Working with matrices in MATLAB

MATLAB has many useful commands for creating, manipulating or operating on matrices. I hope to show you just a small sample here. To see a more full list, type `help elmat` or `help matfun`.

You can create a 3×4 matrix full of zeros:

```
>> A = zeros(3,4)
A =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

or a matrix full of ones:

```
>> B = ones(3,4)
B =
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Note that you can make a row vector full of zeros by typing `zeros(1,5)` or a column vector of zeros using `zeros(5,1)`. (Typing `zeros(5)` will give a 5×5 matrix full of zeros.) Perhaps more useful than a matrix full of ones is the identity matrix, which has ones on the main diagonal and zeros elsewhere. Remember that identity matrices are always square, so that you only need to specify the number of rows in the matrix:

```
>>I = eye(4)
I =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Identifying rows and columns of matrices

The 3^{rd} row of a matrix can be seen in MATLAB by typing:

```
>> I(3,:)
ans =
     0     0     1     0
```

The third column can be seen in a similar fashion:

```
>> I(:,3)
ans =
     0
     0
     1
     0
```

You can even select submatrices in this way. Here we pick off a 3x3 matrix from the upper right corner of the identity matrix

```
>> I(1:3,2:4)
ans =
     0     0     0
     1     0     0
     0     1     0
```

Diagonals and triangles

Before introducing these, let's create a 4x4 random matrix:

```
>> A = rand(4)
A =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
```

You can select the upper triangular part of A using `triu`:

```
>> upper = triu(A)
upper =
    0.9501    0.8913    0.8214    0.9218
         0    0.7621    0.4447    0.7382
         0         0    0.6154    0.1763
         0         0         0    0.4057
```

The lower triangular part may be picked off using `tril`:

```
>> lower = tril(A)
lower =
    0.9501         0         0         0
    0.2311    0.7621         0         0
    0.6068    0.4565    0.6154         0
    0.4860    0.0185    0.7919    0.4057
```

The main diagonal may be extracted (as a vector) using `diag`:

```
>> d = diag(A)
d =
    0.9501
    0.7621
    0.6154
    0.4057
```

One can build matrices from diagonals. For example, useful matrices may be constructed from their diagonals:

```
>> D = diag([1 1 1 1],-1)
D =
     0     0     0     0     0
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0

>> D = D + diag([1 1 1 1],1)
D =
     0     1     0     0     0
     1     0     1     0     0
     0     1     0     1     0
     0     0     1     0     1
     0     0     0     1     0

>> D = D - 2*eye(5)
D =
    -2     1     0     0     0
     1    -2     1     0     0
     0     1    -2     1     0
     0     0     1    -2     1
     0     0     0     1    -2
```

After constructing the matrix with the main diagonals, you may alter the first and last row if they differ from the rest:

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

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

The command `diag(d,n)` takes the vector `d` and places it on the n^{th} diagonal of the matrix (relative to the main diagonal. Typing `diag(d)` will put `d` on the main diagonal. Typing `diag(d,1)` will put it one diagonal above the main diagonal — `diag(d,-1)` will put it one diagonal below.

The Spy Command

Typing `spy(D)` will bring up the figure window and show the location of all the non-zero elements in `D`. This can help you learn a lot about the structure of sparse (and other types of) matrices. Try it with triangular and other matrices just to see. It works best with large matrices. If your matrix is small, you might try `spy(D,'*')` (as I did to generate the figure below) to put asterisks at each non-zero location so that they are more visible.