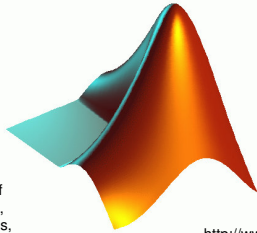


MATLAB® - The Language of Technical Computing

Robert White



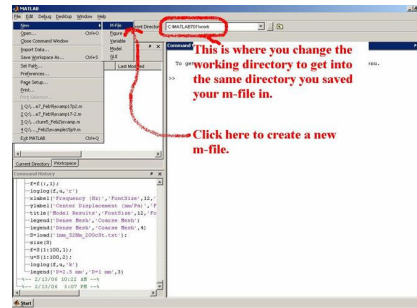
<http://www.mathworks.com>

Session #2: m-files, if statements, for loops, functions, polynomials, algebraic systems, and ODEs

© 2009, R. White

Programming: m-files

M-files are just text files containing a list of Matlab commands to execute in sequence.



© 2009, R. White

Programming: displaying text

`disp()` Displays formatted text to the Matlab prompt. Takes a single string as an argument.

```
>> disp('Hello world')
```

Hello world

If you want to concatenate strings, put them in square parentheses:

```
>> disp(['Hello world' ' again'])
```

Hello world again

If you want to make a number into a string, use the `num2str()` command:

```
>> x=3
```

x =

3

```
>> disp(['Hello world for the ' num2str(x) 'rd time'])
```

Hello world for the 3rd time

© 2009, R. White

Programming: user input

```
A=input('Your prompt here: ','s')
```

This command prompts the user for input. The 's' argument is optional ... without it, the command expects a numerical response from the user. With it, the command expects a string response from the user.

```
username=input('What is your name? ','s');
```

%The 's' is used to indicate a string is expected

```
userage=input('How old are you? ');
```

```
disp([username ' is ' num2str(userage) ' years old.'])
```

© 2009, R. White

Programming: if statements

An "if" statement has the following syntax in Matlab:

```
if (logical expression),  
DO THIS IF LOGICAL EXPRESSION IS TRUE  
else  
DO THIS IF LOGICAL EXPRESSION IS FALSE  
end
```

In Matlab, a logical "true" is the scalar "1". A logical "false" is the scalar "0". If the logical expression evaluates to a vector or matrix, all elements in the vector or matrix must be "1" for it to be considered true.

```
>> if [1 1 0]  
disp('True')  
else  
disp('False')  
end  
False  
>> if [1 1 1]  
disp('True')  
else  
disp('False')  
end  
True
```

© 2009, R. White

Programming: if statements

Examples of logical expressions:

```
x>4    x<4    x==4 (x is equal to 4)    x~=4 (x is not equal to 4)  
x<=4    x>=4
```

Note: "==" is the logical "is equal to" operation. This is NOT the same as "=" which is the "define equal to" operation!!!

```
strcmp(mystring,'mechanical')    isreal(x)    ~isreal(x)  
x>4 & x<8    x>8 | x<4    isinf(1/0)
```

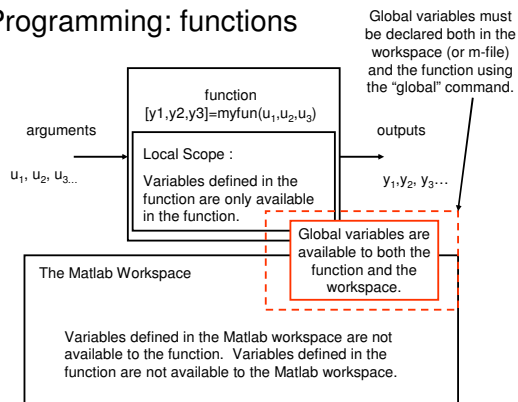
Notes: "~" is the not operator. You can put it in front of any logical expression to negate it. "&" is the "and" logical operation "|" (shift backslash on most computers) is the "or" logical operation. `strcmp()` compares two strings to see if they are the same.

```
rem(x,2)
```

`rem(x,y)` is the remainder of `x/y` so `rem(x,2)=0` if `x` is even, it = 1 if `x` is odd ... hence `rem(x,2)` is "true" when `x` is odd.

© 2009, R. White

Programming: functions



© 2009, R. White

Programming: functions

```

1 %This function solves the quadratic equation using the quadratic formula.
2 %function X=quadratic(A,B,C)
3
4 function X=quadratic(A,B,C)
5
6 %A=1;
7 %B=2;
8 %C=3;
9
10 %Different cases for A=0 and otherwise:
11 if A==0,
12     X=-C/B;
13 else
14     X(1)=(-B+sqrt(B^2-4*A*C))/(2*A);
15     X(2)=(-B-sqrt(B^2-4*A*C))/(2*A);
16 end
17
    
```

Functions are m-files that start with the word "function". They can now take arguments and return results.

© 2009, R. White

Programming: functions

Two notes:

1. The filename must be the same as the function name. So, if your function is named "quadratic", your filename must be "quadratic.m"
2. The function only has local scope... it does not have access to variables in the workspace, and will not overwrite variables in the workspace. It only has access to the arguments you pass it, and it only modifies the variable that it returns to.

```

>> quadratic(1,2,3)
ans =
-1.0000 + 1.4142i -1.0000 - 1.4142i
>> y=quadratic(5,6,7)
y =
-0.6000 + 1.0198i -0.6000 - 1.0198i
    
```

Stop and do problem #2 and #3.

© 2009, R. White

Polynomials

Matlab defines polynomials using a vector of coefficients in descending order. So, the polynomial:

$$3x^3 - 5x^2 + 7x + 3$$

Is defined in Matlab by:

```
>> P=[3 -5 7 3];
```

Now I can do things like find the roots of the polynomial:

```
>> P=[3 -5 7 3];
>> roots(P)
```

```
ans =
1.0000 + 1.4142i
1.0000 - 1.4142i
-0.3333
```

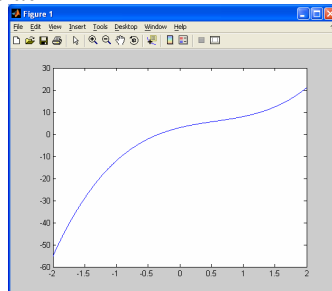
© 2009, R. White

Polynomials

I can also evaluate the polynomial at a bunch of points using the *polyval* function

```
>> x=linspace(-2,2,1000);
>> y=polyval(P,x);
>> plot(x,y)
```

There are a bunch of polynomial functions. Type *help polyval*



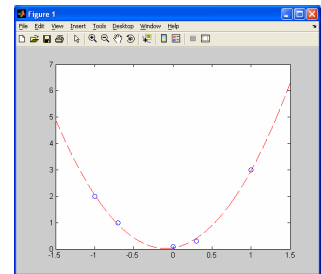
© 2009, R. White

Polynomials: fitting

I can also fit data with an n^{th} order polynomial (here I use a 2nd order... a parabola):

```
>> XData=[-1 -0.7 0 0.3 1];
>> YData=[2 1 0.1 0.3 3];
>> plot(XData,YData,'o')
>> hold on
>> P=polyfit(XData,YData,2)
P =
2.4703 0.4603 0.0422
>> x=linspace(-1.5,1.5,1000);
>> y=polyval(P,x);
>> plot(x,y,'r--')
```

Stop and do problem #4.



© 2009, R. White

Solving linear algebraic equations

Solving a linear system of equations in the form

$$Ax=b$$

is easy. For example, to solve the following for a, b, and c:

$$\begin{aligned} -a + 4b - 6c &= \pi \\ 8a + b &= 2 \\ \cos(\pi/6)a - e^{2.3}c &= 0 \end{aligned}$$

```

>> A=[-1 4 -6;8 1 0;cos(pi/6) 0 -exp(2.3)];
>> b=[pi;2;0];
>> x=A\b
    0.1449
    0.8405
    0.0126
    
```

So, $a=0.1449$, $b=0.8405$, $c=0.0126$

Stop and do problem #5.

© 2009, R. White

Solving nonlinear algebraic equations

A nonlinear system of equations can always be written in the form

$$f_1(x_1, x_2, \dots) = 0$$

$$f_2(x_1, x_2, \dots) = 0$$

...

For example:

$$\begin{aligned} x^2 + 2y^2 &= 3 && \longrightarrow && x^2 + 2y^2 - 3 = 0 \\ y &= 4x^3 + 5x^2 + 6x + 1 && && 4x^3 + 5x^2 + 6x + 1 - y = 0 \end{aligned}$$

© 2009, R. White

Solving nonlinear algebraic equations

$$x^2 + 2y^2 - 3 = 0$$

Set this up as a Matlab function:

$$4x^3 + 5x^2 + 6x + 1 - y = 0$$

```

function F=myfun(X)
1
2 a=1;
3 b=2;
4 c=3;
5 d=4;
6 e=5;
7 f=6;
8 g=7;
9
10 x=X(1);
11 y=X(2);
12
13 F(1)=a*x^2+b*y^2-c;
14 F(2)=d*x^3+e*x^2+f*x+g-y;
    
```

© 2009, R. White

Solving nonlinear algebraic equations

Solve it using the *fsolve* function:

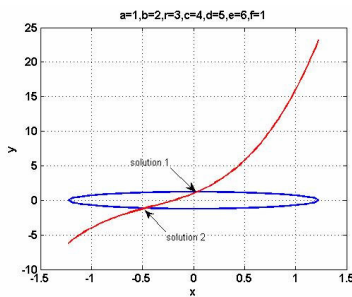
```

>> Evolve='myfun',[1 1]
Optimization terminated: First-order optimality is less than options.TolFun.
ans =
    0.0363    1.2245

>> Evolve='myfun',[-1 -1]
Optimization terminated: First-order optimality is less than options.TolFun.
ans =
   -0.4815   -1.1765
    
```

© 2009, R. White

Solving nonlinear algebraic equations



There are two solutions: Matlab finds the one closest to the guessed values.

Stop and do problem #6.

© 2009, R. White

Solving ordinary differential equations

A system of ordinary differential equations can always be written in the form

$$\dot{x}_1 = f_1(x_1, x_2, \dots, t)$$

$$\dot{x}_2 = f_2(x_1, x_2, \dots, t)$$

...

For example:

$$m\ddot{x} + b\dot{x} + k_1x + k_3x^3 = A \sin(\omega t)$$

Becomes:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(-bx_2 - k_1x_1 - k_3x_1^3 + A \sin(\omega t)) \end{aligned}$$

© 2009, R. White

Solving ordinary differential equations

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{1}{m}(-bx_2 - k_1x_1 - k_3x_1^3 + A\sin(\omega t))$$

Below is a Matlab function defining this set of ODEs:

```

1 function xdot=duff(t,x)
2 A=100;
3 m=1;
4 k1=100;
5 k3=1*k1;
6 b=1;
7 omega=10;
8
9 xdot(1)=x(2);
10 xdot(2)=(1/m)*(-b*x(2)-k1*x(1)-k3*(x(1))^3+A*sin(omega*t));
11
12 xdot=xdot';
13

```

© 2009, R. White

Solving ordinary differential equations

To solve, I call the ode45 Runge-Kutta solver with the name of the function ('duff'), the time limits (t=0 to 5), and the initial conditions (x1(0)=0, x2(0)=0).

```

>> [t,soln]=ode45('duff',[0 5],[0 0]);
>> plot(t,soln(:,1))

```

I plotted the first column of the solution matrix, `soln(:,1)` which corresponds to the solution for x_1 . The solution for x_2 is, naturally, `soln(:,2)`

Stop and do problems #7, 8 and/or 9.

© 2009, R. White

