Numerical Methods in MATLAB

Center for Interdisciplinary Research and Consulting

Department of Mathematics and Statistics

University of Maryland, Baltimore County

www.umbc.edu/circ

Winter 2008

Mission and Goals: The Center for Interdisciplinary Research and Consulting (CIRC) is a consulting service on mathematics and statistics provided by the Department of Mathematics and Statistics at UMBC. Established in 2003, CIRC is dedicated to support interdisciplinary research for the UMBC campus community and the public at large. We provide a full range of consulting services from free initial consulting to long term support for research programs.

CIRC offers mathematical and statistical expertise in broad areas of applications, including biological sciences, engineering, and the social sciences. On the mathematics side, particular strengths include techniques of parallel computing and assistance with software packages such as MATLAB and COMSOL Multiphysics (formerly known as FEMLAB). On the statistics side, areas of particular strength include Toxicology, Industrial Hygiene, Bioequivalence, Biomechanical Engineering, Environmental Science, Finance, Information Theory, and packages such as SAS, SPSS, and S-Plus.

This document is under constant development and will periodically be updated. Standard disclaimers apply.

Acknowledgements: We gratefully acknowledge the efforts of the CIRC research assistants and students in Math/Stat 750 Introduction to Interdisciplinary Consulting in developing this tutorial series.

1 Numerical ODEs

In this section we discuss numerical ordinary differential equations in Matlab. Matlab provides a number of ODE solvers; we will focus our attention to `ode45` which uses a four stage Runge-kutta method to solve a give ordinary differential equation. We will first see how ones an initial value problem of form

$$\frac{dy}{dt} = f(t, y),$$
$$y(t_0) = y_0.$$

We can solve such problems using

```
[T Y]=ode45(f, tspan, y0)
```

In the above syntax, the input argument `f` specifies the right hand side function of the differential equations, `tspan` is the time interval in which we want to solve the equation, and `y0` is the initial value. The output argument `Y` gives the numerical solution over the discretized time interval `T`. Consider the following problem,

$$\frac{dy}{dt} = t - y,$$
$$y(0) = 1.$$

One can solve this problem analytically to get the solution, $y(t) = 2e^{-t} + t - 1$. To solve the problem numerically, we can use

```
>> f = @(t, y)(t - y)
f =
    @(t, y)(t - y)
>> y0=1;
>> [T Y] = ode45(f, [0 2], y0);
```

The reader can see how good the numerical solution is by plotting both the numerical solution and the true solution in the same plot; better yet one can compute and plot the approximation error to get a better picture of how good the solution was. The following Matlab code can used to solve the above problem; it also plots the numerical solution, the true solution, and the approximation error (see Figure 1).

```
function testode
y0 = 1;
[T Y] = ode45(@frhs, [0 2], y0);
y = ftrue(T);

subplot(311);
plot(T, Y);
```

```
title('Approximate Solution');

subplot(312);
plot(T, y);
title('True Solution');

subplot(313);
plot(T, abs(Y - y));
title('Approximation Error');

% subfunctions
function f=frhs(t, y);
f = t - y;

function f=ftrue(t);
f = 2*exp(-t)+t-1;
```

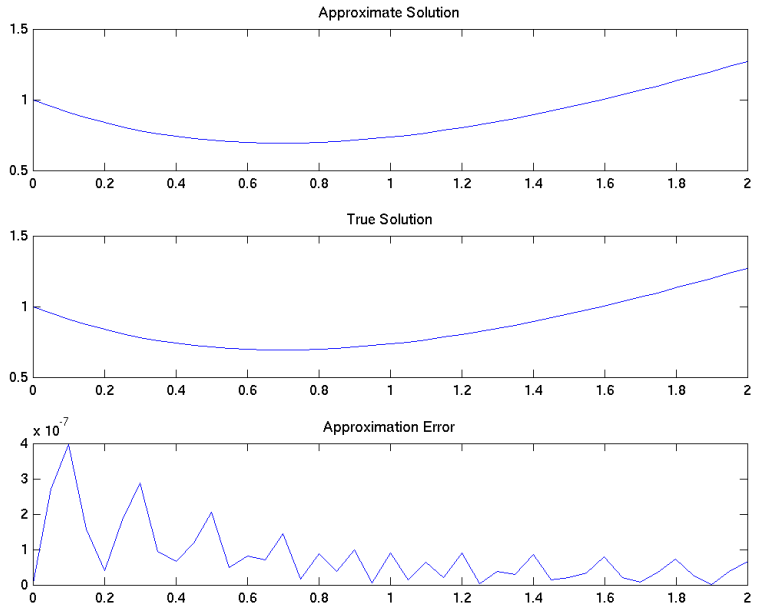

Figure 1: Using ode45 to solve an ODE

We can also solve systems of ODEs using `ode45`. To illustrate the idea we solve a classical predator-prey system. Let's consider the interaction of foxes and rabbits, where foxes are predators and rabbits are the prey. Denote, the rabbit population at any time by $y_1(t)$ and the fox population by $y_2(t)$. The system of differential equations

modeling the dynamics of this predator-prey system is given by the following

$$\frac{dy_1}{dt} = gy_1 - d_1 y_1 y_2$$
$$\frac{dy_2}{dt} = -d_2 y_2 + c y_1 y_2$$

In above equations the parameters g, d_1, d_2, and c denote:

1. g = natural growth rate of rabbit population in absence of foxes.
2. d_1 = the rate at which foxes die in the absence of rabbits.
3. d_2 = the death rate per each (deadly) encounter of rabbits due to foxes.
4. c = the contribution to fox population of each (food making) encounter of rabbits and foxes to fox population.

In addition to specifying the model parameters, we also need to specify the initial population of foxes and rabbits at $t = 0$. Let's choose the model parameters as below:

- $g = 0.04$;
- $d_1 = 0.001$;
- $d_2 = 0.1$;
- $c = 0.002$;

Also, we assume the initial populations start at $y_1(0) = 100$ and $y_2(0) = 100$. To solve the resulting initial value problem, we can use `ode45`; the Matlab function `predatorprey` provided below solves the problem using `ode45` and plots the populations of foxes and rabbits on the same plot (Figure 2); moreover, the function creates a phase-plane diagram (Figure 2) which is a useful tool in analyzing such systems.

```
function predatorprey

[T,Y] = ode45(@yprime,[0 100],[100 100]);
subplot(2,1,1);
plot(T,Y(:,1),'-', T,Y(:,2), '--');
title('Population Dynamics of Foxes and Rabbits');
legend('Rabbit Population', 'Fox Population');
xlabel('t');
ylabel('Population');
grid on;

subplot(2,1,2);
plot(Y(:,1), Y(:,2));
title('Phase Plane Diagram for the fox-rabbit population');
xlabel('Rabbits');
ylabel('Foxes');
grid on;

%rhs function
function dy = yprime(t, y)
g = 0.4;
d1 = 0.001;
d2 = 0.1;
c = 0.002;
dy = [g*y(1)-d1*y(1)*y(2);
      c*y(1)*y(2) - d2*y(2)];
```

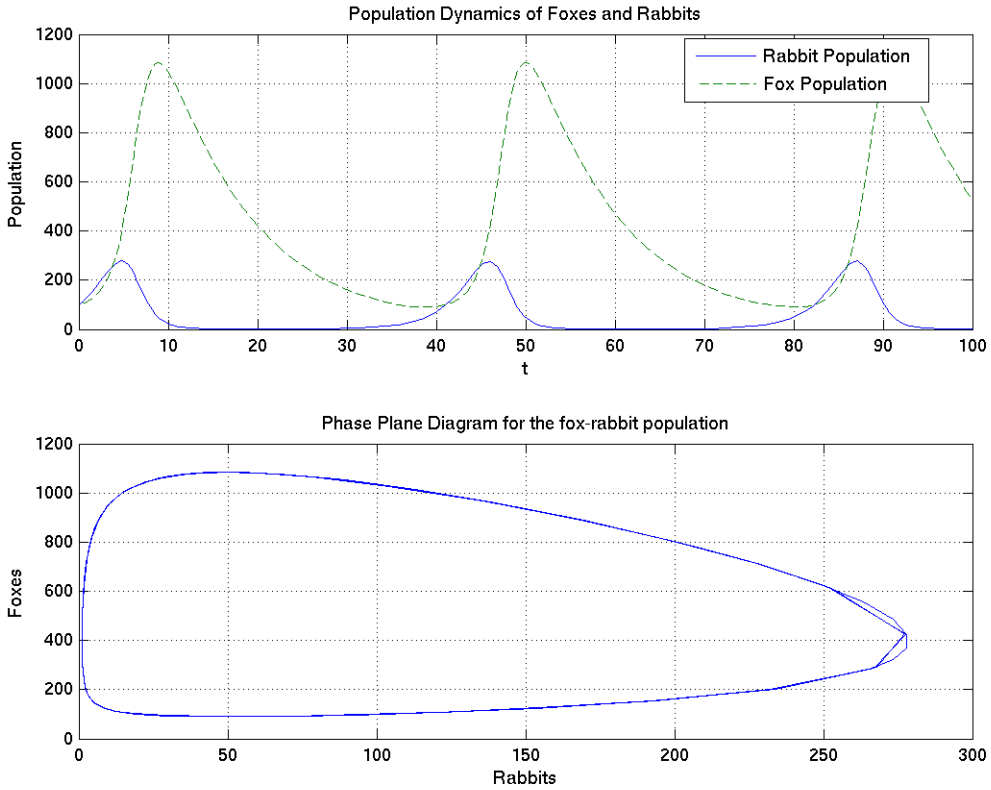

Figure 2: Dynamics of a predator prey (fox/rabbit) system

The reader can further experiment with the above Matlab code to see the outcome with different parameters and different initial populations.

Our discussion of Matlab's ODE solvers here focused on the example of the function `ode45`, which is Matlab's most popular ODE solver. Matlab has a suite of solvers, see `doc ode45` for full documentation and recommendations for when to use which method in table form. We complement this table here by discussing the methods and providing some additional information. See this documentation also for the list of options used to control the method, such as relative and absolute tolerances on the error estimator, as well as for a list of references on the subject of ODE solvers and the methods in particular.

All ODE solvers in Matlab use the same function interface, so it is easy to try several methods on the same problem and observe their behavior. Also, all methods compute an estimator for the error of the solution that is used to automatically select the size of the time step and also of the method order, if it is variable. ODE problems are roughly classified into *stiff* and *nonstiff* problems. General-purpose ODE solvers in Matlab that are appropriate for stiff problems are indicated by the letter "s" at the end of their names, namely `ode15s` and `ode23s`. The most important nonstiff solvers are `45` and `ode113`. The numbers in the names of the two methods `ode15s` and `ode113` that are variable-order methods indicate the method order ranging from 1 to 5 and from 1 to 13, respectively. All other methods are fixed-order methods with the first number indicating the order of the method, such as 4 in `ode45` and 2 in `ode23s`; the second number indicates the order of the second method used simultaneously in the error estimator.

The technical definition of the term *stiff* is difficult, but their practical definition is readily stated: A problem is stiff, if the automatic step size control of the method chooses small steps even for large tolerances. This means that the step sizes are limited by the stability of the method and not the accuracy requested by the user. Note that all ODE solvers in Matlab use automatic step size control based on a sophisticated error estimator, hence the accuracy of their solution is ensured; but if it takes a longer time to compute the solution with nonstiff solvers such as `ode45` or `ode113` than with a stiff solver such as `ode15s`, the problem is considered stiff. In summary, for a particular problem, try `ode45` or `ode113` as potential nonstiff solvers and try also `ode15s` as potential stiff solvers. Then continue using whichever method performed most efficiently.

2 Matlab Toolboxes

In this section, we will discuss Matlab Toolboxes. In general, Matlab toolboxes extend the capabilities of Matlab by providing highly efficient routines which are specialized to handle specific situations. For example, if one is solving some problems in the area of neural networks, the Neural Network Toolbox provides powerful tools to handle problems of that type. As another example, Matlab's Statistics Toolbox provides a wide range of statistical routines.

A good way to learn about a Matlab Toolbox is studying the associated Getting Started Guide; another good place to start is the user's guide for the associated Toolbox. For example, in Figure 3, we have displayed the help screen for Matlab's Statistics Toolbox (from Matlab's Help facility). We can see the various documentations provided for a toolbox.

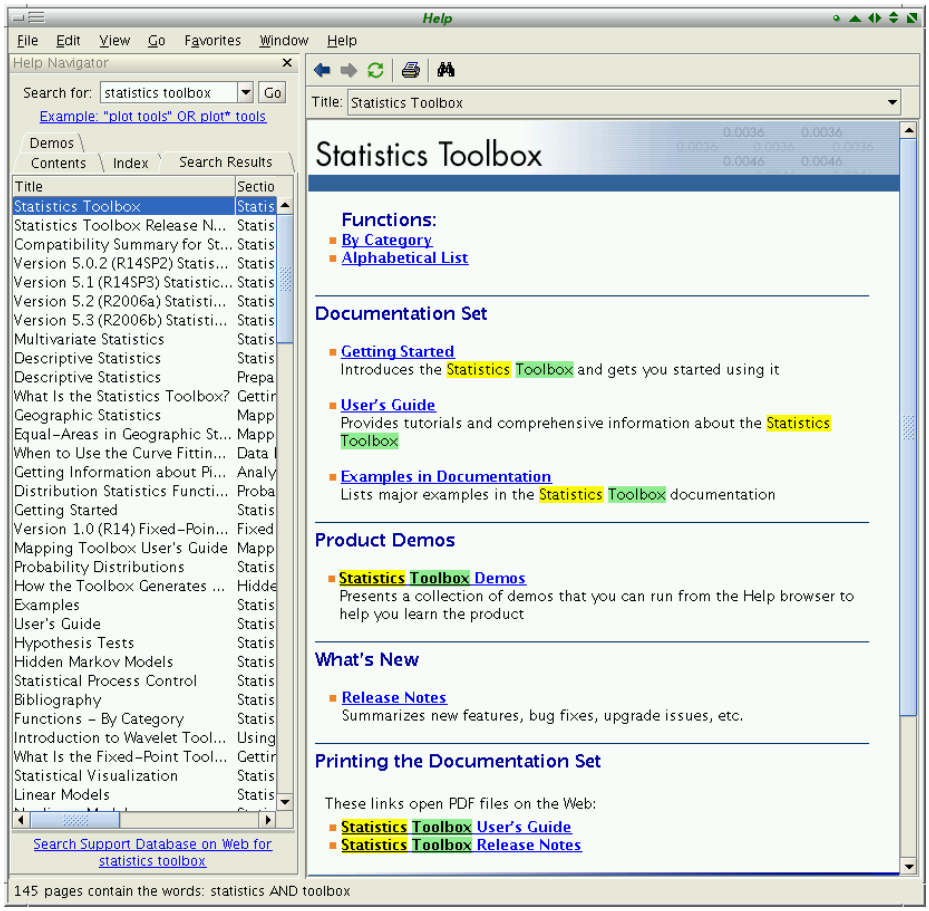

Figure 3: Matlab's Statistics Toolbox

To give the readers an idea of the available Matlab toolboxes, a list of widely used Matlab Toolboxes is provided in Table 1.

To find out which Toolboxes are available in your version of Matlab you can type

Math and Optimization
Optimization Toolbox
Symbolic Math Toolbox
Extended Symbolic Math Toolbox
Partial Differential Equation Toolbox
Genetic Algorithm and Direct Search Toolbox
Statistics and Data Analysis
Statistics Toolbox
Neural Network Toolbox
Curve Fitting Toolbox
Spline Toolbox
Model-Based Calibration Toolbox
Control System Design and Analysis
Control System Toolbox
System Identification Toolbox
Fuzzy Logic Toolbox
Robust Control Toolbox
Model Predictive Control Toolbox
Aerospace Toolbox
Signal Processing and Communications
Signal Processing Toolbox
Communications Toolbox
Filter Design Toolbox
Filter Design HDL Coder Wavelet Toolbox
Fixed-Point Toolbox RF Toolbox
Image Processing
Image Processing Toolbox
Image Acquisition Toolbox
Mapping Toolbox
Test and Measurement
Data Acquisition Toolbox
Instrument Control Toolbox
Image Acquisition Toolbox
SystemTest OPC Toolbox
Computational Biology
Bioinformatics Toolbox
Financial Modeling and Analysis
Financial Toolbox
Financial Derivatives Toolbox
GARCH Toolbox
Datafeed Toolbox
Fixed-Income Toolbox

Table 1: A list of Matlab Toolboxes

```
ver
```

in Matlab's command prompt. Issuing the `ver` command will provide something like the following:

```
>> ver
-------------------------------------------------------------------------------
MATLAB Version 7.3.0.298 (R2006b)
MATLAB License Number: 334413
Operating System: Linux 2.4.31-EL3SMPMath-iptables #1 SMP Thu Jun 2 21:02:32 EDT 2005 i686
Java VM Version: Java 1.5.0 with Sun Microsystems Inc. Java HotSpot(TM)
Client VM mixed mode, sharing
-------------------------------------------------------------------------------
MATLAB                                                Version 7.3        (R2006b)
Simulink                                              Version 6.5        (R2006b)
Communications Blockset                               Version 3.4        (R2006b)
Communications Toolbox                                Version 3.4        (R2006b)
Control System Toolbox                                Version 7.1        (R2006b)
Image Processing Toolbox                              Version 5.3        (R2006b)
Instrument Control Toolbox                            Version 2.4.1      (R2006b)
MATLAB Compiler                                       Version 4.5        (R2006b)
Mapping Toolbox                                       Version 2.4        (R2006b)
Neural Network Toolbox                                Version 5.0.1      (R2006b)
Optimization Toolbox                                  Version 3.1        (R2006b)
Partial Differential Equation Toolbox                 Version 1.0.9      (R2006b)
Real-Time Workshop                                    Version 6.5        (R2006b)
Robust Control Toolbox                                Version 3.1.1      (R2006b)
Signal Processing Blockset                            Version 6.4        (R2006b)
Signal Processing Toolbox                             Version 6.6        (R2006b)
Simulink Control Design                               Version 2.0.1      (R2006b)
Spline Toolbox                                        Version 3.3.1      (R2006b)
Statistics Toolbox                                    Version 5.3        (R2006b)
Symbolic Math Toolbox                                 Version 3.1.5      (R2006b)
System Identification Toolbox                         Version 6.2        (R2006b)
Wavelet Toolbox                                       Version 3.1        (R2006b)

Trademarks
------------------
MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC
TargetBox are registered trademarks of The MathWorks, Inc. Other product or
brand names are trademarks or registered trademarks of their respective holders.
```

Of course, the results may vary depending on the system on which the command `ver` is issued.