition **X**. Here are some examples:

<u>Ex.1: Partition into columns</u>

$$\mathbf{X} = \begin{bmatrix} \mathbf{c_1} & \mathbf{c_2} & \mathbf{c_3} & \mathbf{c_4} \end{bmatrix} \quad \text{where}$$

$$\mathbf{c_1} = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} \quad \mathbf{c_2} = \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} \quad \mathbf{c_3} = \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} \quad \mathbf{c_4} = \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix}$$

<u>Ex.2: Partition into rows</u>

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1'} \\ \mathbf{x_2'} \\ \mathbf{x_3'} \\ \mathbf{x_4'} \end{bmatrix} \quad \text{where} \quad \mathbf{x_1} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{bmatrix} \quad \mathbf{x_2} = \begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \quad \mathbf{x_3} = \begin{bmatrix} x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \end{bmatrix} \quad \mathbf{x_4} = \begin{bmatrix} x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{bmatrix}$$

<u>Ex. 3:</u>

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_{11}} & \mathbf{X_{12}} \\ \mathbf{X_{21}} & \mathbf{X_{22}} \end{bmatrix} \quad \text{where}$$

$$\mathbf{X_{11}} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \quad \mathbf{X_{12}} = \begin{bmatrix} x_{13} & x_{14} \\ x_{23} & x_{24} \end{bmatrix}$$

$$\mathbf{X_{21}} = \begin{bmatrix} x_{31} & x_{32} \\ x_{41} & x_{42} \end{bmatrix} \quad \mathbf{X_{22}} = \begin{bmatrix} x_{33} & x_{34} \\ x_{43} & x_{44} \end{bmatrix}$$

Ex.4:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_{11}} & \mathbf{X_{12}} \\ \mathbf{X_{21}} & \mathbf{X_{22}} \end{bmatrix} \qquad \text{where}$$

$$\mathbf{X_{11}} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad \mathbf{X_{12}} = \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \end{bmatrix}$$

$$\mathbf{X_{21}} = \begin{bmatrix} x_{41} & x_{42} & x_{43} \end{bmatrix} \quad \mathbf{X_{22}} = x_{44}$$

There are two basic "rules" for partitioning a matrix:

1. Adjacent sub-matrices need to share the same dimension along the "glue" line
2. Together, the sub-matrices need to fully contain all original elements, and thus yield the original matrix.

Perhaps the best & most important aspect of partitioned matrices is that they allow us to compactly express complicated matrix products. Partitioning via columns (as in the first example above) or rows (as in example 2) is especially important in this respect.
For example, consider matrix **X** from above. Assume we want to compute or at least visualize the general structure of the matrix product $\mathbf{M} = \mathbf{X}' \times \mathbf{X}$. An element-by-element approach will be tedious. Instead, we partition **X** into column vectors as above and proceed from there. From before:

$$\mathbf{X} = \begin{bmatrix} \mathbf{c_1} & \mathbf{c_2} & \mathbf{c_3} & \mathbf{c_4} \end{bmatrix}$$

When performing matrix operations using partitioned matrices, it is best to look at each element of the partitioned matrix as a "1 by 1" (i.e. "pseudo-scalar"), but to keep the actual dimensions of each sub-matrix in mind.

In this case, we can view **X** as a 1 by 4 "row vector". Naturally, each of these pseudo-scalars has dimension (4 by 1) as is clear from inspection of the original **X**.

Transposing a partitioned matrix
1. Take the transpose of the original partitioned matrix (treating each sub-matrix as a pseudo-scalar)
2. Transpose each sub-matrix.

here:

$$\mathbf{X}' = \begin{bmatrix} \mathbf{c}_1' \\ \mathbf{c}_2' \\ \mathbf{c}_3' \\ \mathbf{c}_4' \end{bmatrix}$$

Now we can use basic matrix multiplication to derive the product. First, let's double-check that the two constructs are conformable in their pseudo-dimensions. **X'** is pseudo- 4 by 1. **X** is pseudo 1 by 4. The result will be pseudo- 4 by 4. Naturally, conformability also has to hold in actual dimensions: **X'** is 4 by 4, and so is **X**. The actual dimensions of the product will thus also be a 4 by 4. Thus, we get:

$$\mathbf{X}'\times\mathbf{X}=\begin{bmatrix}\mathbf{c}_1'\mathbf{c}_1 & \mathbf{c}_1'\mathbf{c}_2 & \mathbf{c}_1'\mathbf{c}_3 & \mathbf{c}_1'\mathbf{c}_4\\ \mathbf{c}_2'\mathbf{c}_1 & \mathbf{c}_2'\mathbf{c}_2 & \mathbf{c}_2'\mathbf{c}_3 & \mathbf{c}_2'\mathbf{c}_4\\ \mathbf{c}_3'\mathbf{c}_1 & \mathbf{c}_3'\mathbf{c}_2 & \mathbf{c}_3'\mathbf{c}_3 & \mathbf{c}_3'\mathbf{c}_4\\ \mathbf{c}_4'\mathbf{c}_1 & \mathbf{c}_4'\mathbf{c}_2 & \mathbf{c}_4'\mathbf{c}_3 & \mathbf{c}_4'\mathbf{c}_4\end{bmatrix}=\begin{bmatrix}\sum_{i=1}^{4}x_{i1}x_{i1} & \sum_{i=1}^{4}x_{i1}x_{i2} & \sum_{i=1}^{4}x_{i1}x_{i3} & \sum_{i=1}^{4}x_{i1}x_{i4}\\ \sum_{i=1}^{4}x_{i2}x_{i1} & \sum_{i=1}^{4}x_{i2}x_{i2} & \sum_{i=1}^{4}x_{i2}x_{i3} & \sum_{i=1}^{4}x_{i2}x_{i4}\\ \sum_{i=1}^{4}x_{i3}x_{i1} & \sum_{i=1}^{4}x_{i3}x_{i2} & \sum_{i=1}^{4}x_{i3}x_{i3} & \sum_{i=1}^{4}x_{i3}x_{i4}\\ \sum_{i=1}^{4}x_{i4}x_{i1} & \sum_{i=1}^{4}x_{i4}x_{i2} & \sum_{i=1}^{4}x_{i4}x_{i3} & \sum_{i=1}^{4}x_{i4}x_{i4}\end{bmatrix}$$

The result is a symmetric matrix. This is always true for any "**X'*X**" operation, regardless of the original dimensions of **X**.

<u>Practice:</u>

Q1: Consider *n* by *k* matrix **X** and *n* by 1 vector **y**. Denote the rows of **X** as $\mathbf{x_i}'$, *i=1..n*, and the columns of **X** as $\mathbf{c_j}$, *j=1…k*. Also, denote the elements of **y** as y_i *i=1…n*. Express the following matrix products first in terms of $\mathbf{x_i}$, then in terms of $\mathbf{c_j}$ (plus any y_i's if needed).

(i) **X'y** (ii) **X'X** (iii) **XX'**

Q2: Consider the following vectors:

$$\mathbf{a}=\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}\quad \mathbf{b}=\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}\quad \mathbf{c}=\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix}\quad \mathbf{x}=\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}\quad \mathbf{y}=\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}\quad \mathbf{z}=\begin{bmatrix}z_1\\z_2\\z_3\end{bmatrix}$$

Compute $\mathbf{a}'\mathbf{b}$ and $\mathbf{ab}'$ (show the full results with all individual elements). Then consider the 3 x 3 matrices $\mathbf{A}=[\mathbf{a}\quad\mathbf{b}\quad\mathbf{c}]$ and $\mathbf{X}=[\mathbf{x}\quad\mathbf{y}\quad\mathbf{z}]$. Show the full results with all individual elements for $\mathbf{A}'\mathbf{X}$. Then express $\mathbf{A}'\mathbf{X}$ in terms of vectors **a, b, c, x, y**, and **z**.

Module III. Linear Dependence and Column Rank

(see Matlab script *ma_mod3*)

General definitions & rules

1. When we talk about the "rank" of a matrix, we usually refer to its *column rank*.
2. The *column rank* of a matrix is the number of its *linearly independent* columns.
3. The column rank of a matrix can never exceed the number of its rows (important if $n<=k$)

Examples:

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 11 & 4 \\ 3 & 3 & 6 \\ 4 & 3 & 8 \end{bmatrix} \qquad \text{rank}(\mathbf{A}_1) = 2 \text{ (why?)}$$

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 11 & 4 \\ 3 & 3 & 5 \\ 4 & 3 & 8 \end{bmatrix} \qquad \text{rank}(\mathbf{A}_2) = 3$$

$$\mathbf{A}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 11 & 4 \end{bmatrix} \qquad \text{rank}(\mathbf{A}_3) = 2 \text{ (why?)}$$

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 1 & 11 & -3.5 \\ 1 & 2 & 2 & 3 \\ 1 & 3 & 4 & 4 \\ 1 & 4 & 0 & 8 \\ 1 & 5 & 6 & 7 \end{bmatrix}$$ rank($\mathbf{A}_4$) = 3 (linear dependence can be subtle...see script *ma_mod3*)

More general definitions & rules:

1. A matrix **X** has "full rank" when *rank(***X***) = k*. A full-rank matrix is often called "non-singular".
2. A rank-deficient matrix is called "singular".
3. A full-rank matrix has a non-zero determinant (though determinants are only defined for square matrices). The determinant of a singular matrix is 0.
4. Full rank is needed for the *inverse* of a matrix to exist (though inverses are only defined for square matrices).
5. rank(**A*B**) = min(rank(**A**), rank(**B**))
6. rank(**A**) = rank(**A'A**) = rank(**AA'**)

Module IV. Matrix Inverses and Systems of Linear Equations

(see Matlab script *ma_mod4*)

Inverses are only defined for square, non-singular matrices.
Consider square (*n* x *n*) matrix **A**. The inverse of **A** is defined as follows:

$\mathbf{B} = \mathbf{A}^{-1} \leftrightarrow \mathbf{B} \times \mathbf{A} = \mathbf{I}$ (identity matrix)

Thus:
$\mathbf{A} \times \mathbf{A}^{-1} = \mathbf{A}^{-1} \times \mathbf{A} = \mathbf{I}$

<u>*Important rules:*</u>

$(\mathbf{ABC})^{-1} = \mathbf{C}^{-1}\mathbf{B}^{-1}\mathbf{A}^{-1}$

$(\mathbf{A}')^{-1} = (\mathbf{A}^{-1})'$

There are three cases in econometrics where the inverse plays an important role:

1. In density functions of multivariate distributions
2. For simplification of complex matrix multiplications
3. Solving systems of linear equations

<u>To illustrate (2):</u>
In a product of matrices, any matrix or cluster of matrices that is immediately pre-or post-multiplied by its inverse will "cancel out". Example: $\mathbf{B} \times (\mathbf{ACD})^{-1}\mathbf{ACD} \times \mathbf{F} = \mathbf{BIF} = \mathbf{BF}$

<u>To illustrate (3):</u>
First, let's see how a system of linear equations can be expressed in matrix terms, using the linear regression model as an example.

In linear regression analysis, we stipulate the following general relationship for a given individual (or firm, household etc) between dependent variable ("outcome") y_i and a vector of explanatory variables $\mathbf{x_i} = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{ik}]'$ (x_{i1} is set to "1" for all *i* if the regression model contains an intercept term):

$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik}$ (omitting any error terms for now)

We can stack these equations for all *i=1..n* individuals:

$$\begin{aligned} y_1 &= \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_k x_{1k} \\ y_2 &= \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_k x_{2k} \\ &\vdots \\ y_n &= \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_k x_{nk} \end{aligned}$$

Next, we note that the left hand side can be compactly expressed as a vector $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]'$. For a given individual, the right hand side can be written as an inner product of vectors:

$\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} = \mathbf{x_i'} \times \boldsymbol{\beta}$ where

$$\mathbf{x_i'} = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{ik}],\ \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

We can now write the entire system as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{x_1'\beta} \\ \mathbf{x_2'\beta} \\ \vdots \\ \mathbf{x_n'\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{x_1'} \\ \mathbf{x_2'} \\ \vdots \\ \mathbf{x_n'} \end{bmatrix} \cdot \boldsymbol{\beta} = \mathbf{X} \cdot \boldsymbol{\beta} \qquad \text{where} \quad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

We now need the inverse to solve systems of linear equations (assuming **X** is full rank). Changing notation from $\boldsymbol{\beta}$ to **b** (the conventional symbol for the regression estimate of $\boldsymbol{\beta}$), we have:

$$\mathbf{y} = \mathbf{Xb} \qquad \text{where} \quad \mathbf{y} = (nx1), \quad \mathbf{X} = (nxk), \quad \mathbf{b} = (kx1)$$

$$(\mathbf{X'X})^{-1}\mathbf{X'y} = (\mathbf{X'X})^{-1}\mathbf{X'Xb} = \mathbf{b}$$

Recall that **X'X** is always square, so the full rank condition of **X** is the only assumption needed for this inverse to exist.

Module V. Quadratic Forms, Characteristic Roots, and Matrix Decomposition

(see Matlab script *ma_mod5*)

Consider an (*nxn*) symmetric matrix **A** and an (*nx1*) vector **x**. Then the scalar $q = \mathbf{x}'\mathbf{A}\mathbf{x}$ is called a "quadratic form". This structure occurs in many optimization problems. It is also used to assess the "definiteness" of a matrix. Specifically:

1. If q>0 for all nonzero **x** (i.e for all **x** that do not contain zeros exclusively), **A** is "positive definite".
2. If q<0 for all nonzero **x**, **A** is "negative definite".
3. If $q \geq 0$ for all nonzero **x**, **A** is "nonnegative definite" or "positive semidefinite".

The definiteness of a matrix becomes an issue in three main situations in Econometrics:

1. When comparing the **relative "magnitude"** of two matrices:

   Assume we have two symmetric matrices of equal dimensions (e.g. the variance-covariance matrices of two competing estimators), **A** and **B**. We would like to determine which matrix is "larger" (e.g. we would like to choose the estimator with "less noise", i.e. with the smaller variance-covariance matrix). We could then compute the quadratic form of the difference and examine its magnitude, i.e. $q = \mathbf{x}'(\mathbf{A} - \mathbf{B})\mathbf{x}$. If q>0, the difference is positive definite, and we would conclude that **A** is "larger" than **B**. The opposite holds if q<0.

2. In **Cholesky factorization** (CF) of a matrix.

   In many econometric analyses we would like to represent a symmetric, positive definite matrix as the product of a lower and an upper triangular matrix. Positive-definiteness is a requirement for this to work. The CF, in turn, is needed to draw from certain multivariate distributions and to ascertain positive-definiteness of variance-covariance matrices in maximum likelihood estimation (MLE).

   Example:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \quad \mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \quad \mathbf{L}' = \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} \text{ where}$$

   the relationship between the elements of A and L has to be such that **A=LL'** holds. This implies:

$$\begin{array}{lll} a_{11} = l_{11}^2 & a_{12} = l_{11}l_{21} & a_{13} = l_{11}l_{31} \\ a_{22} = l_{21}^2 + l_{22}^2 & a_{23} = l_{21}l_{31} + l_{22}l_{32} & a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \end{array}$$

   The Matlab command for the CF is `L = chol(A)`. If **A** is not positive definite, and error message occurs. However, note that in Matlab `chol(A)` produces an upper triangular. So to reconstruct **A**, use **L'*L** (and NOT **L*L'**).

3. In optimization problems, to assure that a maximum is found, the matrix of second derivatives ("**Hessian matrix**") must be negative-definite. It must be positive-definite for a minimum.

Assessing Definiteness:
The only reliable way to assess the definiteness property of a matrix is by inspection of its *eigenvalues* or *characteristic roots*. Consider (*n* x *n*) symmetric matrix **A**. Characteristic roots and their corresponding characteristic vectors are defined via the following equality:

$$\mathbf{AC} = \mathbf{C\Lambda}$$

where all matrices are *n* by *n*. The columns of **C** are called "eigenvectors" or "characteristic vectors" of **A**. Matrix $\mathbf{\Lambda}$ is a diagonal matrix with the *n* eigenvalues of **A** on its diagonal. If all eigenvalues > 0, **A** is positive definite. If all of them are <0, **A** is negative definite. If some are >0, some = 0, **A** is positive semi-definite.

In Matlab, the vector of eigenvalues can be obtained quickly using `eig(A)`. If both eigenvalues and eigenvectors are required, use `[C L]=eig(A)`. `C` Corresponds to matrix **C** above, and `L` corresponds to matrix $\mathbf{\Lambda}$. You can verify that C and L are correct using the rule:

$$\mathbf{A} = \mathbf{C\Lambda C'}$$

This is called the "**spectral decomposition**" of **A**.

Important rules:
1. If **A** is pos. def., so is its inverse.
2. If **A** is *n* by *k* with full column rank, and *n*>*k*, then **A'A** is pos. def. and **AA'** is nonnegative def. (important for linear regression models)
3. If two matrices **A** and **B** are both pos. def, and every eigenvalue of **A** > eigenvalue of **B** when sorted from smallest to largest, (**A** - **B**) is pos. def (so this is ultimately how we can figure out which matrix is "larger")
4. The rank of a matrix is the number of its non-zero eigenvalues.

Practice:

Consider $\mathbf{A} = \begin{bmatrix} 20 & 4 & 3 & 1 \\ 4 & 8 & 6 & 2 \\ 3 & 6 & 5 & 7 \\ 11 & 2 & 0 & 4 \end{bmatrix}$.

Create a separate Matlab script and call it “`m5_practice`”.

Your program should accomplish the following:
- create symmetric **A'*A**, call it **A1**
- get the set of eigenvalues and the matrix of eigenvectors for **A1**.
- verify that A-80, A-84, and A-85 hold (Greene p. 826-827)

a. Deduce the rank of **A1** by inspecting the set of eigenvalues.
b. Would you expect the determinant of **A1** to be zero? Why or why not? Verify this using `det( )`
c. Are the columns in **A1** linearly independent – why or why not?
d. Verify **A1**'s rank using `rank`().

Matrix "Calculus"

Scalar-valued functions:
Consider a general function that relates a vector **x** to a scalar y, i.e.

$y = f(\mathbf{x})$ where $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_k]'$

By convention, the set of derivatives $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is understood to be a *k* x 1 column vector of the form:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_k} \end{bmatrix} = g(\mathbf{x}) \quad \text{where "g" stands for "}\textbf{gradient}\text{"}$$

(If for some reason you wish to obtain a row vector instead, use the notation $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}'}$)

Example:

$f(\mathbf{x}) = 3 + 2x_1 + 4x_2 + 6x_1x_2$

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 2+6x_2 \\ 4+6x_1 \end{bmatrix} \qquad \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}'} = [2+6x_2 \quad 4+6x_1]$$

Taking second derivatives yields a *symmetric k* x *k* matrix:

$$\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}'} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_k} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_k \partial x_1} & \frac{\partial^2 f}{\partial x_k \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_k^2} \end{bmatrix} = H(\mathbf{x}) \quad \text{where "H" stands for "Hessian".}$$

For our example:

$$\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}'} = \begin{bmatrix} 0 & 6 \\ 6 & 0 \end{bmatrix}$$

Some convenient results for scalar valued linear functions of **x** (see Greene Appendix A, p. 839):

$$\frac{\partial \mathbf{a}'\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a} \qquad \frac{\partial \mathbf{a}'\mathbf{x}}{\partial \mathbf{x}'} = \mathbf{a}'$$

$$\frac{\partial \mathbf{x}'\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}')\mathbf{x} \qquad (= 2\mathbf{A}\mathbf{x} \quad \text{if } \mathbf{A} \text{ is symmetric})$$

Systems of equations (or "vector-valued" functions):

Now consider a set of scalar-valued function

$$\begin{aligned} y_1 &= f_1(\mathbf{x}) \\ y_2 &= f_2(\mathbf{x}) \\ &\vdots \\ y_n &= f_n(\mathbf{x}) \end{aligned}$$

Here **x** is the full set of *k* parameters contained in the system. Not all equations may contain all *k* elements, but that's OK. We can compactly write this as

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

In this case, convention dictates that first derivatives are taken with respect to **x'.** The resulting matrix of first derivatives will have dimension *n* by *k* and take the following form:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}'} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_k} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_k} \end{bmatrix}$$

If a certain element of **x** is not represented in a given equation, the corresponding derivative will be automatically set to "0", s.t. the resulting matrix is always *n* by *k*.

Derivatives of equation systems play an important role in regression models for the estimation of variance-covariances for functions of original estimators (using the so called "Delta method").

If the system is linear in **x** without any cross-terms, a convenient result is available:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}'} = \frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}'} = \mathbf{A} \quad \textit{and} \qquad \frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{A}'$$ where **A** is (*n* x *k*)