StatOD MATLAB Overview

Brandon Jones
September 9, 2009

What we’re going to cover:

General MATLAB Introduction

ode45() (and the odeXX suite of integrators)

Symbolic Toolbox

reshape()
And now for something *completely different*…

Get into small groups (3-4 people)

Select a recorder

In your groups, list as many advantages of MATLAB you can think of

In your groups, list as many disadvantages of MATLAB you can think of

MATLAB is a vectorized programming language

Which would you prefer?

### C/C++

```c
for( i=0; i<3; i++ ){
    for( j=0; j<3; j++ ){
        for( k=0; k<3; k++ ){
            out[i][j] += A[i][k]*B[k][j];
        }
    }
}
```

### MATLAB

```matlab
out = A*B;
```
Many MATLAB commands have a vectorized implementation

```
x = 1:1:10;  % x = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
x_sqrt = sqrt(x);
x_sin = sin(x);
x_log = log10(x);
and many others...
```

Some vectorized programming requires a little extra thought than other languages

**Root Mean Square (RMS)**

\[ x_{rms} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}} \]

```
acc_sum = 0.0;
for i = 1:length(x)
    acc_sum = acc_sum + x(i)*x(i);
end
x_rms = sqrt( acc_sum/length(x) );
```
Some vectorized programming requires a little extra thought than other languages.

### Root Mean Square (RMS)

\[
x_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}
\]

### (better) MATLAB

\[
x_{\text{rms}} = \sqrt{\text{sum}(x.*x)/\text{length}(x)};
\]

Now it’s your turn...

### A random function...

\[
y = e^{-\left(\frac{\sum_{i=1}^{N} x_i}{N}\right)}
\]

### MATLAB Solutions

\[
y = \exp(-\text{sum}(x)/\text{length}(x));
\]

\[
Y = \exp(-\text{mean}(x));
\]
MATLAB is inefficient with loops (for multiple reasons)

Inefficient MATLAB

```matlab
x = 1:1:100;
for i = 1:length(x)
    y(i) = sqrt(x(i));
end
```

Better MATLAB

```matlab
x = 1:1:100;
y = zeros(size(x));
for i = 1:length(x)
    y(i) = sqrt(x(i));
end
```

Time for a quick demonstration of some MATLAB plotting tools
Use `fprintf()` to write to the screen (or a file)

MATLAB `fprintf()`

```matlab
x = 1:1:10;
for i = 1:length(x)
    fprintf(’x(%)2d = %g
’, i, x(i));
end
```

MATLAB `fprintf()` To File

```matlab
x = 1:1:10;
fid = fopen(‘output.txt’, ‘w’);
for i = 1:length(x)
    fprintf(fid, ’x(%)2d = %g
’, i, x(i));
end
```

Field widths can be specified using format identifiers

%X.Ye : X is minimum field width, Y is the number of decimal places to display

Now for some practice!

```plaintext
my_var = 12345.67891011…
```

<table>
<thead>
<tr>
<th>Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>%9.3f</td>
<td>12345.679</td>
</tr>
<tr>
<td>%8.1f</td>
<td>_12345.7</td>
</tr>
<tr>
<td>%2.3f</td>
<td>12345.679</td>
</tr>
</tbody>
</table>
Here are some general programming tips...

Include comments!!!

Keep units consistent (i.e. angles in radians)

Function and variable names should be descriptive

“Derivative” or “Integrator” are too generic

two_body(), two_body_J2(), two_body_J2drag(), etc.

What we’re going to cover:

General MATLAB Introduction

ode45() (and the odeXX suite of integrators)

Symbolic Toolbox

reshape()
MATLAB includes a suite of tools to help solve the ‘initial value problem’

Given: \( \vec{x}_0 \)

What is: \( \vec{x}_t \)

General description of odeXX() suite:


Initial value solvers require knowledge of system dynamics

IVS Determines: \( \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \)

IVS Needs: \( \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \)

Gravity, Drag, Thrust, etc.
Most integrators are either fixed-step or variable step

Fixed Step:  Variable Step:

Why do we want to use a variable step integrator?

odeXX() Suite

Suite of numerical integration routines

The XX refers to the order of the scheme used
ode45() compares a 4th order to a 5th order to determine time step

Higher order does not necessarily provide higher accuracy
reentry problem (problem becomes “stiff”)

ode45() is fine for this course
Need to set the integration tolerance to select the time step

```matlab
tol = 1e-13 % states match at least 13 digits
options = odeset('RelTol', tol);
[time,state] = ode45(@two_body, [time0, timeF], ... 
x0, options);
```

Make sure you pass the options to ode45()!!

What we’re going to cover:

- General MATLAB Introduction
- `ode45()` (and the `odeXX` suite of integrators)
- Symbolic Toolbox
- `reshape()`
The symbolic toolbox provides tools for the manipulation of equations

```matlab
syms x y z
radius = sqrt( x^2 + y^2 + z^2 );
vec = [ x; y; z ];
drdx = diff( radius, x ); % dR/dx
radius = int( drdx, 'x' );
radius = int( drdx, 'x', 0, 1 );
drdv = jacobian( radius, vec );
```

Group exercise:

Given: \[ \ddot{\mathbf{r}} = -\frac{\mu}{|\mathbf{r}|^3} \mathbf{r} \]

How would you solve for the A matrix?

Now that you have solved for it, how would you use it in your code?

(Do not solve for it repeatedly in your derivative function!)
Use `fprintf()` to get the symbolic A matrix into your code

```matlab
fid = fopen('A_matrix.txt', 'w');
for i = 1:length(A(:,1))
    for j = 1:length(A(1,:))
        if( A(i,j) ~= 0 )
            fprintf(fid, 'A(%d,%d)=%s;
', i, j, char(A(i,j)));
        end
    end
    fprintf(fid, '\n');
end
fclose(fid);
```

MATLAB: Store A Matrix

What we’re going to cover:

- General MATLAB Introduction
- `ode45()` (and the `odeXX` suite of integrators)
- Symbolic Toolbox
- `reshape()`
reshape() is used to easily change the dimensions of a matrix

MATLAB: reshape() Example

```matlab
>> x = [ 1, 2; 3, 4 ];
>> y = reshape( x, 4,1 );
y =
   1
   2
   3
   4
```

reshape() cannot change the number of elements in the variable

Used to convert $\Phi$ from a matrix to a vector, thus it can be numerically integrated in ode45()