Reading text data from a file

There are three steps to reading text from a file: (1) open the file, (2) read in the data, and (3) close the file. Here is an example for reading a plain text file (ASCII characters) and storing the text in a cell array.

```
function C = txtfile2cell(fname)
% fname is a string that names a plain text file in the current directory.
% C is a cell array with C{k} being the k-th line in the file,
%   i.e., C{k} is a row vector of characters.
fid= fopen(fname, 'r'); % fid is the identifier for the file (opened for reading)
k= 0;
while  ~feof(fid)
   k= k+1;
   C{k}= fgetl(fid);
end
fclose(fid);
```

The built-in functions `fopen` and `fclose` open and close a text file, as suggested by the function names. When you open a file, you need to specify the filename as well as how the file will be used: 'r' for read; 'a' for append, i.e., append to an existing file; or 'w' for write, i.e., create a new file or replace any text if the file already exists.

In a plain text file, there are hidden markers that mark the end of each line and the end of a file. The built-in function `feof` checks the current location in the file (initially at the beginning of the first line) and returns true (1) if the end of file marker is seen. Otherwise `feof` returns false (0).

The built-in function `fgetl` "gets" the current line. Every call of `fgetl` gets one line from the file, not including the end of line marker.

Getting only the (few) pieces of data that you want from a (big) file: Instead of reading and storing all the data in a file, you can select only the pieces that you want by writing some extra code. You would need to know something about the format of the data though. Below are several lines of a text file containing the data on the protein *2cro* downloaded from the protein database `http://www.rcsb.org`. The entire file has more than 800 lines, and the website explains how the data are laid out in the text file. Suppose you only need the position data for the atoms in the protein. That information is contained in the lines that begin with the word 'ATOM' and the x, y, and z values are written as *text* in columns 33–38, 41–46, and 49–54, respectively.

```
HEADER    GENE REGULATING PROTEIN                 08-DEC-88   2CRO
TITLE     STRUCTURE OF PHAGE 434 CRO PROTEIN AT 2.35 ANGSTROMS
COMPND   2 MOLECULE: REGULATORY PROTEIN CRO;
SCALE1      0.020072  0.011589  0.000000        0.00000
SCALE2      0.000000  0.023177  0.000000        0.00000
SCALE3      0.000000  0.000000  0.006485        0.00000
ATOM      1  N   MET A  -1      22.763   5.580  12.506  1.00 57.49           N
ATOM      2  CA  MET A  -1      21.346   5.232  12.330  1.00 57.64           C
ATOM      3  C   MET A  -1      21.110   4.818  10.877  1.00 57.49           C
ATOM      4  O   MET A  -1      19.943   4.746  10.455  1.00 58.01           O
```

Below is a script to extract the position data from the data file `2cro.txt`. The built-in function `strcmp` compares two strings and returns true if the strings are the same. The built-in function `str2double` converts a string that indicates a numeric value (a string containing digits and possibly the decimal point and the negative sign) to a double precision numeric value. For example, `str2double('2.01')` converts the length 4 *string* `'2.01'` into the type *double* scalar storing the value 2.010000....

```
fid= fopen('2cro.txt', 'r');
x=[];  y=[];  z=[];
while  ~feof(fid)
   s= fgetl(fid);
   if strcmp(s(1:4), 'ATOM')
      x= [x; str2double(s(33:38))];
      y= [y; str2double(s(41:46))];
      z= [z; str2double(s(49:54))];
   end
end
fclose(fid);
```