****

Principio del formulario

*Search for* [[Help]](http://lava-lamp.mit.edu:8765/help/)

Final del formulario

**How to write EXTERNAL FUNCTIONS and SCRIPTS**

EXTERNAL FUNCTIONS

==================

Matlab allows you to write your own functions. You can execute these just

like any other function in matlab. These functions exist in files whose

name ends in a ".m", and are therefore called m-files.

Most of the functions that are used in matlab are actually just m-files.

Therefore, you can see how they work and modify them to suit your needs.

To list a m-file, just

>> type fname

where fname is the function you want to look at. As an example look at,

>> type flipud

(Note: you can also see this by typing

more /mit/matlab/lib/matlab/flipud.m

at the athena% prompt.)

----------------------------------------

function y = flipud(x)

%FLIPUD Flip matrix in the up/down direction.

% FLIPUD(X) returns X with columns preserved and rows flipped

% in the up/down direction. For example,

%

% X = 1 4 becomes 3 6

% 2 5 2 5

% 3 6 1 4

%

% See also FLIPLR.

[m,n] = size(x);

y = x(m:-1:1,:);

----------------------------------------

Let us examine the structure of this file:

1. The first line begins with the word "function"

This tells matlab that this is function. If you leave this

out, matlab will treat the file as a "script" (see below).

2. Following the function declaration there is a list of output

arguments (in this case 1), a statement of the function name, and

a list of input arguments.

Output arguments are listed in the format, [V1,V2,V3... etc] (if

you have only 1 you don't need the brackets).

The name you use must be the same as the filename (minus the .m).

The input arguments are listed after the function name in the

format (x1,x2,x3,...etc) These variables are assigned values when

the function is called. By default all variables in m-files are

local, so you may call them whatever you want.

3. After the function declaration, there are a series of lines that

start with "%" (these are comments).

The first contiguous group of lines that begin with "%" are

treated as the help blurb for the function. You can see this by,

>> help flipud

When you write your own files, it is useful to use this "help"

facility to remind you of its syntax and purpose.

Here is an example of a simple m-file that will scramble the elements in a

vector, and output both the scrambled vector and a vector containing

indices used to descramble the vector.

----------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

scram.m

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function [Xs,Ir]=scram(X)

% SCRAM Scramble the elements in a vector.

% SCRAM(X) returns a scrambled vector and a vector

% containing the indices used to descramble it.

N=max(size(X));

I=-ones(1,N);

for j=1:N

Q=-1;

while Q~=0

tmp=round(rand\*N+.5);

Q=sum(tmp==I);

end

I(j)=tmp;

end

Xs=X(I);

Ir(I)=1:N;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

>> help scram

SCRAM Scramble the elements in a vector.

SCRAM(X) returns a scrambled vector and a vector

containing the indices used to descramble it.

>> Y=1:2:20

Y =

1 3 5 7 9 11 13 15 17 19

>> [Ys,I]=scram(Y)

Ys =

9 7 17 19 13 5 11 15 3 1

I =

5 4 9 10 7 3 6 8 2 1

>> Ys(I)

ans =

1 3 5 7 9 11 13 15 17 19

SCRIPTS

=======

If you leave out the "function" line, then a mfile is considered to be a

"script". Calling a script is equivalent to entering each of the lines in

the script at the ">>" prompt. All variables defined in the parent

workspace are accessible to the script and can be modified by the script.

In this sense, all variables are considered global in a script. This

feature is both handy and dangerous. You don't have to explicitly pass

arguments to a script, but if you assign a value to a variable in a script

that exists in the parent workspace, its value is overwritten.

PROGRAMMING HINTS

=================

1. Vectorize wherever possible to save immense amounts of computation time.

>> for i=1:500, % This took about 2 minutes

for j=1:500, % to complete.

A(i,j)=i\*j;

end

end

>> A=(1:500)'\*(1:500); % Takes less than 1 second

% to complete.

Wherever possible, use matrix operations rather than loops.

2. Functions can take variable numbers of arguments. "nargin" and

"nargout" are permanent variables that contain the number of input and

output arguments.

3. You can do almost anything in a mfile that you can do from the matlab

prompt including; Loading and saving, issuing system commands ("!command",

see ">> type print"), writing to files (fprintf), call other mfiles

(including itself), plot, etc..

4. An easy way to call programs written in other languages is:

o Save variables in a file

o Run external program which reads the file and writes output to

another file.

o Load the data back in.

For example:

function y = garfield(a,b,q,r)

save gardata a b q r

!gareqn

load gardata

5. Editing a mfile causes matlab to recompile it the next time it is

called in matlab. You can, therefore, run a function, modify the mfile,

and then run the new version. Note: Modifying a mfile during the execution

of the function does not affect the current process.

6. Use the Matlab debugging commands to help track down programming

errors. For more information, consult the "Debugging MATLAB .m files"

answer in the stock answers browser.

|  |  |
| --- | --- |
| *Comments to* [*othomas@mit.edu*](mailto:othomas@mit.edu) *Web targets last updated Thursday, 31 December 1998* | [MIT](http://web.mit.edu/) |