Using Mathematica to Solve Differential Equations

John Douglas Moore

February 1, 2010

In solving differential equations, it is sometimes necessary to do calculations which would be prohibitively difficult to do by hand. Fortunately, computers can do the calculations for us, if they are equiped with suitable software, such as Matlab or Mathematica. Here we give a brief introduction to the use of Mathematica for doing such calculations.

Most computers which are set up to use Mathematica contain an on-line tutorial, "Tour of Mathematica," which can be used to give a brief hands-on introduction to Mathematica. Using this tour, it should be possible to learn enough Mathematica to get started without referring to lengthy technical manuals.

Our purpose here is to give a very brief introduction to the use of Mathematica. After launching Mathematica, you can use it as a "more powerful than usual graphing calculator." For example, if you type in

```
(11 - 5)/3
```

the computer will perform the subtraction and division, and respond with

```
Out[1] = 2
```

The notation for multiplication is $*$, so if you type in

```
2 * (7 + 4)
```

the computer will respond with

```
Out[2] = 22
```

You can also use a space instead of $*$ to denote multiplication, so if you input

```
2 8
```

the computer will respond with

```
Out[3] = 16
```

The computer can do exact arithmetic with integers or rational numbers. For example, since $\wedge$ is the symbol used for raising to a power, if you type in

```
2∧150
```

the computer will calculate the 150th power of two and give the exact result:

```
Out[4] = 1427247692705959881058285969449495136382746624
```

On the other hand, the symbol `N` tells the computer to use an approximation with a fixed number of digits of precision, so entering

```
N[2∧150]
```

will give an approximation to the exact answer, expressed in scientific notation:

```
Out[5] = 1.42725 10^45
```

Real numbers which are not rational, such as π, have infinite decimal expansions which cannot be stored within a computer. However, we can approximate a real number by a finite decimal expansion with a given number of digits of precision. For example, since Mathematica uses the name `E` to denote the number e, typing in

```
N[E]
```

will tell Mathematica to give a rational approximation to e to a standard number of digits of precision:

```
Out[6] = 2.71828
```

In principle, the computer can calculate rational approximations to an arbitrary number of digits of precision. Since the number π is represented by Mathematica as `Pi`, typing in

```
N[Pi,40]
```

will tell Mathematica to give a rational approximation to π to 40 digits of precision:

```
Out[7] = 3.141592653589793238462643383279502884197 2
```

The computer can represent not only numbers but functions. Within Mathematica, built-in functions are described by names which begin with capital letters. For example, `Log` denotes the natural base e logarithm function. Thus entering

```
N[Log[2],40]
```

will give the logarithm of 2 with 40 digits of precision:

```
Out[8] = 0.693147180559945309417232121458176568076
```

Of course, you are familiar with using .7 as a very rough approximation for the logarithm of 2. It is this fact which enables you to state without hesitation that a population which is growing at 1% per year will double in $.7/.01 = 70$ years.

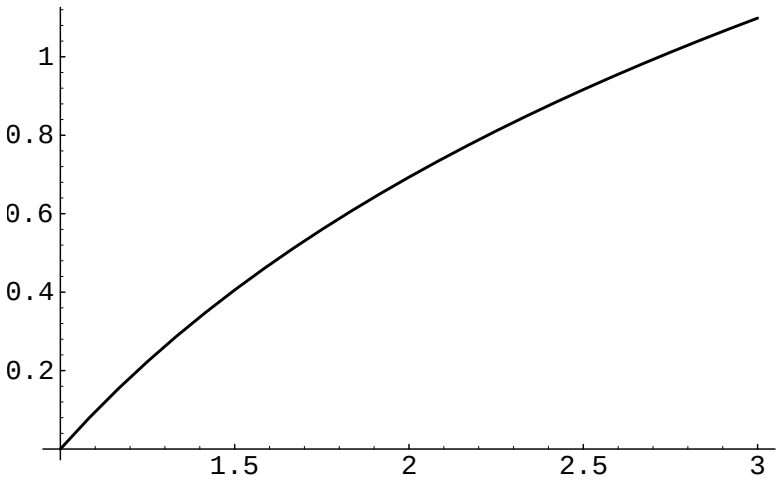

Figure 1: Graph of the logarithm function.

One can also plot functions with Mathematica. For example, to plot the logarithm function from the values 1 to 3, one simply inputs

```
Plot[Log[t],{t,1,3}]
```

and Mathematica will automatically produce the plot shown in Figure 1.

We can also define functions ourselves, being careful not to capitalize them, because capitals are reserved by Mathematica for built-in functions. Thus we can define the function

$$y(t) = ce^{kt}$$

by typing

```
y[t_] := c * E^(k * t);
```

Mathematica will remember the function we have defined until we quit Mathematica. We must remember the exact syntax of this example (use of the underline character and the colon followed by the equal sign) when defining functions. In this example c and k are *parameters* which can be defined later. Just as in the case of functions, we must be careful to represent parameters by lower case letters, so they will not be confused with built-in constants. Further entry of

```
c = 1; k = 1; N[y[1]]
```

yields the response

```
Out[11] = 2.71828
```

while entering

```
Plot[y[t],{t,0,2}]
```

will give a plot of the function we have defined, for t in the interval $[0, 2]$.

We can use Mathematica to solve matrix differential equations of the form

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x}, \tag{1}$$

where A is a square matrix with constant entries.

The first step consists of using Mathematica to find the eigenvalues and eigenvectors of A. To see how this works, we must first become familiar with the way in which Mathematica represents matrices. Since Mathematica reserves upper case letters for descriptions of built-in functions, it is prudent to denote the matrix A by lower case a when writing statements in Mathematica. The matrix

$$A = \begin{pmatrix} -2 & 5 \\ 1 & -3 \end{pmatrix}$$

can be entered into Mathematica as a collection of row vectors,

```
a = {{-2,5},{1,-3}}
```

with the computer responding by

```
Out[1] = {{-2,5},{1,-3}}
```

Thus a matrix is thought of as a "vector of vectors." Entering

```
MatrixForm[a]
```

will cause the computer to give the matrix in the familiar form

```
Out[2] =
          ( -2   5  )
          (  1  -3  )
```

To find the eigenvalues of the matrix A, we simply type

```
Eigenvalues[a]
```

and the computer will give us the exact eigenvalues

$$\frac{-5 - \sqrt{21}}{2}, \qquad \frac{-5 + \sqrt{21}}{2},$$

which have been obtained by using the quadratic formula. Quite often numerical approximations are sufficient, and these can be obtained by typing

```
eval = Eigenvalues[N[a]]
```

the response this time being

```
Out[4] = {-4.79129, -0.208712}
```

Defining `eval` to be the eigenvalues of A in this fashion, allows us to refer to the eigenvalues of A later by means of the expression `eval`.

We can also find the corresponding eigenvectors for the matrix A by typing

```
evec = Eigenvectors[N[a]]
```

and the computer responds with

```
Out[5] = {{-0.873154, 0.487445}, {0.941409, 0.337267}}
```

Putting this together with the eigenvalues gives the general solution to the original linear system (1) for our choice of the matrix A:

$$\mathbf{x}(t) = c_1 \begin{pmatrix} -0.873154 \\ 0.487445 \end{pmatrix} e^{-4.79129t} + c_2 \begin{pmatrix} 0.941409 \\ 0.337267 \end{pmatrix} e^{-0.208712t}.$$

Mathematica can also be used to find numerical solutions to nonlinear differential equations. The following Mathematica programs will use Mathematica's differential equation solver (which is called up by the command `NDSolve`), to find a numerical solution to the initial value problem

$$dy/dx = y(1-y), \qquad y(0) = .1$$

for the logistic equation, give a table of values for the solution, and graph the solution curve on the interval $0 \le x \le 6$. The first step

```
sol := NDSolve[{ y'[x] == y[x] (1 - y[x]), y[0] == .1 },
y, {x,0,6}]
```

generates an "interpolation function" which approximates the solution and calls it sol, an abbreviation for solution. We can construct a table of values for the interpolation function by typing

```
Table[Evaluate[y[x] /.  sol], {x,0,6,.1}];
```

or graph the interpolation function by typing

```
Plot[Evaluate[y[x] /.  sol], {x,0,6}]
```

This leads to a plot like that shown in Figure 2.

Readers can modify these simple programs to graph solutions to initial value problems for quite general differential equations of the canonical form

$$\frac{dy}{dx} = f(x, y).$$

All that is needed is to replace the first argument of `NDSolve` with the differential equation one wants to solve, remembering to replace the equal signs with double equal signs, as in the example.

In fact, it is no more difficult to treat initial value problems for higher order equations or systems of differential equations. For example, to solve the initial value problem

$$\frac{dx}{dt} = -xy, \quad \frac{dy}{dt} = xy - y, \quad x(0) = 2, \quad y(0) = .1. \tag{2}$$

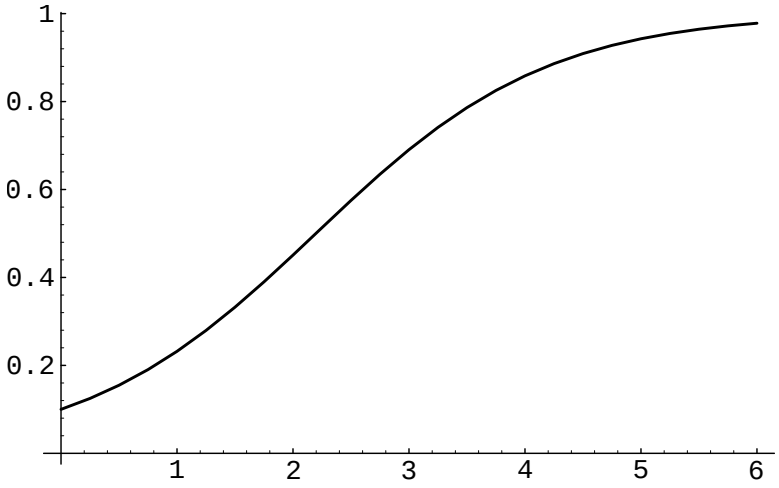

Figure 2: Solution to $y' = y(1-y)$, $y(0) = .1$.

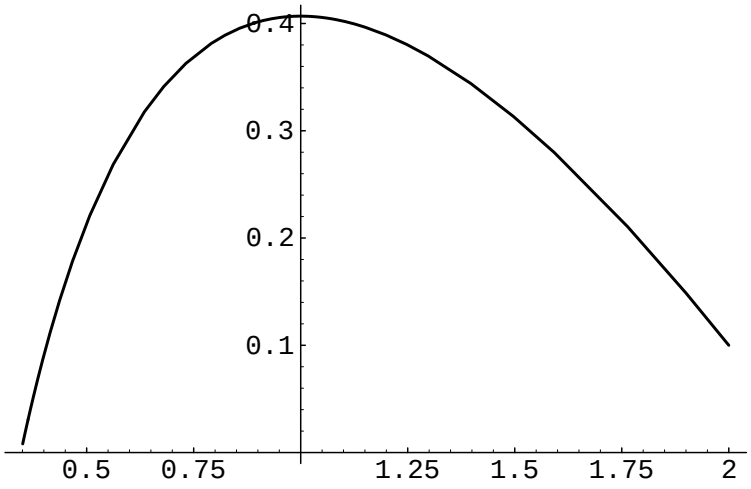

Figure 3: A parametric plot of a solution to $dx/dt = -xy$, $dy/dt = xy - y$.

one simply types in

```
sol := NDSolve[{ x'[t] == - x[t] y[t], y'[t] == x[t] y[t] - y[t],
x[0] == 2, y[0] == .1 },
{x,y}, {t,0,10}]
```

Once we have this solution, we can print out a table of values for y by entering

```
Table[Evaluate[y[t] /.  sol], {t,0,2,.1}]
```

We can also plot $y(t)$ as a function of t by entering

```
Plot[Evaluate[y[t] /.  sol], {t,0,10}]
```

Figure 3 shows a parametric plot of $(x(t), y(t))$ which is generated by entering

```
ParametricPlot[Evaluate[{ x[t], y[t]} /.  sol], {t,0,10}]
```