Linear Algebra Using Matlab

MATLAB is short for "MATrix LABoratory". It provides a friendly environment for manipulating matrices and also contains a large number of built routines that make matrix algebra simple.

Basic Matrix Commands

If M is a matrix in MATLAB, then M(2,3) denotes the entry in the second row, third column of M. For example:

```
>> M =[ 2 4 5 -1; 3 2 -2 1; 0 3 -2 1]
M =
     2     4     5    -1
     3     2    -2     1
     0     3    -2     1

>> M(2,3)
ans =
    -2
```

Try `M(3,2)`.

Matrix manipulation is made easy by the 'all' operator ':'. For instance, to access all elements in the second row of `M`,

```
>> M(2,:)
ans =
     3     2    -2     1
```

Similarly, all elements of the first column of `M`

```
>> M(:,1)
ans =
     2
     3
     0
```

We can also easily refer to sub-matrices. For example if we wanted the matrix made up of both the first and third rows of `M`:

```
>> M([1 3],:)
ans =
     2     4     5    -1
     0     3    -2     1
```

To produce the matrix formed by eliminating the second row and third column of `M`:

```
>> M([1 3],[1 2 4])
ans =
     2     4    -1
     0     3     1
```

Notice this allows us to easily interchange individual rows or columns. For example, if we wanted to switch rows 1 and 3 of `M`:

```
>> M([1 3],:) = M([3 1],:)
M =
     0     3    -2     1
     3     2    -2     1
     2     4     5    -1
```

Finally, Matlab makes taking the transpose of a matrix easy. Try typing `M'`.

Elementary Row Operations

Suppose we have a set of equations given by

$$Mx = 0$$

where `M` is the matrix defined above. Our solution technique involves reducing the matrix using elementary row operations. These are easily accomplished in Matlab without worrying about keeping the arithmetic straight.

Start by reloading `M`. (Note, placing a semicolon at the end of a Matlab line suppresses the output that we don't care to see.)

```
>> M =[ 2 4 5 -1; 3 2 -2 1; 0 3 -2 1];
```

Now, start the reduction by dividing the first row by the first element. Since we are working with a matrix of rational numbers, we will tell Matlab to output all results in terms of rationals.

```
>> format rat
>> M(1,:) = M(1,:)/M(1,1)
M =
      1            2            5/2          -1/2
      3            2           -2             1
      0            3           -2             1
```

The next step is to zero the other entries in column 1.

```
>> M(2,:) = M(2,:) - M(2,1)*M(1,:)
M =
      1            2            5/2          -1/2
      0           -4          -19/2           5/2
      0            3           -2             1
```

Next, we set the leading entry in row 2 to one,

```
>> M(2,:) = M(2,:)/M(2,2)
M =
      1            2            5/2          -1/2
      0            1           19/8          -5/8
      0            3           -2             1
```

Now zero the rest of the elements in column 2.

```
>> M(1,:) = M(1,:) - M(1,2)*M(2,:)
M =
      1            0           -9/4          3/4
      0            1           19/8         -5/8
      0            3            -2            1

>> M(3,:) = M(3,:) - M(3,2)*M(2,:)
M =
      1            0           -9/4          3/4
      0            1           19/8         -5/8
      0            0          -73/8         23/8
```

Finally, set the leading element in row 3 to unity and then zero all other elements in column 3:

```
>> M(3,:) = M(3,:)/M(3,3)
M =
      1            0           -9/4          3/4
      0            1           19/8         -5/8
      0            0             1        -23/73

>> M(1,:) = M(1,:) - M(1,3)*M(3,:)
M =
      1            0             0          3/73
      0            1           19/8         -5/8
      0            0             1        -23/73

>> M(2,:) = M(2,:) - M(2,3)*M(3,:)
M =
      1            0             0          3/73
      0            1             0          9/73
      0            0             1        -23/73
```

The solution can now be read off directly:

$$\begin{aligned} x_1 + 3/73x_4 &= 0 \\ x_2 + 9/73x_4 &= 0 \\ x_3 - 23/73x_4 &= 0 \end{aligned}$$

Here x_4 is a free (independent) variable so the solution has one free parameter. With x_4 set equal to α, the solution space is given by

$$\mathbf{x} = \alpha \begin{pmatrix} -3 \\ -9 \\ 23 \\ 1 \end{pmatrix}$$

Rank and Dimension of Solution Space

Recall, the size (dimension) of S, the solution space (the number of free parameters (independent variables) in the solution) is given by

$$dim(S) = m - rank(M)$$

where M is an $(n \times m)$ matrix and $rank(M)$ is the number of non-zero rows in reduced M.

Matlab has a convenient command for finding the rank of a matrix *without* having to reduce it; for any matrix `A`, just type `rank(A)` . Try it out by determining the dimension of the solution space in the following:

$$\begin{pmatrix} 4 & -4 & -8 & -2 \\ 0 & 2 & 2 & -5 \\ 1 & -2 & -3 & 2 \end{pmatrix} \mathbf{X} = \mathbf{0}$$

NonHomogeneous Equations, Inverses and the `rref` Command

Dealing with nonhomogeneous equations is straightforward, we simply append the right hand side to form an augmented matrix. Consider the equation set:

$$\begin{aligned} 2x_2 + 2x_3 + 3x_4 &= -4 \\ -2x_1 + 4x_2 + 2x_3 - x_4 &= -6 \\ 3x_1 - 4x_2 - x_3 + 2x_4 &= 86 \end{aligned}$$

Translating to MatLab,

```
>> A = [0 2 2 3; -2 4 2 -1; 3 -4 -1 2]
A =
      0            2            2            3
     -2            4            2           -1
      3           -4           -1            2

>> b = [-4;-6;8]
b =
     -4
     -6
      8
```

Now lets form the augmented matrix, `M`:

```
>> M = [A b]
M =

      0            2            2            3           -4
     -2            4            2           -1           -6
      3           -4           -1            2            8
```

We can perform a series of elementary row operations on `M` (try this!) to produce the reduced matrix:

```
ans =
       1          0          1          0        16/5
       0          1          1          0        -1/5
       0          0          0          1        -6/5
```

In this case the solution is given by:

$$\begin{aligned} x_1 + x_3 &= 16/5 \\ x_2 + x_3 &= -1/5 \\ x_4 &= -6/5 \end{aligned}$$

Or, in vector form:

$$\mathbf{X} = \begin{pmatrix} 16/5 \\ -1/5 \\ 0 \\ -6/5 \end{pmatrix} + \alpha \begin{pmatrix} -1 \\ -1 \\ 1 \\ 0 \end{pmatrix}$$

Obviously Matlab simplifies the process of matrix reduction, but surely there must be an easier way of doing this than actually doing each elementary row operation. There is, simply use the built-in command **rref** (reduced row echelon form).

For example:

```
>> M = [1 2 3 4; 5 6 7 8; 10 12 14 16]
M =

     1     2     3     4
     5     6     7     8
    10    12    14    16

>> rref(M)
ans =

     1     0    -1    -2
     0     1     2     3
     0     0     0     0
```

For square matrices, an alternate procedure for finding the solution to nonhomogeneous problems is to produce an *inverse* for the matrix `A`. Since the inverse, A^{-1} has the property: $\mathbf{A^{-1}A} = \mathbf{I}$, the solution to

$$\mathbf{AX} = \mathbf{b}$$

is simply given by

$$\mathbf{A^{-1}AX} = \mathbf{X} = \mathbf{A^{-1}b}$$

(of course this assumes the inverse of `A` exists!). Matlab has a routine for calculating an inverse matrices directly.

```
>> A = [ 1 3 4 1; 1 -2 3 4; -2 -2 1 3; -1 2 1 3]
A =
      1            3            4            1
      1           -2            3            4
     -2           -2            1            3
     -1            2            1            3

>> A1 = inv(A)
A1 =
   -20/139       44/139      -63/139       11/139
     5/139      -11/139      -19/139       32/139
    42/139       -9/139       35/139      -37/139
   -24/139       25/139      -20/139       41/139

>> b = [2 3 4 5]'   /note the use of the transpose/
b =
      2
      3
      4
      5

>> X = A1*b
X =
  -105/139
    61/139
    12/139
   152/139
```

To check the answer, try multiplying `X` by the original matrix `A`:

```
>>A*X
ans =
      2
      3
      4
      5
```

Determinants and Linear Independence

Matlab also provides for simple calculation of the determinant of a matrix. This gives us a way of determining if a collection of vectors are linearly independent. For example, consider the following three vectors in $\mathcal{R}^3$:

$$v_1 = \begin{pmatrix} 4 \\ 1 \\ -1 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix} \quad v_3 = \begin{pmatrix} -2 \\ 1 \\ 5 \end{pmatrix}$$

One can easily check for linear dependence of the vectors by looking at the determinant of the matrix formed by taking each vector as a column (or a row).

```
>> M = [[4 1 -1]' [0 1 3]' [-2 1 5]']  %note: transpose changes row vectors to columns
M =
     4     0    -2
     1     1     1
    -1     3     5

>> det(M)
ans =
     0
```

The zero determinant tells us the vectors are linearly dependent, one of them can be written as a linear combination of the other two. How can we tell how to write v_3 in terms of v_1 and v_2? Or, better yet, what is the minimum set of vectors whose linear combination can describe all three v_1, v_2, v_3?

Since the vectors are linearly dependent, we know that there exists some set of c_1, c_2, c_3 not all zero such that:

$$c_1 v_1 + c_2 v_2 + c_3 v_3 = 0$$

But this is the same as solving the homogeneous equation:

$$Mc = 0$$

where c is a column vector of coefficients. In other words, to see how v_1, v_2 and v_3 are related, find the solution to the homogeneous equation.

```
>> rref(M)
ans =
    1.0000         0   -0.5000
         0    1.0000    1.5000
         0         0         0
```

This gives the relation between the coefficients of the three vectors, namely (after multiplying through by 2):

$$2c_1 = c_3 \ , \quad 2c_2 = -3c_3$$

or, since c_3 is arbitrary and not zero:

$$1v_1 - 3v_2 + 2v_3 = 0$$