MATH 100 – Introduction to the Profession

Differential Equations in MATLAB

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2012

What is a Differential Equation?

Answer

Unlike an algebraic equation such as

$$2x = x^2 + 1 \quad \Leftrightarrow \quad x^2 - 2x + 1 = 0 \quad \Leftrightarrow \quad (x-1)^2 = 0 \quad \Leftrightarrow \quad x = 1,$$

or a transcendental equation such as

$$x = \cos(x) \quad \overset{\text{fixed point iteration}}{\Longrightarrow} \quad x = 0.739085133215160641655\,3\ldots,$$

where the unknown quantities are given by a number, x, and algebraic or transcendental expressions in terms of x, the unknown quantity in a differential equation is a function along with some of its derivatives, and therefore the solution is also a function. For example, we saw earlier that

$$P'(t) = rP(t),\ P(0) = P_0 \quad \Longleftrightarrow \quad P(t) = P_0 e^{rt},\ t > 0.$$

Since derivatives can be interpreted as rates of change, differential equations are used to express problems such as

- change in account balance in relation to the amount of money in an account (as a function of time)

$$A'(t) = rA(t) \quad \Longrightarrow \quad A(t) = \dots$$

- population growth rate in relation to the size of a population (as a function of time)

$$P'(t) = \left(1 - \frac{P(t)}{C}\right) P(t) \quad \Longrightarrow \quad P(t) = \dots$$

- acceleration of a body in relation to the position of the body (as a function of time)

$$x''(t) = -x(t) \quad \Longrightarrow \quad x(t) = \dots$$

- change in temperature of a body (as a function of location and time)

$$\frac{\partial T(x,t)}{\partial t} = -k^2 \frac{\partial^2 T(x,t)}{\partial x^2} \quad \Longrightarrow \quad T(x,t) = \dots$$

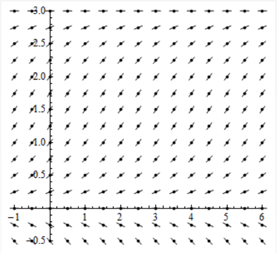
Visualizing Differential Equations

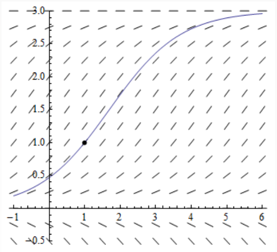
Slope fields are a great way to visualize what's going on in a differential equation.

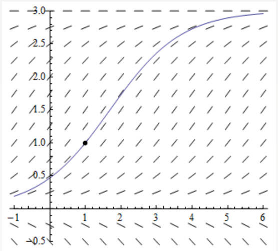
A differential equation by itself does not constitute a well-posed problem. There are graphs of many different functions that fit into any given slope field.

A slope field usually represents a family of infinitely many solutions to the differential equation.

In order to guarantee a unique solution (and make the problem well-posed) we need to specify an initial condition (or initial point).







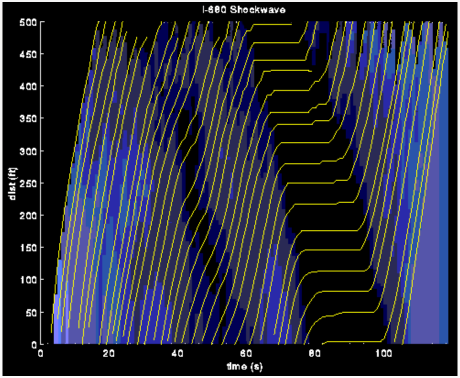
We can use the Mathematica Demo `SlopeFieldsEdited.cdf` to illustrate, e.g.,

$$y' = 3y \qquad \text{(i.e., } A'(t) = rA(t)\text{)}$$

$$y' = \left(1 - \frac{y}{3}\right) y \qquad \text{(i.e., } P'(t) = \left(1 - \frac{P(t)}{C}\right) P(t)\text{)}$$

Other examples of “real life” slope fields are

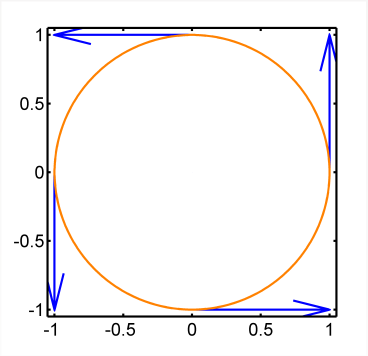
- grade markers along a road [ExM]
- speed sensors along a highway (group project)
- complex flow visualization



Shock wave on I-680 in Walnut Creek, CA, during rush hour [Coifman]. The graph shows distance traveled (vertical axis) as a function of time (horizontal axis).
Each yellow line describes the path of an individual vehicle. Line slopes correspond to vehicle speed at any position and time (background color ranges from black=0 mph to blue=40 mph).

What should a slope field for a circular orbit look like?

Point $(x(t), y(t))$	Tangent $(x'(t), y'(t))$
(1,0)	(0,1)
(0,1)	(-1,0)
(-1,0)	(0,-1)
(0,-1)	(1,0)



Differential equations:

$$\begin{aligned} x'(t) &= -y(t) \\ y'(t) &= x(t) \end{aligned}$$

Which functions $x = x(t)$ and $y = y(t)$ satisfy this system of differential equations? $x(t) = \cos(t),\ y(t) = \sin(t)$

Remark

In more realistic applications these orbits are derived based on physical laws (such as Kepler's or Newton's laws of motion).

Standard form of first-order ODE systems

Instead of

$$x'(t) = -y(t)$$
$$y'(t) = x(t)$$

one usually changes over to vector notation. We introduce $\boldsymbol{y}(t) = [y_1(t), y_2(t)]^T = [x(t), y(t)]^T$, so that the circle orbit is described by

$$y_1'(t) = -y_2(t)$$
$$y_2'(t) = y_1(t)\,.$$

If we abbreviate $\boldsymbol{f}(t, \boldsymbol{y}(t)) = [-y_2(t), y_1(t)]^T$, then we get

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t))\,.$$

This is the standard form of a (system of) ordinary differential equations, and also the way MATLAB expects to get an ODE.

Orbits via `ode23`

The system

$$\begin{aligned} y_1'(t) &= -y_2(t) \\ y_2'(t) &= y_1(t) \end{aligned}$$

with $\boldsymbol{f}(t, \boldsymbol{y}(t)) = [-y_2(t), y_1(t)]^T$ can be implemented in MATLAB:

```
mycircle = @(t,y) [-y(2); y(1)];   % f(t,y(t))
tspan = [0 2*pi];   % range for t
y0 = [1; 0];   % starting point
[t y] = ode23(mycircle,tspan,y0);
plot(y(:,1),y(:,2),'-o')
axis(1.1*[-1 1 -1 1])
axis square
```

An alternate form using a MATLAB function M-file is described in [ExM].

What does the following change[1] do?

$$\begin{aligned} y_1'(t) &= -y_2(t) \\ y_2'(t) &= y_1(t) \end{aligned} \quad \longrightarrow \quad \begin{aligned} y_1'(t) &= y_2(t) \\ y_2'(t) &= -y_1(t) \end{aligned}$$

- We have $\boldsymbol{y}' \longrightarrow -\boldsymbol{y}'$.
- The orientation of the orbit is reversed (counterclockwise $\longrightarrow$ clockwise).
- The solution is given by

$$\begin{aligned} y_1(t) &= \cos(t) \\ y_2(t) &= \sin(t) \end{aligned} \quad \longrightarrow \quad \begin{aligned} y_1(t) &= \sin(t) \\ y_2(t) &= \cos(t) \end{aligned}$$

- Try this out in MATLAB.

[1]This is how the circular orbit is described in [ExM], but standard mathematical positive orientation is counterclockwise.

Let's try some other problems

Example

The following should result in a spiral-shaped orbit:

$$\begin{aligned} x'(t) &= -x(t) + y(t), \qquad & x(0) &= 1 \\ y'(t) &= -x(t) - y(t), \qquad & y(0) &= 1 \end{aligned}$$

Fill in the blanks (xxxxx) in the MATLAB code:

```
myorbit = @(t,y) [xxxxx; xxxxx];    % f(t,y(t))
tspan = [0 10];    % range for t
y0 = [xxxxx; xxxxx];    % starting point
[t y] = ode23(myorbit,tspan,y0);
plot(y(:,1),y(:,2),'-o')
```

Example

The following should result in an attracting circular orbit:

$$\begin{aligned} x'(t) &= x(t) + y(t) - x^3(t) - x(t)y^2(t), & x(0) &= 2 \\ y'(t) &= -x(t) + y(t) - x^2(t)y(t) - y^3(t), & y(0) &= 0 \end{aligned}$$

Fill in the blanks (xxxxx) in the MATLAB code:

```
myorbit = @(t,y) [xxxxx; xxxxx];   % f(t,y(t))
tspan = [0 10];   % range for t
y0 = [xxxxx; xxxxx];   % starting point
[t y] = ode23(myorbit,tspan,y0);
plot(y(:,1),y(:,2),'-o')
```

Try different starting points!

The topics we have discussed up to now will come up in later MATH classes such as

- MATH 152 (Calculus II): polar coordinates, curves in parametric form
- MATH 251 (Multivariable Calculus): functions in vector form
- MATH 252 (Introduction to Differential Equations): (systems of) differential equations
- MATH 350 (Introduction to Computational Mathematics): numerical methods like `ode23`

Approximating Derivatives

Recall the definition of the derivative:

$$y'(t) = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h}.$$

We can easily turn this into a numerical method by dropping the limit:

$$y'(t) \approx \frac{y(t+h) - y(t)}{h},$$

a so-called forward difference approximation.

Example

While this is in general only an approximation, it is exact for linear functions.

Let $y(t) = mt + b$ so that $y'(t) = m$. Then

$$\frac{y(t+h) - y(t)}{h} = \frac{[m(t+h) + b] - [mt + b]}{h} = \frac{mh}{h} = m.$$

Circular Space Orbit Revisited

We now construct our own method to follow the circular orbit (a simple differential equation solver).
Remember the system of ODEs:

$$\begin{aligned} x'(t) &= -y(t) \\ y'(t) &= x(t) \end{aligned}$$

Using the forward difference approximation we get

$$\begin{aligned} \frac{x(t+h)-x(t)}{h} &= -y(t) \\ \frac{y(t+h)-y(t)}{h} &= x(t) \end{aligned} \iff \begin{aligned} x(t+h) &= x(t) - h\,y(t) \\ y(t+h) &= y(t) + h\,x(t) \end{aligned}$$

If we start at any point $(x(t), y(t))$, then we can get a new point $(x(t+h), y(t+h))$ near the orbit by using the above formulas with a small stepsize h.

Space War Orbit [ExM]

```
x = 1;  y = 0;    % starting point
h = 1/4;    % small stepsize
n = 2*pi/h;    % to get one full revolution
plot(x,y,'.')
hold on
for k = 1:n    % compute new points near orbit
    x = x - h*y;
    y = y + h*x;
    plot(x,y,'.')
end
hold off
axis([-1.1 1.1 -1.1 1.1])
axis square
```

This algorithm is known as Euler's method.
Repeat with $h = 1/32$ and explain.

Important questions to be investigated in later classes:

- How accurate is Euler's method? (→ MATH 350, uses Taylor series from MATH 152)
- Are there other more accurate or more efficient methods to solve ODEs? (→ MATH 350)
- How does one solve differential equations analytically? (→ MATH 152, MATH 252, MATH 461, MATH 488, MATH 489)
- Learn about other kinds of differential equations problems:
  - we looked only at initial value problems
  - there are also boundary value problems
  - differential equations involving derivatives with respect to more than one independent variables become partial differential equations
- How sensitive are differential equations to their initial conditions? (→ MATH 488, dynamical systems and chaos)

Second-order ODEs as Systems of First-order ODEs

Consider the simple mathematical model for a falling body (see, e.g., [Gowers, Ch. 1]):

$$x''(t) = -g \quad \overset{\text{integrate}}{\Longrightarrow} \quad x'(t) = -gt + v_0 \quad \overset{\text{integrate}}{\Longrightarrow} \quad x(t) = -\frac{gt^2}{2} + v_0 t + x_0.$$

Let's pretend we can't integrate this equation, and use MATLAB instead.

We need to convert this second-order ODE to a system of first-order ODEs (since that's all that MATLAB understands, and since this corresponds to standard form).

If we introduce a second unknown function, $v(t) = x'(t)$, then we have

$$\begin{aligned} x'(t) &= v(t) \\ v'(t) &= -g \end{aligned}$$

Solving the Falling Rock Problem

From the previous slide we know that the system

$$x'(t) = v(t)$$
$$v'(t) = -g$$

has solution $x(t) = -\frac{gt^2}{2} + v_0 t + x_0$, where x_0 and v_0 are the initial height and initial velocity, respectively.

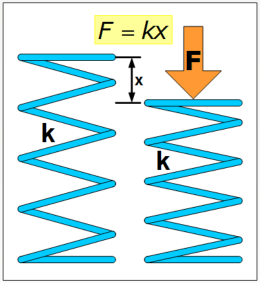
```
myrock = @(t,y) [y(2); -9.81];    % f(t,y(t))
tspan = [0 2];    % range for t
y0 = [20; 0];    % starts at height 20, 0 velocity
[t y] = ode23(myrock,tspan,y0);
plot(t,y(:,1),'-o')   % MATLAB's solution
hold on   % analytical solution from integration
tt = linspace(0,2,100);
plot(tt, -9.81*tt.^2/2 + y0(2)*tt + y0(1),'r')
hold off
```

A Simple Spring-Mass Model

Hooke's law states that

$$F = -kx,$$

i.e., the force required to restore a spring from a displacement of x units out of equilibrium is proportional (with spring constant k) to the displacement.



Since, from Newton's second law of motion, we also know that $F = ma$ (mass $\times$ acceleration), and $a(t) = x''(t)$, we get the following basic mathematical model for a spring-mass system[2]:

$$x''(t) = -x(t).$$

[2]For simplicity we set $m = k = 1$

We actually can't solve

$$x''(t) = -x(t).$$

by simple integration, so we again use MATLAB (in MATH 252 you learn how to do this analytically as well).
Introducing $v(t) = x'(t)$ as a second unknown function we get

$$\begin{aligned} x'(t) &= v(t) \\ v'(t) &= -x(t) \end{aligned}$$

This looks suspiciously like our earlier orbit system! We have periodic motion, a so-called harmonic oscillator.

Solving the Spring Problem

We might be able to guess that

$$x''(t) = -x(t)$$

can be satisfied by $x(t) = \cos(t)$, or by $x(t) = \sin(t)$.
Again, the ODE by itself is not a well-posed problem. Additional initial conditions will give us a unique solution:

```
myspring = @(t,y) [y(2); -y(1)];   % f(t,y(t))
tspan = [0 10];   % range for t
y0 = [2; 0];   % initial displacement 2, velocity 0
[t y] = ode23(myspring,tspan,y0);
plot(t,y(:,1),'-o')  % MATLAB's solution
```

Example

If we add a damping term to the spring equations, then we get the so-called damped harmonic (van der Pol) oscillator:

$$x''(t) = \mu(1 - x(t)^2)x'(t) - x(t), \qquad x(0) = 2,\ x'(0) = 0.$$

Convert to a first-order system:

$$\begin{aligned} x'(t) &= v(t), & x(0) &= 2 \\ v'(t) &= \mu(1 - x(t)^2)v(t) - x(t), & v(0) &= 0 \end{aligned}$$

Fill in the blanks (xxxxx) in the MATLAB code. Experiment with different values of $\mu = 0.01, 0.1, 1, 10, 100$:

```
mu = xxxxx;
vanderPol = @(t,y) [xxxxx; xxxxx];    % f(t,y(t))
tspan = [0 1000];    % range for t
y0 = [xxxxx; xxxxx];    % starting point
[t y] = ode23(vanderPol,tspan,y0);
plot(y(:,1),y(:,2),'-o')
```

Stiff ODEs

The van der Pol problem becomes very stiff for large values of the damping parameter μ.

For such cases, there are special ODE solvers.

Try `ode23s` instead of `ode23` in those cases.

More about the complicated phenomenon of stiffness is discussed in MATH 350 and MATH 478.

Connection to Matrices

Linear constant coefficient homogeneous ODEs (such as those we looked at above) can be written in matrix-vector form.

Example

The system

$$\begin{aligned} y_1'(t) &= -y_2(t) \\ y_2'(t) &= y_1(t) \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} y_1'(t) &= 0y_1(t) - 1y_2(t) \\ y_2'(t) &= 1y_1(t) + 0y_2(t) \end{aligned}$$

is equivalent to

$$\boldsymbol{y}'(t) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{y}(t), \quad \text{where } \boldsymbol{y}(t) = [y_1(t), y_2(t)]^T.$$

To solve this system analytically we then need to find the eigenvalues and eigenvectors of the matrix ($\rightarrow$ MATH 252, MATH 332).

References I

T. A. Driscoll.
Learning MATLAB.
SIAM, Philadelphia, 2009.
http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot115

Gowers, Timothy.
Mathematics: A Very Short Introduction.
Oxford University Press, 2002.

D. J. Higham and N. J. Higham.
MATLAB Guide (2nd ed.).
SIAM, Philadelphia, 2005.
http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot92

C. Moler.
Numerical Computing with MATLAB.
SIAM, Philadelphia, 2004.
http://www.mathworks.com/moler/index_ncm.html

References II

C. Moler.
Experiments with MATLAB.
Free download at
http://www.mathworks.com/moler/exm/chapters.html

The MathWorks.
MATLAB 7: Getting Started Guide.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

B. Coifman.
Wave propagation on freeways.
http://www2.ece.ohio-state.edu/~coifman/shock/