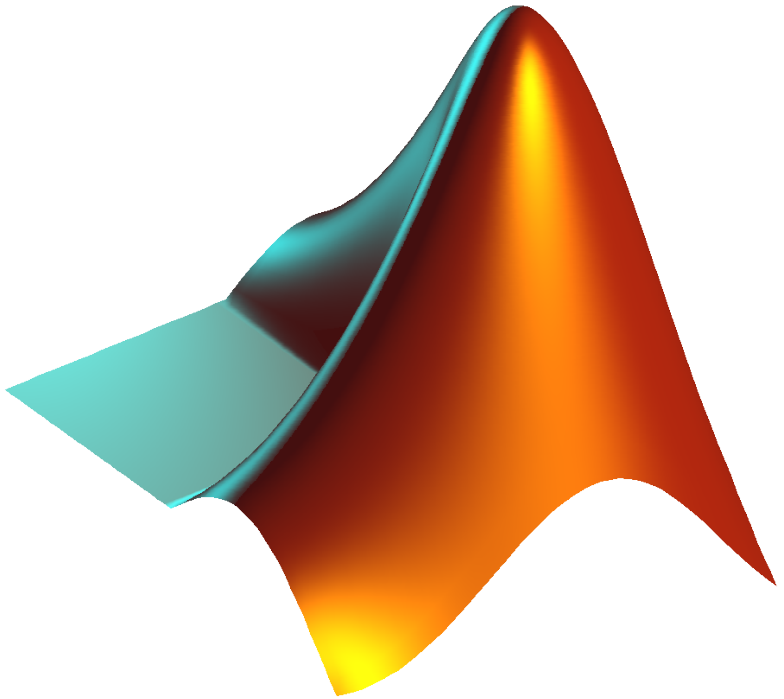
StatOD MATLAB Overview

Brandon Jones

September 14, 2011



MATLAB is a vectorized programming language

Which would you prefer?

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

```
out = A*B;
```

Many MATLAB commands have a vectorized implementation

MATLAB

```
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)

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

(bad) MATLAB

```
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)

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

(better) MATLAB

```
x_rms = sqrt(sum(x.*x)/length(x));
```

MATLAB is inefficient with loops (for multiple reasons)

Inefficient MATLAB

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

Better MATLAB

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

Use fprintf() to write to the screen (or a file)

MATLAB fprintf()

```
x = 1:1:10;
for i = 1:length(x)
      fprintf('x(%2d) = %g\n', i, x(i));
end
```

MATLAB fprintf() To File

```
x = 1:1:10;
fid = fopen('output.txt', 'w');
for i = 1:length(x)
      fprintf(fid, 'x(%2d) = %g\n', 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 examples…

my_var = 12345.67891011…

%9.3f		12345.679
%8.1f	⟩⟩⟩➡	_12345.7
%2.3f		12345.679

Here are some general programming tips…

Include comments!!!

Keep units consistent (e.g. 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: $\vec{x}_\circ$

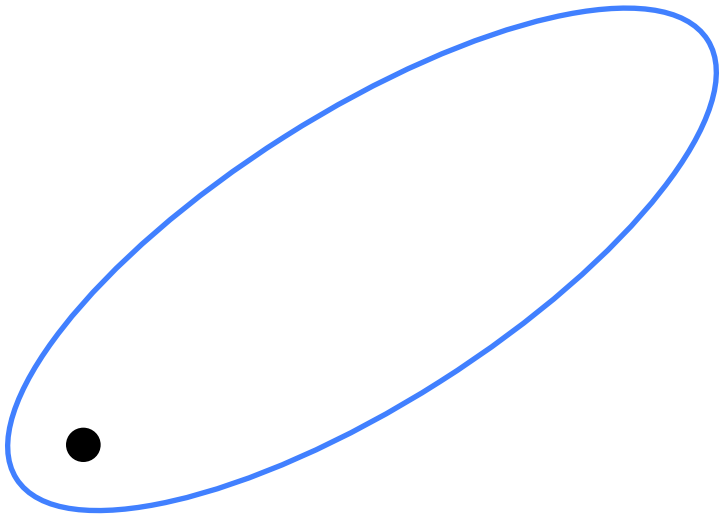
What is: $\vec{x}_t$

General description of odeXX() suite:

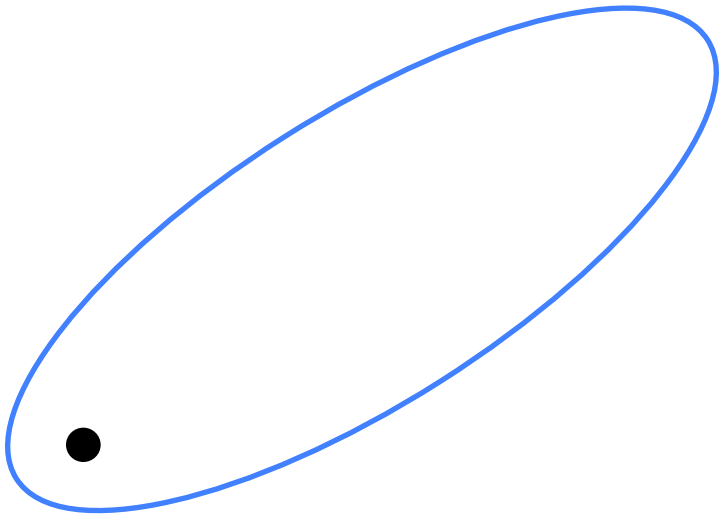
Shampine and Reichelt, "The MATLAB ODE Suite, " SIAM Journal of Scientific Computing, v. 18, p. 1-22, 1997.

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 );
```

Use fprintf() to get the symbolic A matrix into your code

MATLAB: Store A Matrix

```
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;\n', ...
                        i, j, char(A(i,j)));
      end
      fprintf(fid, '\n');
end
fclose(fid);
```

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

```
>> x = [ 1, 2; 3, 4 ];

>> y = reshape( x, 4,1 );

y =
        1
        3
        2
        4
```

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

Used to convert Φ from a matrix to a vector, thus it can be easily propagated in ode45()