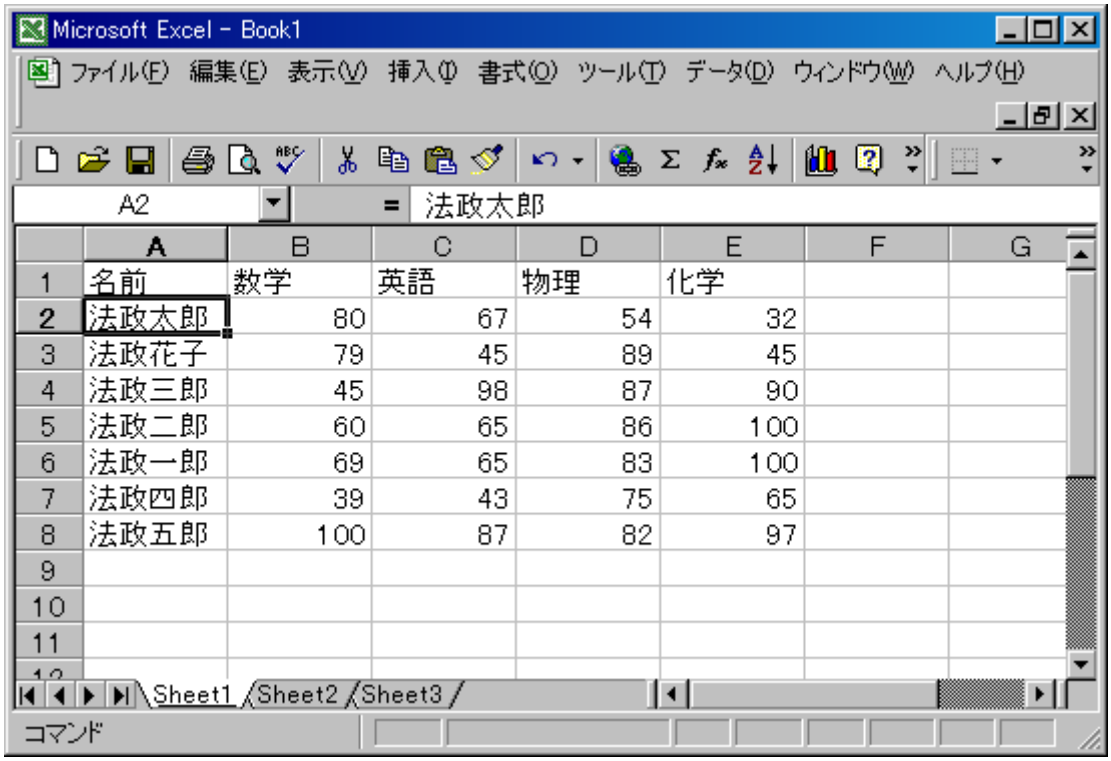
CELL 配列応用
MATLAB に Excel などで作成された大量のデータをインポートするには？

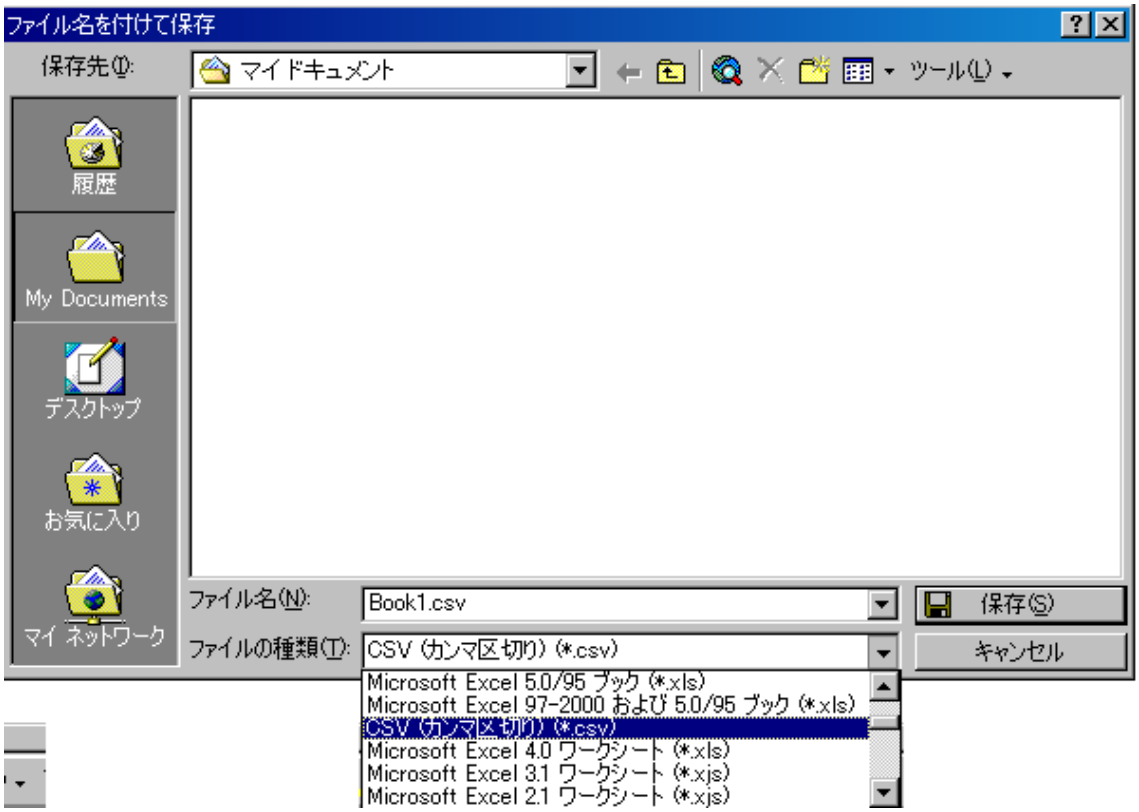
MATLAB で Excel などのデータを元に解析したい場合、標準関数 wk1read 関数、csvread 関数などが用意されているが、途中で文字が入っていたりすると読み込めなかったり、実用上は使いづらい。

そこでここでは、MATLAB 変数のセル配列と構造体使ったデータ読み込みの例を紹介する。
次のようなデータを MATLAB に取り込むことを考える。



	A	B	C	D	E	F	G
1	名前	数学	英語	物理	化学		
2	法政太郎	80	67	54	32		
3	法政花子	79	45	89	45		
4	法政三郎	45	98	87	90		
5	法政二郎	60	65	86	100		
6	法政一郎	69	65	83	100		
7	法政四郎	39	43	75	65		
8	法政五郎	100	87	82	97		
9							
10							
11							

Excel 形式のファイルを取り込むには、ActiveX などを使えば取り込むことができるが、ここでは、タイトル、名前などもデータとして取り込むため、一度 CSV 形式のファイルに変換し保存する。



ここでは、Book1.csv 形式で保存した例を示している。
CSV 形式とは、カンマと改行で区切られたデータ形式である。MATLAB では、CSV 形式のファイルを読み込む関数として CSVREAD 関数もあるが、このように一行目に漢字、２行目、３行目の最初に漢字があるようなファイルを読み込むことはできない。このような形式のファイルを読み込むためには、C 言語などでファイルを読み込む時と同様に fopen 関数などにより読み込む必要がある。
ここでは、構造体を使った簡単な例を順を示す。

```
%Book1.csv をオープンする。
fid=fopen('Book1.csv','r');
ii=1;
while(1)
%fgets 関数により 1 行づつ tline 変数に読み込む。
        tline=fgets(fid);
%読み込まれた 1 行分のデータからカンマ，の位置を見つけ出す。
        tlinenum=findstr(tline,',');
% カンマとカンマにはさまれた数値を見つけるため
%カンマの無い最初と最後の位置を追加する。
        tlinenum=[0 tlinenum length(tline)+1];
        if tline==-1,break;end
        for i=1:(length(tlinenum)-1)
%カンマとカンマに区切られたデータを抽出する
%データがもし数値であるならば str2num 関数により数値に変換する
%数値で無い場合 str2num 関数の出力は、空行列[]が出力される
```

```
                ttmp=str2num(tline([(tlinenum(i)+1):(tlinenum(i+1)-1)]));
%数値が空の場合には、そのまま ttline{ii}{i}に文字列としてセル配列に保存する。
                if isempty(ttmp)
                        ttline{ii}{i}=tline([(tlinenum(i)+1):(tlinenum(i+1)-1)]);
                else
                        ttline{ii}{i}=ttmp;
                end
        end
        ii=ii+1;
end
fclose(fid);
```

以上の操作により変数 ttline にセル配列として保存される。セル配列では、データの大きさやサイズなど任意のサイズで定義でき、数値、文字列など問わず保存できるが、行列ではないため、MATLAB でデータとして扱う場合若干使いづらい。ttline をタイプして見てみると、

```
>> ttline
ttline =
   {1x5 cell}   {1x5 cell}   {1x5 cell}   {1x5 cell}   {1x5 cell}   {1x5 cell}   {1x5 cell}   {1x5 cell}
```

となってしまう。１行目を見るには、

```
>> ttline{1}
ans =
   '名前'   '数学'   '英語'   '物理'   [1x6 char]
```

と中括弧でくくる必要がある。
そこでここでは、これらデータを分かりやすく構造体として保存しなおすとよい。
今、セル配列として ttline があると仮定し次の処理を行うとよい。なお、この場合、全てのデータを行列として保存する seisekidata{i}.all も追加してある。
データを行列に変換し保存したい場合には、cell2mat 関数を用いるとよい。

```
for i=1:length(ttline);
        seisekidata{i}.name=ttline{i}{1};
        seisekidata{i}.math=ttline{i}{2};
        seisekidata{i}.eng=ttline{i}{3};
        seisekidata{i}.physics=ttline{i}{4};
        seisekidata{i}.all=cell2mat(ttline{i}(2:end));
end
```

実行した結果できたセル配列 seisekidata を見てみると先ほどと同様に

```
>> seisekidata
seisekidata =
   [1x1 struct]   [1x1 struct]   [1x1 struct]   [1x1 struct]   [1x1 struct]   [1x1 struct]   [1x1 struct]   [1x1 struct]
```

となる。この場合一番最初のセル配列にデータ名などが保存される。
つまり、

```
>> seisekidata{1}
ans =
      name: '名前'
      math: '数学'
       eng: '英語'
   physics: '物理'
       all: [1x18 char]
```

として確認することができる。
１以降２からは、データがセル配列として保存される。

```
>> seisekidata{2}
ans =
      name: '法政太郎'
      math: 80
       eng: 67
   physics: 54
       all: [80 67 54 32]
```

たとえば、全科目データを取り出したい場合には、

```
>> seisekidata{2}.all
ans =    80    67    54    32
```

一番目の英語の点を取り出したい場合には、

```
seisekidata{2}.eng
または、
seisekidata{2}.all(2)
```

とすれば取り出すことができる。この時、１番目なのにセル配列を２と指定している理由は、１番目のセル配列には、タイトルなどが入っており、実際のデータは、２番目以降のセル配列に記録されているためである。もう一つの２は、英語の点のデータは、変数 all の中の２番目に記録されていることを意味する。
また、all のデータを元に全てをまとめて行列として再構成したい場合には、

```
>> alldata=[];for i=2:length(seisekidata);alldata=[alldata;seisekidata{i}.all];end;alldata
alldata =
   80    67    54    32
   79    45    89    45
   45    98    87    90
   60    65    86   100
   69    65    83   100
   39    43    75    65
  100    87    82    97
```

とすれば、MATLAB の配列として再現することができる。