```
% Course 18.06: Linear Algebra *** MATLAB Demonstration
%
% This is a transcript of a simple MATLAB session.
% It was originally saved as the file called rundemo -- and now
% I have edited the original file with these comments.
%

>> diary rundemo

% Here is some information on the command called diary.

>> help diary

 DIARY  Save the text of a MATLAB session.
        DIARY file_name  causes a copy of all subsequent terminal input
        and most of the resulting output to be written on the named
        file. DIARY OFF suspends it. DIARY ON turns it back on.
        DIARY, by itself, toggles the diary state.

% Each row of a vector (or matrix) ends with a semicolon.
% MATLAB is case sensitive, so b and B are different variables.

>> b = [4; 8; 14]

b =

     4
     8
    14

>> A = [1 1 1; 1 2 4; 1 3 9]

A =

     1     1     1
     1     2     4
     1     3     9

% slu computes the LU factorization of a square matrix.

>> help slu

 SLU    Simple, square, LU factorization.
        [L,U] = SLU(A) for a square matrix A, illustrates the use of
```

```
        Gaussian elimination to compute a lower triangular matrix L
        and an upper triangular matrix U so that L*U = A.

        The algorithm does no pivoting and so will fail if a small
        pivot is encountered.

        See also SLV, PLU, LU.

>> [L, U] = slu(A)

L =

     1     0     0
     1     1     0
     1     2     1


U =

     1     1     1
     0     1     3
     0     0     2

% Let's verify that the product L * U is the same as A.

>> L * U

ans =

     1     1     1
     1     2     4
     1     3     9

>> A

A =

     1     1     1
     1     2     4
     1     3     9

% whos shows your variables and other information.

>> whos
               Name        Size       Elements     Bytes    Density   Complex
```

```
  A       3 by 3             9        72       Full      No
  L       3 by 3             9        72       Full      No
  U       3 by 3             9        72       Full      No
ans       3 by 3             9        72       Full      No
  b       3 by 1             3        24       Full      No

Grand total is 39 elements using 312 bytes

% Let's use slv to solve A x = b.

>> help slv

 SLV    Simple linear equation solver.
        x = SLV(A,b) tries to use the LU factorization computed by SLU(A)
        to solve the linear equation A*x = b.
        Since SLU does no pivoting, SLV may fail if a small pivot is
        encountered.

        See also SLU, SOLVE, \ .

>> x = slv(A,b)

x =

     2
     1
     1

% Let's verify that A * x is the same as b.

>> A * x

ans =

     4
     8
    14

>> b

b =

     4
     8
```

```
    14

% We can also find the solution x by computing Ainv * b.

>> help inv

 INV    Matrix inverse.
        INV(X) is the inverse of the square matrix X.
        A warning message is printed if X is badly scaled or
        nearly singular.

>> Ainv = inv(A)

Ainv =

    3.0000   -3.0000    1.0000
   -2.5000    4.0000   -1.5000
    0.5000   -1.0000    0.5000

% The solution is the same as x.

>> soln = Ainv * b

soln =

     2
     1
     1

% eye is the command for creating the identity matrix.

>> I = eye(3,3)

I =

     1     0     0
     0     1     0
     0     0     1

% Notice how we create the augmented matrix Aaug.

>> Aaug = [ A I ]

Aaug =
```

```
     1     1     1     1     0     0
     1     2     4     0     1     0
     1     3     9     0     0     1

% This is an example of Gauss-Jordan elimination for computing Ainv.

>> help ref

 REF    Reduced Row Echelon Form.
        R = ref(A) uses the pivoting LU factorization computed by PLU
        to find the reduced row echelon form of a rectangular matrix A.

>> R = ref(Aaug)

R =

    1.0000         0         0    3.0000   -3.0000    1.0000
         0    1.0000         0   -2.5000    4.0000   -1.5000
         0         0    1.0000    0.5000   -1.0000    0.5000

% The row operations give the augmented matrix R = [ I   Ainv ].
% Let's display Ainv, again.

>> Ainv

Ainv =

    3.0000   -3.0000    1.0000
   -2.5000    4.0000   -1.5000
    0.5000   -1.0000    0.5000

% size gives the number of rows and number of columns.

>> size(A)

ans =

     3     3

>> size(Aaug)

ans =

     3     6
```

```
% A matrix can be tranposed using the command   '

>> R

R =

    1.0000         0         0    3.0000   -3.0000    1.0000
         0    1.0000         0   -2.5000    4.0000   -1.5000
         0         0    1.0000    0.5000   -1.0000    0.5000

>> R'

ans =

    1.0000         0         0
         0    1.0000         0
         0         0    1.0000
    3.0000   -2.5000    0.5000
   -3.0000    4.0000   -1.0000
    1.0000   -1.5000    0.5000

>> diary off
```