Nuts and bolts of Matlab

Akashdeep Kamra & Utkarsh Upadhyay
Indian Institute of Technology, Kanpur

May 21, 2009

1 Differential Equations

Consider the following equations from the *ode45* help section:

$$\begin{aligned} y_1' &= y_2\, y_3 && (1)\\ y_2' &= -y_1\, y_3 && (2)\\ y_3' &= -0.51\, y_1\, y_2 && (3)\\ \equiv \dot{Y} &= f(t, Y) && (4) \end{aligned}$$

with the initial conditions:

$$\begin{aligned} y_1(0) &= 0 && (5)\\ y_2(0) &= 1 && (6)\\ y_3(0) &= 1 && (7) \end{aligned}$$

They are solved in the following way:

```
function dy = f(t,y)      % The derivative function.
      dy = zeros(3,1);    % a column vector
      dy(1) = y(2) * y(3);
      dy(2) = -y(1) * y(3);
      dy(3) = -0.51 * y(1) * y(2);
end
```

and on the prompt:

```
>> [T,Y] = ode45(@rigid,[0 12],[0 1 1]);
>> plot(T,Y(:,1),'-',T,Y(:,2),'-.',T,Y(:,3),'.');
```

1.1 Question 1

Solve and plot the solution of the following problem, which is a *Aging Spring problem*:
To see the solution in Mathematica, visit *http://demonstrations.wolfram.com/TheAgingSpring/*

$$mx'' + cx' + k(t)x = a\,cos(2\pi ft) \quad (8)$$
$$k(t) = k_1 + (k_0 - k_1)e^{-\epsilon t} \quad (9)$$

Optional: Can you explain physically what is it that the equation says? Can you suggest a few initial values of the constants which will yield familiar solutions?

Solve the equation for the following values:

$m = 3.4,\ c = 2.5,\ a = 13,\ f = 0.6,\ k_0 = 0.75,\ k_1 = 4.5$

and plot the position (x), with respect to $t \in [0, 11]$ and with the initial conditions $x(0) = 1,\ x'(0) = -9$ to see how complicated the solution can actually get.

Hint: Remember that any higher order equation can always be expressed in a matrix form (equation 4) which involves only one differentiation in each row.

1.2 Question 2

However, for some problems, *ode45* is not enough (called *stiff* problems[1]). Again, from the example problem set, try *ode45* on the follwing problem for the time period $t \in [0, 3000]$ (*Van der Pol* equations):

$$y_1' = y_2 \quad (10)$$
$$y_2' = 1000(1 - y_1^2)y_2 - y_1 \quad (11)$$

Initial condition:

$$y_1(0) = 2 \quad (12)$$
$$y_2(0) = 0 \quad (13)$$

Also try *ode*15*s* on the problem.

[1]There are regions in the domain of t where the rate of change is very low

2 Symbolic calculations

Requires the Symbolic Calculation Toolbox

```
>> syms x a b c
>> y = a*x^2 + b*x + c

y =

a*x^2+b*x+c


>> subs(y,1)

ans =

a+b+c


>> solve(y)

ans =

 -1/2*(b-(b^2-4*a*c)^(1/2))/a
 -1/2*(b+(b^2-4*a*c)^(1/2))/a

>> z = subs(y,x,2)

z =

4*a+2*b+c


>> subs(z,a,1)

ans =

4+2*b+c

>> diff(y)

ans =

2*a*x+b


>> int(y)

ans =

1/3*a*x^3+1/2*b*x^2+c*x
```

2.1 Question 3

Find the symbolic solutions for:

1. The inverse of a 2D matrix
2. The equations $[a\ b; c\ d] \times [p; q] + [e; f] = 0$ for p and q.
3. The solution of a cubic and a bi-quadratic equation in its most general form.
4. The solution of $ae^x + bx = c$

3 Really plotting functions

Try the following:

```
>> z = x^2 + 2*x + 3

z =

x^2+2*x+3


>> ezplot(z,[-5 5]);
>> ezplot(@(x) sin(x)*x);
>> ezplot( @sin, @cos );
```

Explore the *ez* family of plotting functions and using them, do the following:

- Plot $y = 2e^x - 4x + 1$ for $x \in [-10, 10]$
- Plot $z = sin\theta$ on the unit circle, i.e. $x^2 + y^2 = 1$.

4 Profiler

Find and start the profiler. Then get hold of the file *zero_hold_slow.m* and take a look at it. (It can be found at *home.iitk.ac.in/ utkarshu/zero_hold_slow.m*).

Create the test matrix:

```
>> z = zeros(100000,1);
>> z(1:5:100000) = 1:5:100000;
```

4.1 Question 4

The objective here is minimizing the running time of the *zero_hold_slow* function while making *minimum* changes to the source code. We will be extending the function to become a first order hold too later.

4.1.1 Question 4.1

Run the profiler on the program to find the most expensive function.

4.1.2 Question 4.2

Make minimal changes to the function while keeping the function extension in mind, and compare timings.

4.1.3 Question 4.3

Make changes to the program to make the hold a first order one, i.e., join the two points by a line segment instead of just repeating the first value.

Hint: Use *linspace*

5 Curve fitting

You have already seen how to use *polyval* to evaluate polynomials. We will describe another way of doing the same.

```
>> z = poly(1)

z =

     1    -1

>> poly2sym(z)

ans =

x-1

>> y = 2*x^2 + 3*x + 5;
>> sym2poly(y)

ans =

   2 3 5
```

Also, finally there is the small task of fitting points to curves.

5.1 Question 5

Plot the function $y = mx + c$. Using *ezplot* for $m = -2.5$ and $c = 1$.

5.2 Question 6

Make a program (script) which allows user to input points he wants, the desired degree of a polynomial and then finds the polynomial that best fits the points and then plot the points as well as the estimated function.

Hints:

- Take a look at: *polyfit*
- For ease of input, consider: *ginput*
- Recall: *input* function and its case-sensitivity
- Try both *ezplot* as well as *polyval* and *plot* and notice the differences.