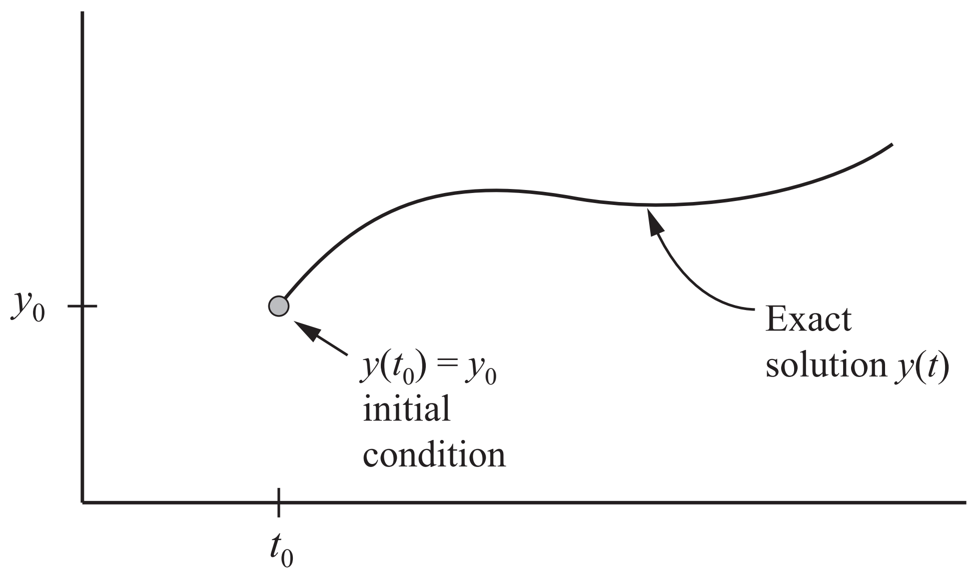
Euler's Method for Integration of Ordinary Differential Equations for Initial Value Problems

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@pdx.edu

Numerical Integration of ODEs

Graphical Interpretation of exact solution with initial condition.



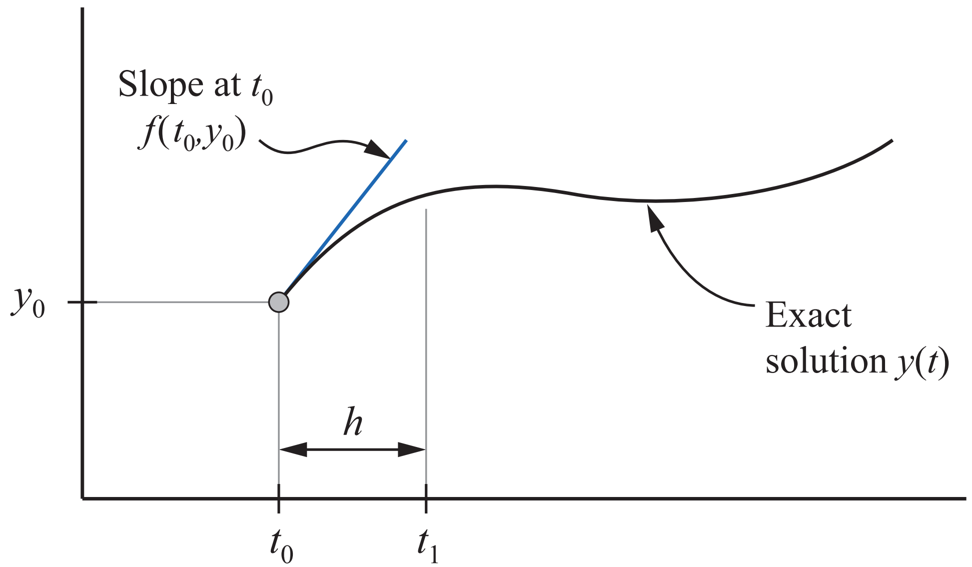
Nomenclature for first-order ODE
(initial value problem)

$$\frac{dy}{dt} = f(t, y), \quad t \geq 0; \quad y(t = 0) = y_0$$

$$y(t) = \text{exact solution}$$

Numerical Integration of ODEs

Use the slope at (t_0, y_0) to predict $y(t > 0)$. We can compute $f(t_0, y_0)$ exactly because $y_0 = y(t_0)$ is known.



Nomenclature for first-order ODE

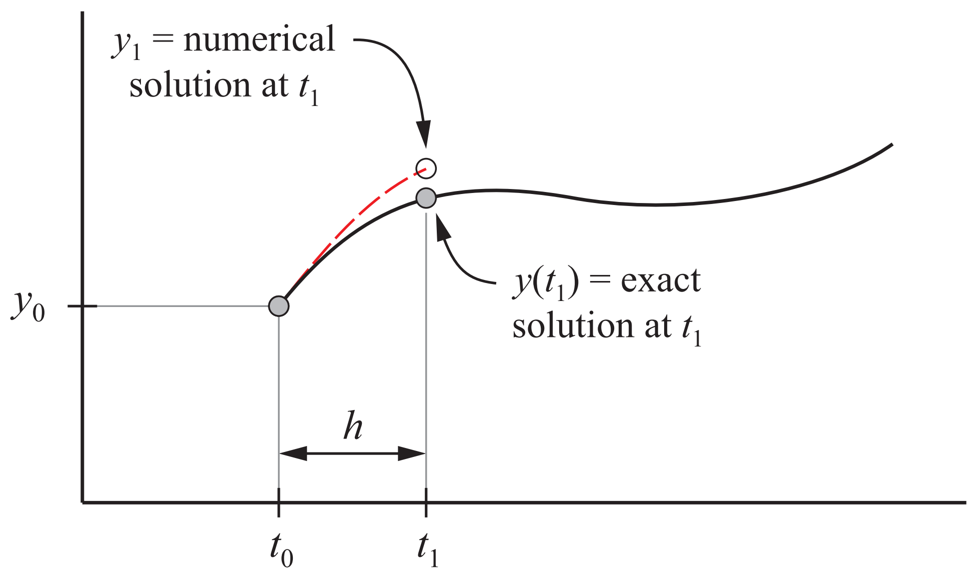
$$\frac{dy}{dt} = f(t, y), \quad t \geq 0; \quad y(t = 0) = y_0$$

$y(t)$ = exact solution

$y(t_j)$ = exact solution evaluated at t_j

Numerical Integration of ODEs

Numerical solution at t_1 may use other estimates of slope.



Nomenclature for first-order ODE

$$\frac{dy}{dt} = f(t, y), \quad t \geq 0; \quad y(t = 0) = y_0$$

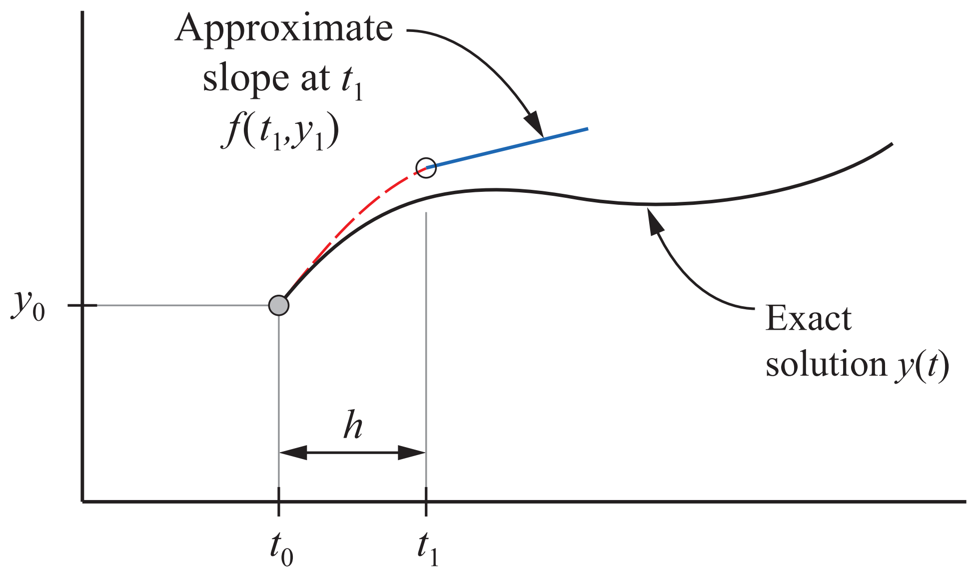
$$y(t) = \text{exact solution}$$

$$y(t_j) = \text{exact solution evaluated at } t_j$$

$$y_j = \text{approximate solution at } t_j$$

Numerical Integration of ODEs

Repeat process for step 2:
$f(t_1, y_1)$ is an *approximation* to the slope, since $y \approx y(t_1)$.



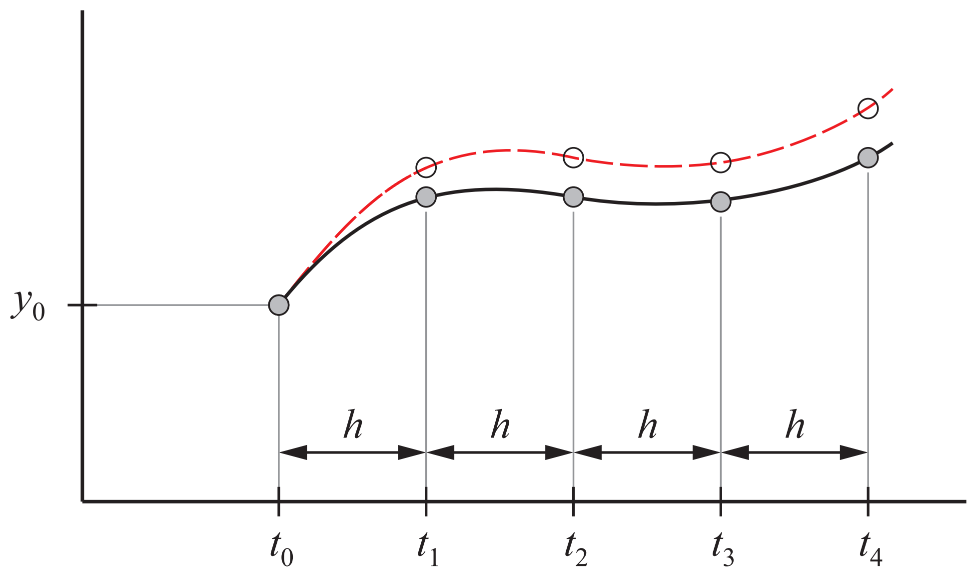
Nomenclature for first-order ODE

$$\frac{dy}{dt} = f(t, y), \quad t \geq 0; \quad y(t=0) = y_0$$

$$
\begin{aligned}
y(t) &= \text{exact solution} \\
y(t_j) &= \text{exact solution evaluated at } t_j \\
y_j &= \text{approximate solution at } t_j \\
f(t_j, y_j) &= \text{approximate r.h.s. at } t_j
\end{aligned}
$$

Numerical Integration of ODEs

The numerical solution is a set of discrete points. The dashed red curve is just to "guide your eye".



Nomenclature for first-order ODE

$$\frac{dy}{dt} = f(t, y), \quad t \geq 0; \quad y(t = 0) = y_0$$

$$\begin{aligned} y(t) &= \text{exact solution} \\ y(t_j) &= \text{exact solution evaluated at } t_j \\ y_j &= \text{approximate solution at } t_j \\ f(t_j, y_j) &= \text{approximate r.h.s. at } t_j \end{aligned}$$

Euler's Method 1

Consider a Taylor series expansion in the neighborhood of t_0

$$y(t) = y(t_0) + (t - t_0) \left.\frac{dy}{dt}\right|_{t_0} + \frac{(t - t_0)^2}{2} \left.\frac{d^2y}{dt^2}\right|_{t_0} + \ldots$$

Euler's Method 1

Consider a Taylor series expansion in the neighborhood of t_0

$$y(t) = y(t_0) + (t - t_0)\left.\frac{dy}{dt}\right|_{t_0} + \frac{(t - t_0)^2}{2}\left.\frac{d^2y}{dt^2}\right|_{t_0} + \ldots$$

Retain only first derivative term and define

$$f(t_0, y_0) \equiv \left.\frac{dy}{dt}\right|_{t_0}$$

Euler's Method 1

Consider a Taylor series expansion in the neighborhood of t_0

$$y(t) = y(t_0) + (t - t_0)\left.\frac{dy}{dt}\right|_{t_0} + \frac{(t - t_0)^2}{2}\left.\frac{d^2y}{dt^2}\right|_{t_0} + \ldots$$

Retain only first derivative term and define

$$f(t_0, y_0) \equiv \left.\frac{dy}{dt}\right|_{t_0}$$

to get

$$y(t) \approx y(t_0) + (t - t_0)f(t_0, y_0)$$

or

$$y(t) \approx y(t_0) + hf(t_0, y_0)$$

Euler's Method 2

Given $h = t_1 - t_0$ and initial condition, $y = y(t_0)$, compute

$$y_1 = y_0 + h\, f(t_0, y_0)$$

Euler's Method 2

Given $h = t_1 - t_0$ and initial condition, $y = y(t_0)$, compute

$$y_1 = y_0 + h\, f(t_0, y_0)$$

$$y_2 = y_1 + h\, f(t_1, y_1)$$

Euler's Method 2

Given $h = t_1 - t_0$ and initial condition, $y = y(t_0)$, compute

$$
\begin{aligned}
y_1 &= y_0 + h\, f(t_0, y_0) \\
y_2 &= y_1 + h\, f(t_1, y_1) \\
\vdots \quad & \quad \vdots \\
y_{j+1} &= y_j + h\, f(t_j, y_j)
\end{aligned}
$$

Euler's Method 2

Given $h = t_1 - t_0$ and initial condition, $y = y(t_0)$, compute

$$\begin{aligned} y_1 &= y_0 + h\, f(t_0, y_0) \\ y_2 &= y_1 + h\, f(t_1, y_1) \\ &\vdots \qquad \vdots \\ y_{j+1} &= y_j + h\, f(t_j, y_j) \end{aligned}$$

or, shifting indices by 1

$$\boxed{y_j = y_{j-1} + h\, f(t_{j-1}, y_{j-1})}$$

Example: Euler's Method by Hand

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

Example: Euler's Method by Hand

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

The exact solution is

$$y = \frac{1}{4}\left[2t - 1 + 5e^{-2t}\right]$$

Example: Euler's Method by Hand

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

The exact solution is

$$y = \frac{1}{4}\left[2t - 1 + 5e^{-2t}\right]$$

j	t_j	$f(t_{j-1}, y_{j-1})$	Euler $y_j = y_{j-1} + h\,f(t_{j-1}, y_{j-1})$	Exact $y(t_j)$	Error $y_j - y(t_j)$
0	0.0	NA	(initial condition) 1.0000	1.0000	0

Example: Euler's Method by Hand

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

The exact solution is

$$y = \frac{1}{4}\left[2t - 1 + 5e^{-2t}\right]$$

j	t_j	$f(t_{j-1}, y_{j-1})$	Euler $y_j = y_{j-1} + h\, f(t_{j-1}, y_{j-1})$	Exact $y(t_j)$	Error $y_j - y(t_j)$
0	0.0	NA	(initial condition) 1.0000	1.0000	0
1	0.2	$0 - (2)(1) = -2.000$	$1.0 + (0.2)(-2.0) = 0.6000$	0.6879	-0.0879

Example: Euler's Method by Hand

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

The exact solution is

$$y = \frac{1}{4}\left[2t - 1 + 5e^{-2t}\right]$$

j	t_j	$f(t_{j-1}, y_{j-1})$	Euler $y_j = y_{j-1} + h\,f(t_{j-1}, y_{j-1})$	Exact $y(t_j)$	Error $y_j - y(t_j)$
0	0.0	NA	(initial condition) 1.0000	1.0000	0
1	0.2	$0 - (2)(1) = -2.000$	$1.0 + (0.2)(-2.0) = 0.6000$	0.6879	-0.0879
2	0.4	$0.2 - (2)(0.6) = -1.000$	$0.6 + (0.2)(-1.0) = 0.4000$	0.5117	-0.1117
3	0.6	$0.4 - (2)(0.4) = -0.400$	$0.4 + (0.2)(-0.4) = 0.3200$	0.4265	-0.1065

Simple MATLAB Implementation

Note: The first index in a MATLAB array is 1, not 0.

Therefore, we need to interpret the formula for Euler's method as having an initial condition at `t(1)` with a value of `y(1)`. This is not hard, but it does take a conscious shift for us to associate `t(1)` with y_0.

But why did we use t_0 and y_0 to designate the initial condition?

Answers: First it's convention. Second it is natural to associate the initial condition with a time of zero. The subscript t_0 reinforces that idea for analytical work.

Simple Matlab Implementation

Euler's method is easy to implement in Matlab

```
h = 0.2;             %  stepsize
tn = 1;              %  stopping time
y0 = 1;              %  initial condition

t = (0:h:tn)';       %  Column vector of elements with spacing h
n = length(t);       %  Number of elements in the t vector
y = y0*ones(n,1);    %  Preallocate y for speed

%   Euler scheme; j=1 for initial condition
for j=2:n
   y(j) = y(j-1) + h*( t(j-1) - 2*y(j-1) );
end
```

Simple MATLAB Implementation

Euler's method is easy to implement in MATLAB

```
h = 0.2;            %  stepsize
tn = 1;             %  stopping time
y0 = 1;             %  initial condition

t = (0:h:tn)';      %  Column vector of elements with spacing h
n = length(t);      %  Number of elements in the t vector
y = y0*ones(n,1);   %  Preallocate y for speed

%   Euler scheme; j=1 for initial condition
for j=2:n
   y(j) = y(j-1) + h*( t(j-1) - 2*y(j-1) );
end
```

This code is limited because the $f(t, y)$ function is hard-coded. We need a more general solution.

A general implementation of Euler's method separates the evaluation of f (the right hand side function) from the basic algorithm that advances the ODE.

Implementation of Euler's Method

```
function [t,y] = odeEuler(diffeq,tn,h,y0)
% odeEuler  Euler's method for integration of a single, first order ODE
%
% Synopsis:   [t,y] = odeEuler(diffeq,tn,h,y0)
%
% Input:      diffeq = (string) name of the m-file that evaluates the right
%                      hand side of the ODE written in standard form
%             tn  = stopping value of the independent variable
%             h   = stepsize for advancing the independent variable
%             y0  = initial condition for the dependent variable
%
% Output:     t = vector of independent variable values:  t(j) = (j-1)*h
%             y = vector of numerical solution values at the t(j)

t = (0:h:tn)';           %  Column vector of elements with spacing h
n = length(t);           %  Number of elements in the t vector
y = y0*ones(n,1);        %  Preallocate y for speed

%   Begin Euler scheme; j=1 for initial condition
for j=2:n
   y(j) = y(j-1) + h*feval(diffeq,t(j-1),y(j-1));
end
```