

Alan Watt and Fabio Policarpo, 3D Games: Real-time Rendering and Software Technology, Addison-Wesley, 2001

Zill, D. G., A First Course in Differential Equations with Modelling Applications, Sixth Edition, Brooks/Cole Publishing Company, 1997.

David Baraff and Andrew Witkin, Physically Based Modeling: Principles and Practice, <http://www.cs.cmu.edu/~baraff/sigcourse> (originally appeared as SIGGRAPH 97 course notes)

Chris Hecker, Physics articles in Game Developer magazine, 1996-97, available at <http://www.d6.com/users/checker/dynamics.htm>

## Topic 6 Physically-based Modelling

S2-2003

Physically-based Modelling

6-2

### Newton's Laws

The fundamental equations of mechanics are Newton's Laws of Motion:

1. *Law of Inertia.* A body will remain at rest or move with constant velocity unless acted upon by an external force.
2. The net force  $\sum \mathbf{F}$  on a body with mass  $m$  is related to the acceleration  $\mathbf{a}$  by

$$\sum \mathbf{F} = m\mathbf{a}$$

The vector form may be written in its scalar component form:

$$\begin{aligned}\sum F_x &= ma_x \\ \sum F_y &= ma_y \\ \sum F_z &= ma_z\end{aligned}$$

S2-2003

Physically-based Modelling

6-3

3. If body  $A$  exerts a force  $\mathbf{F}_{BA}$  on body  $B$ , then  $B$  must exert a force  $\mathbf{F}_{AB}$  on body  $A$ . The forces are equal in magnitude and opposite in direction:

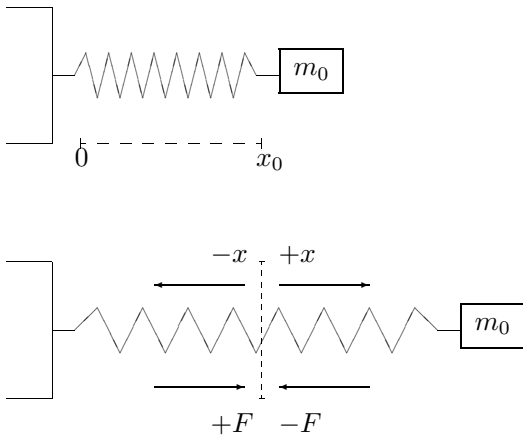
$$\mathbf{F}_{AB} = -\mathbf{F}_{BA}$$

S2-2003

# Spring Motion

The equations describing *spring motion* can be derived from Newton's Laws of Motion.

Consider the simple spring as motion in 1D



The following points can be observed for a simple spring.

- The more a spring is compressed or extended, the greater the resistance it applies to the force compressing or extending it.
- When an *ideal* spring is compressed or extended and then released, it tends to oscillate.
- When a *real* spring is compressed or extended and then released, it tends to return to its rest length (eventually).

From these, we can conclude that for an ideal spring, the force  $F$ , is proportional to the position  $x$ . It can be expressed as

$$F(t) = -kx(t), \quad k = \text{the 'spring constant'} \quad (1)$$

Applying Newton's second law,  $F = ma$ , we get

$$F(t) = ma = m \frac{d^2x}{dt^2} = -kx(t) \quad (2)$$

This is an example of a *differential equation*.

S2-2003

S2-2003

## Spring Motion: Differential Equation

A *differential equation* is an equation containing the derivatives on one or more dependent variables, with respect to one or more independent variables.

Spring motion can be written as a differential equation:

$$a = \frac{d^2x}{dt^2} = x'' = \ddot{x} = \frac{-kx}{m}$$

or

$$\frac{d^2x}{dt^2} + \frac{kx(t)}{m} = 0$$

The above differential equation is known as a *second-order, ordinary* differential equation. It is second-order as the highest order derivative is 2. It is ordinary as it does not involve partial derivatives (whereupon it would become a *partial* differential equation).

Ordinary differential equations have the acronym *ODE*.

S2-2003

## Solving ODEs

Differential equations may in some cases be *solved*. This means finding a *function* which when substituted into a differential equation gives an identity equation.

For ideal springs, such a function is  $\omega^2 = k/m$ , so that we have

$$\frac{d^2x}{dt^2} + \omega^2 x(t) = 0$$

and given that  $F_0 = 0$ , a general solution would be

$$x = A \cos \omega t + B \sin \omega t$$

or

$$x = C \cos(\omega t - D),$$

where  $C = \sqrt{A^2 + B^2}$ ,  $D = \tan^{-1} B/A$ , which are the equations of circular motion.

S2-2003

In many cases it is difficult or impossible to solve an ODE. And even when it is possible it may be preferable to use a *numerical method* to *approximately* solve an ODE. This is particularly true in computer animation.

A solution of an ODE is a set of points in the cartesian plane or in cartesian space (or higher dimensions in some cases).

The basis of numerical methods for solving ODEs is to take discrete steps in the independent variable (typically time for animations) and then work out *approximately* the value of the dependent variable (say, position).

The equation

$$y = c$$

defines a set or family of horizontal lines.

More generally, an equation of the form

$$f(x, y) = c$$

defines a family of curves, all with the same shape.

Likewise, a differential equation of the form

$$\frac{dy}{dx} = c$$

defines a family of curves. In this case, for a given value of  $c$  the *first-derivative*, i.e., the *gradient*, of the curves is constant.

Again, more generally, a differential equation of the general form

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

defines a *direction field*. For a given value  $f(x, y) = c$  a curve, or *isocline*, is defined along which the gradient is a constant. By plotting short lines along the isoclines the direction field can be drawn.

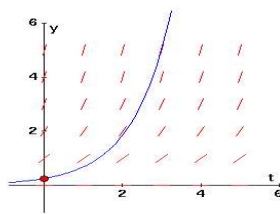


Figure 1: Direction field for  $y' = y$

The solution to this case, is  $y = y_0 e^x$ .

$$\begin{matrix} y & = & 0 & 1 & 2 \\ y' & = & 0 & 1 & 2 \end{matrix}$$

Notice that the gradient does not depend on  $x$ . This is an example of an *Initial Value Problem* (IVP).

Each of the curves  $\frac{dy}{dx} = f(x, y) = c$  has the property that the tangents to that curve are parallel.

A direction field can thus be thought of as defining all solutions to an ODE. A particular solution can be found by giving a point and by moving from that point through the direction field guided by the direction lines.

You start at an initial point and follow the direction field. Even small deviations in initial conditions can result in large differences later on, as is shown below.

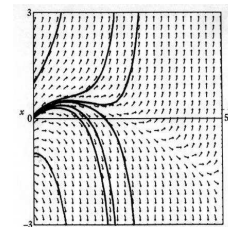


Figure 2: from p. 6 of Acheson

# Euler's Method

Euler's method is one of the simplest numerical methods for solving differential equations. It is an iterative technique which calculates an approximate value for  $y$  for  $x = x_0, x_0 + h, x_0 + 2h, \dots$

To solve the initial value problem (IVP)

$$y' = f(x, y), y(x_0) = y_0$$

or if time is the independent variable

$$y' = f(x, t), x(t_0) = x_0$$

Euler's method uses the gradient  $y'(x_0, y_0)$  at the point  $(x_0, y_0)$  to compute an approximate value for the solution curve at  $x_0 + h$  by using the *linearization*

$$L(x) = y'(x_0)(x - x_0) + y_0$$

The graph of the linearization is a straight line tangent to the graph of  $y = y(x)$  at the point  $(x_0, y_0)$ .

Let  $h$  be a positive increment on the  $x$ -axis. Then replacing  $x$  by  $x_0 + h$  in the linearization we get

$$L(x_1) = y'(x_0)(x_0 + h - x_0) + y_0 = y_0 + hy'_0$$

or

$$y_1 = y_0 + hf(x_0, y_0)$$

where  $y'_0 = y'(x_0) = f(x_0, y_0)$  and  $y_1 = L(x_1)$ .

The point  $(x_1, y_1)$  is a *local linear approximation* to the point  $(x_1, y(x_1))$  on the solution curve. That is,  $L(x_1) \approx y(x_1)$ . The accuracy of the solution is dependent on the step size  $h$ .

---

S2-2003

---

S2-2003

---

Physically-based Modelling

6-14

The process may be repeated to get an approximation  $(x_3, y_3)$  for  $(x_0 + 2h, y(x_0 + 2h))$  where two increments  $h$  have been taken. And so on ...

In general we get

$$y_{n+1} = y_n + hf(x_n, y_n)$$

---

Physically-based Modelling

6-15

## Example: Euler's Method Applied

In a general first-order IVP we have

$$y' = f(x, y), y(x_0) = y_0$$

Example:

$$y' = 0.2xy, y(1) = 1$$

Here  $f(x, y) = 0.2xy$ .

So

$$y_{n+1} = y_n + h(0.2x_n y_n)$$

The following table shows the results for  $h = 0.1$ . Decreasing the stepsize  $h$  increases the accuracy. The following table shows the results for  $h = 0.05$ .

| $x_n$ | $y_n$  | True.Value | Abs.Error | %Rel.error |
|-------|--------|------------|-----------|------------|
| 1.00  | 1.0000 | 1.0000     | 0.0000    | 0.00       |
| 1.05  | 1.0100 | 1.0103     | 0.0003    | 0.03       |
| 1.10  | 1.0206 | 1.0212     | 0.0016    | 0.06       |
| 1.15  | 1.0316 | 1.0328     | 0.0019    | 0.09       |
| 1.20  | 1.0437 | 1.0450     | 0.0013    | 0.12       |
| 1.25  | 1.0562 | 1.0579     | 0.0016    | 0.16       |
| 1.30  | 1.0694 | 1.0714     | 0.0020    | 0.19       |
| 1.35  | 1.0833 | 1.0857     | 0.0024    | 0.22       |
| 1.40  | 1.0980 | 1.1008     | 0.0028    | 0.25       |
| 1.45  | 1.1133 | 1.1166     | 0.0032    | 0.29       |
| 1.50  | 1.1295 | 1.1331     | 0.0037    | 0.32       |

---

S2-2003

---

S2-2003

The following graph shows the true solution and a range of  $h$  values.

| $x_n$ | $y_n$  | True.Value | Abs.Error | %Rel.error |
|-------|--------|------------|-----------|------------|
| 1.00  | 1.0000 | 1.0000     | 0.0000    | 0.00       |
| 1.10  | 1.0200 | 1.0212     | 0.0012    | 0.12       |
| 1.20  | 1.0424 | 1.0450     | 0.0025    | 0.24       |
| 1.30  | 1.0575 | 1.0714     | 0.0040    | 0.37       |
| 1.40  | 1.0952 | 1.1008     | 0.0055    | 0.50       |
| 1.50  | 1.1256 | 1.1331     | 0.0055    | 0.64       |

Acheson shows this graphically in fig 4.7:

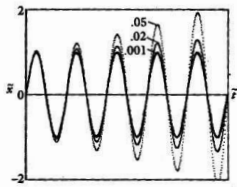


Fig. 4.7 Euler's method applied to (4.28), with  $\bar{x}=0, \bar{y}=1$  at  $\bar{t}=0$ , for three different (scaled) step sizes  $h = \bar{t}_{new} - \bar{t}_{old}$ .

Figure 3: from p. 48 of Acheson

## Euler's Method Applied to Second-Order IVPs

Forces give rise to accelerations, and accelerations are second derivatives of position with respect to time.

To apply Euler's method to a second-order IVP it is turned into two (related) first-order IVPs to each of which Euler's method is separately applied.

To obtain an approximate solution to the second-order IVP

$$y'' = f(x, y, y'), \quad y(x_0) = y_0, \quad y'(x_0) = y_1$$

we let  $y' = u$  and the problem then becomes

$$\begin{aligned} y' &= u \\ u' &= f(x, y, u) \\ y(x_0) &= y_0, \quad u(x_0) = y_1 \end{aligned}$$

These ODEs are then solved numerically by applying Euler's method to each equation

$$\begin{aligned} y_{n+1} &= y_n + hu_n \\ u_{n+1} &= u_n + hf(x_n, y_n, u_n) \end{aligned}$$

S2-2003

S2-2003

### Example: Euler's Method Applied to a Second-Order IVP

A second-order IVP:

$$y'' + xy' + y = 0, \quad y(0) = 1, \quad y'(0) = 2$$

To solve it using Euler's method let  $y' = u$ , then

$$\begin{aligned} y' &= u \\ u' &= -xu - y \end{aligned}$$

Thus we get

$$\begin{aligned} y_{n+1} &= y_n + hu_n \\ u_{n+1} &= u_n + h(-x_n u_n - y_n) \end{aligned}$$

Choosing a stepsize  $h$  of 0.1 and using the initial values of  $y_0 = 1$  and  $u_0 = 2$  we get

$$\begin{aligned} y_1 &= y_0 + (0.1)u_0 = 1 + (0.1)2 = 1.2 \\ u_1 &= u_0 + (0.1)(-x_0 u_0 - y_0) = 2 + (0.1)(-(0)(2) - 1) = 1.9 \\ y_2 &= y_1 + (0.1)u_1 = 1.2 + (0.1)(1.9) = 1.39 \\ u_2 &= u_1 + (0.1)(-x_1 u_1 - y_1) = 1.9 + (0.1)(-(0.1)(1.9) - 1.2) = 1.761 \end{aligned}$$

S2-2003

### Error and Stability in Numerical Methods

There are two sources of errors in numerical methods: *round-off* and *truncation*.

Round-off error is due to finite precision of floating point numbers.

Truncation error is due to numerical methods *approximating* true solution curve as a series of points  $(x_n, y_n)$ .

At each step in the method a *local truncation* error occurs. As successive approximations are calculated using previous approximations a *global truncation* error accumulates.

Euler's method has a local truncation error of  $O(h^2)$ , where  $h$  is the step size and a *global truncation* error of  $O(h)$ .

Another issue in numerical methods is *stability*. A stable method is *insensitive* to small changes in initial conditions. A numerical method is *unstable* if it is not stable. Some methods may be more stable than others.

S2-2003

# Euler's Improved Method

Euler's method is simple, but inaccurate. For this reason it is rarely used in serious applications (although it may suffice in not-so-serious applications such as games).

A relatively straightforward modification to Euler's method gives *the improved Euler's Method*, reducing the global truncation error to  $O(h^2)$ .

In the improved Euler's method the new value for  $y$  is computed as follows

$$y_{n+1} = y_n + h \frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)}{2}$$

The formula

$$y_{n+1}^* = y_n + hf(x_n, y_n)$$

is known as the *improved Euler formula*.

Euler's formula is used to obtain the initial estimate  $y_{n+1}^*$ .

S2-2003

The values  $f(x_n, y_n)$  and  $f(x_{n+1}, y_{n+1}^*)$  are estimates of the slope of the solution curve of the differential equation at  $(x_n, y(x_n))$  and  $(x_{n+1}, y(x_{n+1}^*))$ .

The average of these two values

$$\frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)}{2}$$

is an estimate of the average slope between the slope at  $x_n$  and at  $x_{n+1}$ . This value is used as the gradient to compute  $y_{n+1}$ .

Thus the difference between Euler's method and the improved Euler method is that the improved method uses a slope which is an estimate of the average slope over the interval  $x_n$  and  $x_{n+1}$  rather than the slope at  $x_n$ .

The improved Euler method is a *predictor-corrector* method. The method predicts the value  $y_{n+1}$  using Euler's method and then corrects it using

$$y_{n+1} = y_n + h \frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)}{2}$$

S2-2003

Acheson shows this graphically in fig 4.9:

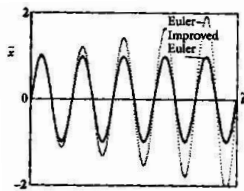


Fig. 4.9 Euler method v. improved Euler method: an example.

Figure 4: from p. 50 of Acheson

S2-2003

## Other Numerical Methods

There are many other numerical methods for solving differential equations which have reduced truncation error and better stability. Generally there is a trade-off between accuracy and compute time.

Some other numerical methods are: Taylor methods Runge-Kutta methods multi-step methods including Adams-Bashforth and Adams-Moulton methods.

Numerical methods is a field of study in its own right. There are specialist textbooks on numerical methods and numerical analysis. One is "Numerical Recipes in C".

S2-2003