Matlab Notes Part 3

Austin Baird
UNC Department of Mathematics
UNC Department of Biology

February 4, 2014

Summary

- Last week we learned:
    - For loops and control sequences.
    - Functions: anonymous and traditional functions.
    - Using these functions.
    - writing our first function
- Today we will:
    - Learn how to use built in Matlab functions.
    - Systems of equations and how they relate to matrix functions.
    - Understanding the output.
    - Integrating them into functions.

Goal of Today

We want to be able to analyze a system of ODE’s numerically and produce an output that makes sense using matlab built in functions to solve systems of ODEs:

$$
\begin{aligned}
\frac{dy_1}{dt} &= f(y_1, y_2, ..., t) \\
\frac{dy_2}{dt} &= f(y_1, y_2, ..., t) \\
\frac{dy_3}{dt} &= f(y_1, y_2, ..., t) \\
&... \\
\frac{dy_n}{dt} &= f(y_1, y_2, ..., t)
\end{aligned}
$$

Systems of ODEs

Note that this is equivalent to writing the system in the form:

$$\frac{d\hat{y}}{dt} = F(\hat{y}, t)$$

Which in the end is a matrix equation. We will discuss this at length now!

Systems of Equations

Because Matlab is a matrix manipulator we want to see the connection between a system of equations and writing it as a matrix system (Ax=b).

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n &= y_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n &= y_2 \\ ... \\ a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n &= y_m \end{aligned}$$

Systems of Equations

The system of equations on the previous page can now be written as:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can now solve this by solving: $A_{m,n}X_n = Y_m$

Solving the System

There are many ways to solve this system! The most obvious way is using the power of Matlab!

$$\gg X = inv(A)Y$$

Done! Or we can do!

$$\gg X = A\backslash Y$$

Check to make sure these do the same thing!

Functions in Matlab

Solving Systems of ODEs with Matlab

Now that we've seen how to simply solve systems of equations, lets solve systems of ODEs. Usually this can be done using built in solvers (We will go over the algorithms in the future):

$\gg$ $[t_{out}, y_{out}]$ = ode**(@F,time,y_0,options]

Generally we want to solve $y' = f(y,t)$ over a given time domain.

- Here t_{out} and time are arrays of time values,
- @F is $f(y,t)$, which can be an array if there is a system of ODEs (example code)
- y_0 are the initial conditions .
- y_{out} is an array of values which correspond to the solution at each *dt* (discretized time domain).

Let's Try it

First we will try and solve the ODE: $\frac{dy}{dx} = xy^2 + y$, to do this we will define a function:

F= @(x,y) xy^2 + y

We can then solve it! [x,y] = ode45(f,[0,0.5],1)
Then plot the output:
plot(x,y)

Note: If you allow the domain of this function to reach 1, then the error will explode! (why?)

Solving a system

To solve a system of ODEs we can store them in an array with a separate function, then call that function when using ode45.

function f = odefun(t,y)

Define your parameters here...

We now want to store the functions in an array:

f(1) = y(1)+y(2)
f(2) = y(1)+y(2)

or...

f = [y(1) + y(2) ; y(1) + y(2)], Note: y(1) = x and y(2) = y. If you had a z, y(3) = z.

Finishing it all up!

Now that we have a function which stores the functions corresponding to our system of odes we call our function:

initial conditions = array corresponding to [y(1) y(2) ...]

time values = array corresponding to [time(start) time(end)]

now we call the function: [t,y] = ode45(@f,time,initial)

Now the solutions are stored as the columns of y. They can be called by: y(:,1), y(:,2)...

Homework 3

Lorenz equations were initially developed as a weather model and mathematically have served as an example of chaotic behavior seen in non-linear systems. The Lorenz equations are defined to be:

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x)\\ \frac{dy}{dt} &= x(\rho - z) - y\\ \frac{dz}{dt} &= xy - \beta z\end{aligned}$$

Note: σ, ρ, β are constants and can be defined however you want them to. (i.e make them all 10)
I want you to solve these equations using ode45 and to create a few plots:

Homework 3

items to do:

- Alter the functions I gave you so that they represent the lorenz equations.
- Make a script which plots the strange attractor: plot(x,z) on the same graph.
- Have this script also plot x, y, z, over the time domain, in the same figure (using subplots)
- Have it make a graph of x for one set of initial conditions = $[x_0 y_0 z_0]$ Then on the same figure a plot with these initial conditions: $[(x_0 + 0.1)(y_0 + 0.1)(z_0 + 0.1)]$.
  Note: Be sure to keep all parameter values greater than zero, other than that you can choose them as you like. Some are more interesting than others...