

MATLAB IV

Computational Physics

MATLAB
(you need to know)
Lecture 4

Outline

- Program Output
- File Input and Output
 - Output to Files (*fprintf*)
 - Input from Files (*load*)
 - Input from Files using Import Data Wizard
- MATLAB Functions

Program Output

- MATLAB outputs variables and arrays by default in its own format.
 - `;` at end of line inhibits this.
- Use MATLAB function ***fprintf*** to write formatted output to screen.
- ***fprintf*** may be used with ***fopen*** and ***fclose*** to write an output file.

Output Example Program

```
% define an array - ; means don't print  
x = 0:1:10;  
  
% print out x using MATLAB default  
x  
  
% now write x to file my_file.dat  
  
% open the file my_file.dat for writing  
fid = fopen('my_file.dat', 'w')  
  
% now write each element of x  
for i=1:11  
    fprintf(fid, '%f \n', x(i)) % to file  
    fprintf( '%f \n', x(i))      % to screen  
end  
  
% finally close the file  
fclose(fid)
```

fopen "opens" a file for reading or writing. It returns *fid*, which is the file identifier used to reference the file throughout the program

fprintf is MATLAB function like C printf and fprintf.

Must "close" the file at end of script with *fclose*

About fprintf

- Format follows "C" standard
fprintf(file_id, 'format-string', variables)
- Format String
 - Characters (note % does NOT mean comment)
 - %f - floating point number
 - %d - integer
 - Number format: let *l* be number of spaces and *p* be number of decimals then specify float format: *%l.pf*
 - Special character for "new line": \n
- Example:

```
fprintf( '%5.2f', x) % prints value of x as xx.xx
```

MATLAB for General Plotting using *load* command

- MATLAB can be used to plot data saved in files.
- Steps:
 - Use ***load*** to read array from file
 - Create vectors from columns
 - Plot vectors

```
% my_data.dat contains  
% array with 3 cols and  
% N rows ... e.g.
```

```
%   x1  y1  z1
```

```
%   x2  y2  z2
```

```
%   ...  ...  ...
```

```
%   xN  yN  zN
```

```
load my_data.dat
```

```
% vector from first col
```

```
x = my_data(:,1);
```

```
% vector from second col
```

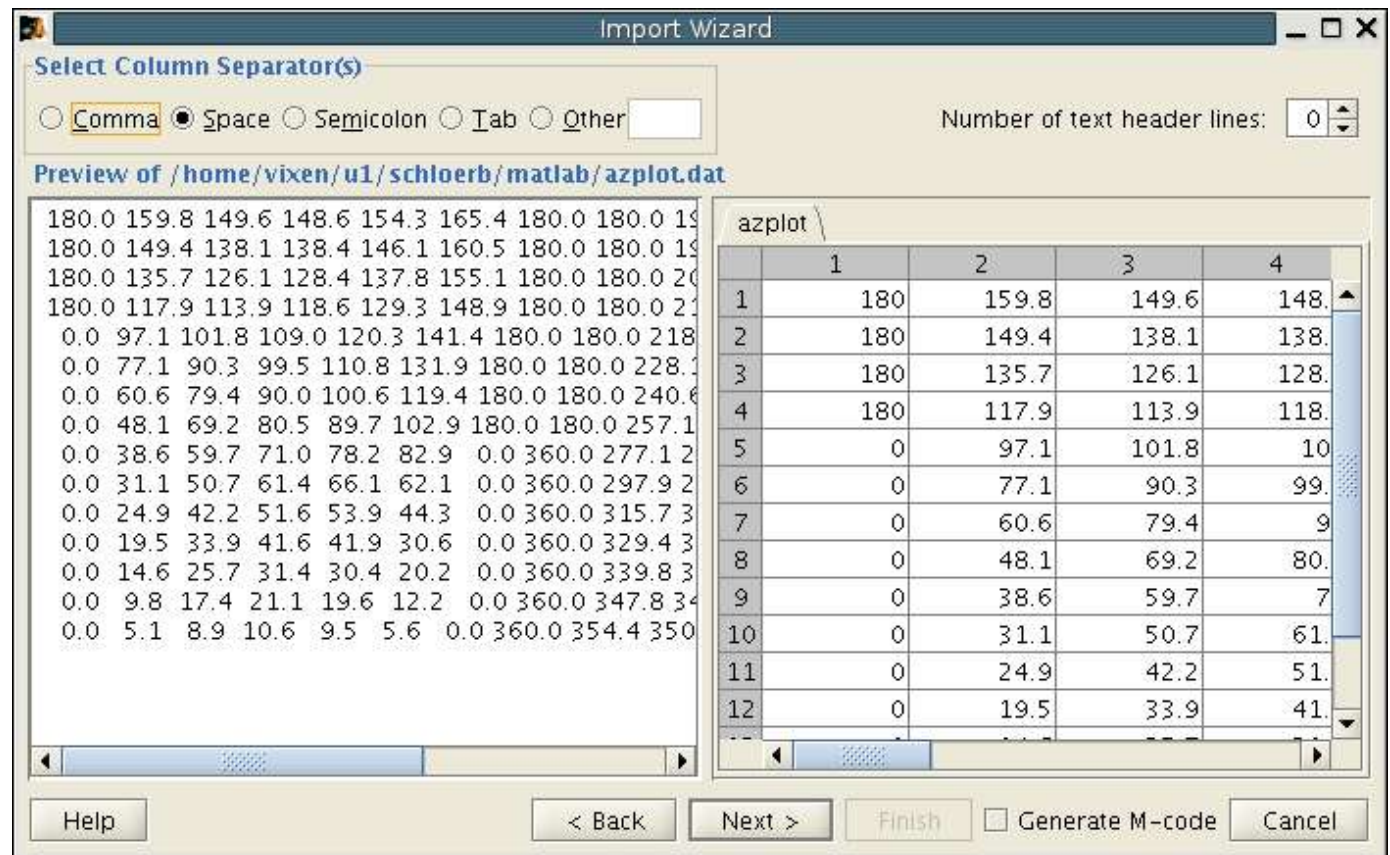
```
y = my_data(:,2);
```

```
plot(x,y);
```

Importing data using the Import Data Wizard

- Data may be imported from files using GUI

- Select Import Data...
- Enter File Name
- Select column separator
- Approve import...



MATLAB Functions

- Sometimes you find that same set of commands is repeated many times in your program.
- You can define a *function* in MATLAB to replace these lines with a single command
- Advantages
 - Simpler program - easier to debug
 - Function code only has to be checked in function - not several times within single program
 - Functions can be reused.

FILE: *target.m* The “main program”

```
% The “target” game
% Author: F. P. Schloerb

% set parameters for calculation
g = 9.8; %grav. acceleration
v0 = 100; %initial velocity
tolerance = 0.1; % tolerance for a hit

% randomly select the target distance.
rand('state',sum(100*clock)); % seeds random number
etrue = rand()*pi/2;
dtrue = cannon_ball_distance(etrue, g, v0);

% challenge the player
fprintf('Select cannon elevation to hit the target!\n');
fprintf('NOTE: g=%3.1f m/s/s; v= %6.1f m/s\n',g,v0);
fprintf('The target distance is: %f m\n',dtrue);

% now prompt the player for guesses
KeepTrying = true;
while (KeepTrying)
    e = input('Enter Elevation Angle (deg.): ');
    e = e*pi/180; %convert to radians
    d = cannon_ball_distance(e, g, v0);
    if (abs(d-dtrue)<tolerance)
        e = e/pi*180; %convert to degrees
        fprintf('HIT! elevation=%f distance= %f\n',e,d);
        KeepTrying = false;
    else
        fprintf('MISS! (error = %f )\n',d-dtrue);
    end
end
```

Function Example

```
function result=cannon_ball_distance(e,g,v)
% compute the time of flight
tof = 2*v*sin(e)/g;
% compute the distance
result = v*cos(e)*tof;
```

FILE:
cannon_ball_distance.m
is executed when
called by
***target.m* script**

Some Smart Things about the *target.m* example

- Code required for calculation is called in two different places => we don't have to replicate it and risk errors.
- Calculation of distance is separated from the “game”. The coding of the game is not dependent on the calculation in any way.
- Function script is flexible enough to allow change of parameters ... no “hard coding” of parameters.

