Ordinary Differential Equations

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

$$\frac{dy}{dx} = z(x)$$

$$\frac{dz}{dx} = r(x) - q(x)z(x)$$

Problems involving ODEs can always be reduced to a set of first order differential equations. For example,

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

By introducing a new variable z, this can be rewritten as:

$$\frac{dy}{dx} = z(x)$$
$$\frac{dz}{dx} = r(x) - q(x)z(x)$$

This exemplifies the procedure for an arbitrary ODE. The usual choice for the new variables is to let them just be derivatives of each other and of the original variable.

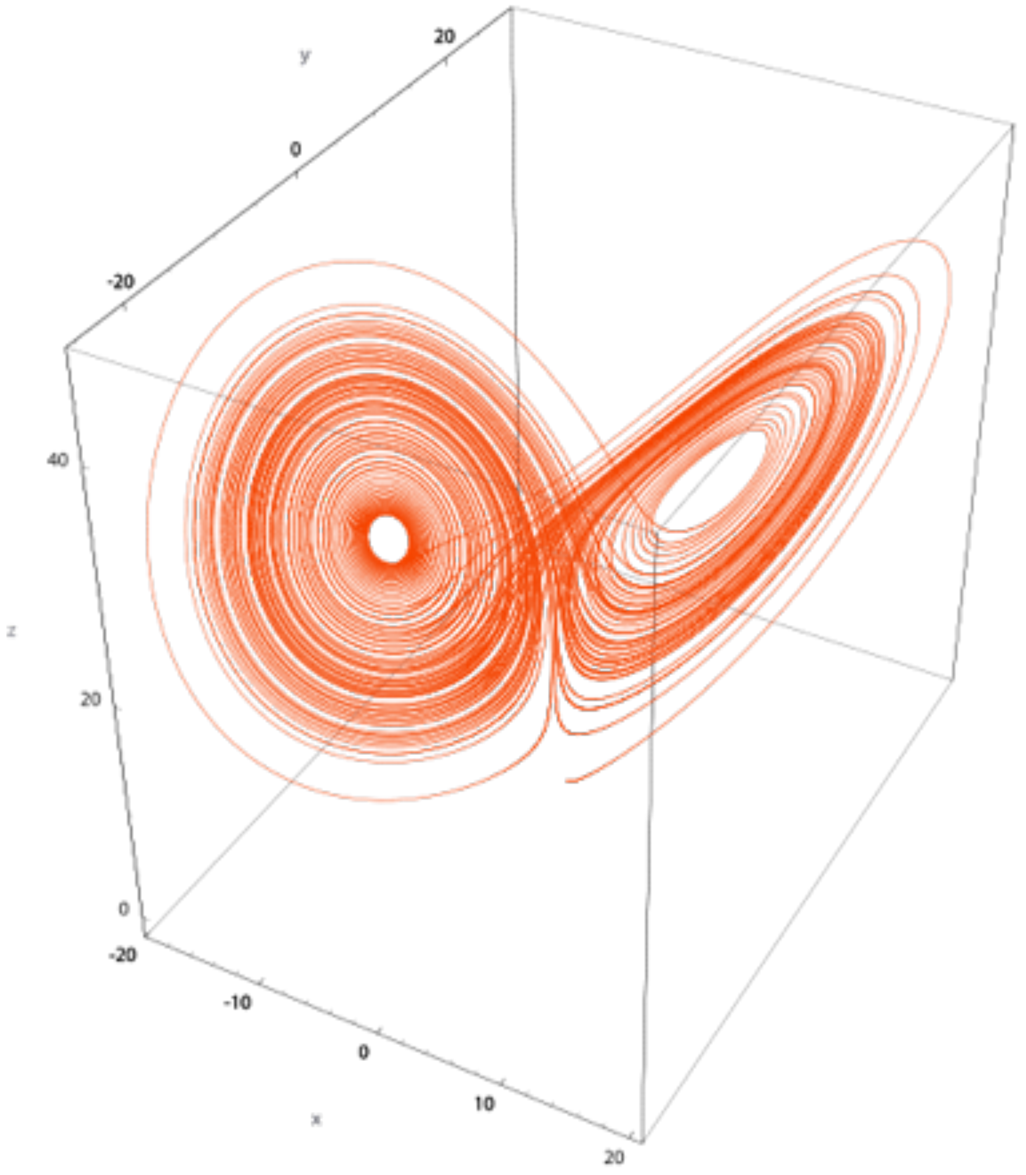
Let common sense be your guide: If you find that the original variables are smooth in a solution, while your auxiliary variables are doing crazy things, then figure out why and choose different auxiliary variables.

The generic problem in ordinary differential equations is thus reduced to the study of a set of N coupled *first-order* differential equations for the functions $y_i, \ i = 1, 2, \ldots, N$, having the general form

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, \ldots, y_N), \qquad i = 1, \ldots, N$$

where the functions f_i on the right-hand side are known.

Lorenz Oscillator



The equations that govern the Lorenz oscillator are:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

where σ is called the **Prandtl number** and ρ is called the **Rayleigh number**. All σ, ρ, β > 0, but usually $\sigma = 10$, $\beta = 8 / 3$ and ρ is varied. The system exhibits chaotic behavior for $\rho = 28$ but displays knotted periodic orbits for other values of ρ. For example, with $\rho = 99.96$ it becomes a *T*(3,2) torus knot.

$$\frac{\mathrm{d[O(^1D)]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k_6^{bi}[O(^1D)][N_2]} & -\mathrm{k_7^{bi}[O(^1D)][O_2]} & -\mathrm{k_{32}^{bi}[H_2O][O(^1D)]} \\
-\mathrm{k_{37}^{bi}[N_2O][O(^1D)]} & -\mathrm{k_{39}^{bi}[H_2][O(^1D)]} & -\mathrm{k_{40}^{bi}[CH_4][O(^1D)]} \\
-\mathrm{k_{45}^{bi}[CH_4][O(^1D)]} & -\mathrm{k_{61}^{bi}[N_2O][O(^1D)]} & -\mathrm{k_{63}^{bi}[O(^1D)][CF_2Cl_2]} \\
-\mathrm{k_{75}^{bi}[CH_4][O(^1D)]} & -\mathrm{k_{76}^{bi}[O(^1D)][CH_3Br]} & -\mathrm{k_{77}^{bi}[O(^1D)][O_3]} \\
-\mathrm{k_{78}^{bi}[O(^1D)][O_3]} & -\mathrm{k_{92}^{tri}[O(^1D)][N_2]} & +\mathrm{j_{103}[BrO]} \\
+\mathrm{j_{111}[CO_2]} & +\mathrm{j_{119}[H_2O]} & +\mathrm{j_{134}[N_2O]} \\
+\mathrm{j_{138}[NO_2]} & +\mathrm{j_{142}[O_2]} & +\mathrm{j_{145}[O_3]} \\
+\mathrm{j_{147}[OClO]} & &
\end{array}$$

$$\frac{\mathrm{d[O(^3P)]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k_1^{bi}[O(^3P)][O_3]} & +\mathrm{k_6^{bi}[O(^1D)][N_2]} & +\mathrm{k_7^{bi}[O(^1D)][O_2]} \\
+\mathrm{k_{16}^{bi}[N][O_2]} & -\mathrm{k_{17}^{bi}[HCHO][O(^3P)]} & -\mathrm{k_{21}^{bi}[ClO][O(^3P)]} \\
-\mathrm{k_{27}^{bi}[O(^3P)][H_2O_2]} & -\mathrm{k_{28}^{bi}[O(^3P)][ClONO_2]} & -\mathrm{k_{34}^{bi}[OH][O(^3P)]} \\
-\mathrm{k_{35}^{bi}[HO_2][O(^3P)]} & -\mathrm{k_{36}^{bi}[NO_2][O(^3P)]} & +\mathrm{k_{38}^{bi}[N][NO]} \\
+\mathrm{k_{47}^{bi}[OH][OH]} & +\mathrm{k_{58}^{bi}[H][HO_2]} & +\mathrm{k_{64}^{bi}[N][NO_2]} \\
-\mathrm{k_{74}^{bi}[BrO][O(^3P)]} & +2\mathrm{k_{78}^{bi}[O(^1D)][O_3]} & -\mathrm{k_{82}^{tri}[O(^3P)][O_2]} \\
-\mathrm{k_{88}^{tri}[NO_2][O(^3P)]} & -\mathrm{k_{91}^{tri}[NO][O(^3P)]} & -2\mathrm{k_{93}^{tri}[O(^3P)][O(^3P)]} \\
-\mathrm{k_{96}^{tri}[CO][O(^3P)]} & +\mathrm{j_{102}[BrO]} & +\mathrm{j_{110}[CO_2]} \\
+\mathrm{j_{115}[ClO]} & +\mathrm{j_{117}[ClOO]} & +\mathrm{j_{118}[H_2O]} \\
+\mathrm{j_{126}[HO_2]} & +\mathrm{j_{133}[CH_3OOH]} & +\mathrm{j_{135}[N_2O_5]} \\
+\mathrm{j_{137}[NO]} & +\mathrm{j_{139}[NO_2]} & +\mathrm{j_{140}[NO_3]} \\
+\mathrm{j_{142}[O_2]} & +2\mathrm{j_{143}[O_2]} & +\mathrm{j_{144}[O_3]} \\
+\mathrm{j_{146}[OClO]} & &
\end{array}$$

$$\frac{\mathrm{d[O_3]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k_1^{bi}[O(^3P)][O_3]} & -\mathrm{k_2^{bi}[OH][O_3]} & -\mathrm{k_3^{bi}[HO_2][O_3]} \\
-\mathrm{k_8^{bi}[NO][O_3]} & -\mathrm{k_9^{bi}[NO_2][O_3]} & -\mathrm{k_{10}^{bi}[H][O_3]} \\
-\mathrm{k_{20}^{bi}[Cl][O_3]} & -\mathrm{k_{65}^{bi}[Br][O_3]} & -\mathrm{k_{77}^{bi}[O(^1D)][O_3]} \\
-\mathrm{k_{78}^{bi}[O(^1D)][O_3]} & +\mathrm{k_{82}^{tri}[O(^3P)][O_2]} & -\mathrm{j_{144}[O_3]} \\
-\mathrm{j_{145}[O_3]} & &
\end{array}$$

$$\frac{\mathrm{d[N]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k_{16}^{bi}[N][O_2]} & -\mathrm{k_{38}^{bi}[N][NO]} & -\mathrm{k_{55}^{bi}[N][OH]} \\
-\mathrm{k_{64}^{bi}[N][NO_2]} & +\mathrm{j_{137}[NO]} &
\end{array}$$

$$\frac{\mathrm{d[H]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k}^{\mathrm{bi}}_{10}[\mathrm{H}][\mathrm{O_3}] & +\mathrm{k}^{\mathrm{bi}}_{19}[\mathrm{Cl}][\mathrm{H_2}] & +\mathrm{k}^{\mathrm{bi}}_{26}[\mathrm{H_2}][\mathrm{OH}] \\
+\mathrm{k}^{\mathrm{bi}}_{29}[\mathrm{CO}][\mathrm{OH}] & +\mathrm{k}^{\mathrm{bi}}_{34}[\mathrm{OH}][\mathrm{O(^3P)}] & +\mathrm{k}^{\mathrm{bi}}_{39}[\mathrm{H_2}][\mathrm{O(^1D)}] \\
+\mathrm{k}^{\mathrm{bi}}_{55}[\mathrm{N}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{57}[\mathrm{H}][\mathrm{HO_2}] & -\mathrm{k}^{\mathrm{bi}}_{58}[\mathrm{H}][\mathrm{HO_2}] \\
-\mathrm{k}^{\mathrm{bi}}_{59}[\mathrm{H}][\mathrm{HO_2}] & +\mathrm{k}^{\mathrm{bi}}_{75}[\mathrm{CH_4}][\mathrm{O(^1D)}] & -\mathrm{k}^{\mathrm{bi}}_{79}[\mathrm{H}][\mathrm{NO_2}] \\
-\mathrm{k}^{\mathrm{tri}}_{83}[\mathrm{H}][\mathrm{O_2}] & +\mathrm{j}_{109}[\mathrm{CH_4}] & +2\mathrm{j}_{118}[\mathrm{H_2O}] \\
+\mathrm{j}_{120}[\mathrm{H_2O}] & +\mathrm{j}_{123}[\mathrm{HCHO}] & +2\mathrm{j}_{124}[\mathrm{HCHO}] \\
+\mathrm{j}_{125}[\mathrm{HCl}] & +\mathrm{j}_{133}[\mathrm{CH_3OOH}] & +\mathrm{k}^{\mathrm{cr}}_{184}[\mathrm{M}] \\
+\mathrm{k}^{\mathrm{cr}}_{186}[\mathrm{HCl}] & &
\end{array}$$

$$\frac{\mathrm{d[OH]}}{\mathrm{dt}} = \begin{array}{lll}
-\mathrm{k}^{\mathrm{bi}}_{2}[\mathrm{OH}][\mathrm{O_3}] & +\mathrm{k}^{\mathrm{bi}}_{3}[\mathrm{HO_2}][\mathrm{O_3}] & +\mathrm{k}^{\mathrm{bi}}_{10}[\mathrm{H}][\mathrm{O_3}] \\
-\mathrm{k}^{\mathrm{bi}}_{11}[\mathrm{OH}][\mathrm{ClONO_2}] & -\mathrm{k}^{\mathrm{bi}}_{12}[\mathrm{CH_4}][\mathrm{OH}] & +\mathrm{k}^{\mathrm{bi}}_{17}[\mathrm{HCHO}][\mathrm{O(^3P)}] \\
-\mathrm{k}^{\mathrm{bi}}_{23}[\mathrm{HCl}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{25}[\mathrm{OH}][\mathrm{H_2O_2}] & -\mathrm{k}^{\mathrm{bi}}_{26}[\mathrm{H_2}][\mathrm{OH}] \\
+\mathrm{k}^{\mathrm{bi}}_{27}[\mathrm{O(^3P)}][\mathrm{H_2O_2}] & -\mathrm{k}^{\mathrm{bi}}_{29}[\mathrm{CO}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{30}[\mathrm{HNO_3}][\mathrm{OH}] \\
+\mathrm{k}^{\mathrm{bi}}_{31}[\mathrm{NO}][\mathrm{HO_2}] & +2\mathrm{k}^{\mathrm{bi}}_{32}[\mathrm{H_2O}][\mathrm{O(^1D)}] & -\mathrm{k}^{\mathrm{bi}}_{33}[\mathrm{OH}][\mathrm{HO_2}] \\
-\mathrm{k}^{\mathrm{bi}}_{34}[\mathrm{OH}][\mathrm{O(^3P)}] & +\mathrm{k}^{\mathrm{bi}}_{35}[\mathrm{HO_2}][\mathrm{O(^3P)}] & +\mathrm{k}^{\mathrm{bi}}_{39}[\mathrm{H_2}][\mathrm{O(^1D)}] \\
+\mathrm{k}^{\mathrm{bi}}_{40}[\mathrm{CH_4}][\mathrm{O(^1D)}] & -\mathrm{k}^{\mathrm{bi}}_{41}[\mathrm{HCHO}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{44}[\mathrm{OH}][\mathrm{HO_2NO_2}] \\
-\mathrm{k}^{\mathrm{bi}}_{46}[\mathrm{OH}][\mathrm{CH_3OOH}] & -2\mathrm{k}^{\mathrm{bi}}_{47}[\mathrm{OH}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{48}[\mathrm{ClO}][\mathrm{OH}] \\
-\mathrm{k}^{\mathrm{bi}}_{49}[\mathrm{ClO}][\mathrm{OH}] & -\mathrm{k}^{\mathrm{bi}}_{50}[\mathrm{HOCl}][\mathrm{OH}] & +\mathrm{k}^{\mathrm{bi}}_{54}[\mathrm{Cl}][\mathrm{HO_2}] \\
-\mathrm{k}^{\mathrm{bi}}_{55}[\mathrm{N}][\mathrm{OH}] & +2\mathrm{k}^{\mathrm{bi}}_{59}[\mathrm{H}][\mathrm{HO_2}] & -\mathrm{k}^{\mathrm{bi}}_{68}[\mathrm{OH}][\mathrm{HBr}] \\
-\mathrm{k}^{\mathrm{bi}}_{69}[\mathrm{CH_3Br}][\mathrm{OH}] & +\mathrm{k}^{\mathrm{bi}}_{79}[\mathrm{H}][\mathrm{NO_2}] & -2\mathrm{k}^{\mathrm{tri}}_{84}[\mathrm{OH}][\mathrm{OH}] \\
-\mathrm{k}^{\mathrm{tri}}_{85}[\mathrm{OH}][\mathrm{NO_2}] & +\mathrm{j}_{120}[\mathrm{H_2O}] & +2\mathrm{j}_{121}[\mathrm{H_2O_2}] \\
+\mathrm{j}_{126}[\mathrm{HO_2}] & +\mathrm{j}_{128}[\mathrm{HO_2NO_2}] & +\mathrm{j}_{129}[\mathrm{HOBr}] \\
+\mathrm{j}_{130}[\mathrm{HOCl}] & +\mathrm{j}_{131}[\mathrm{HNO_3}] & +\mathrm{j}_{132}[\mathrm{CH_3OOH}] \\
+\mathrm{k}^{\mathrm{cr}}_{184}[\mathrm{M}] & +\mathrm{k}^{\mathrm{cr}}_{187}[\mathrm{HNO_3}] &
\end{array}$$

Non-linearities may exist

$$\frac{d[H]}{dt} = \begin{array}{lll}
-k^{bi}_{10}[H][O_3] & +k^{bi}_{19}[Cl][H_2] & +k^{bi}_{26}[H_2][OH] \\
+k^{bi}_{29}[CO][OH] & +k^{bi}_{34}[OH][O(^3P)] & +k^{bi}_{39}[H_2][O(^1D)] \\
+k^{bi}_{55}[N][OH] & -k^{bi}_{57}[H][HO_2] & -k^{bi}_{58}[H][HO_2] \\
-k^{bi}_{59}[H][HO_2] & +k^{bi}_{75}[CH_4][O(^1D)] & -k^{bi}_{79}[H][NO_2] \\
-k^{tri}_{83}[H][O_2] & +j_{109}[CH_4] & +2j_{118}[H_2O] \\
+j_{120}[H_2O] & +j_{123}[HCHO] & +2j_{124}[HCHO] \\
+j_{125}[HCl] & +j_{133}[CH_3OOH] & +k^{cr}_{184}[M] \\
+k^{cr}_{186}[HCl] & &
\end{array}$$

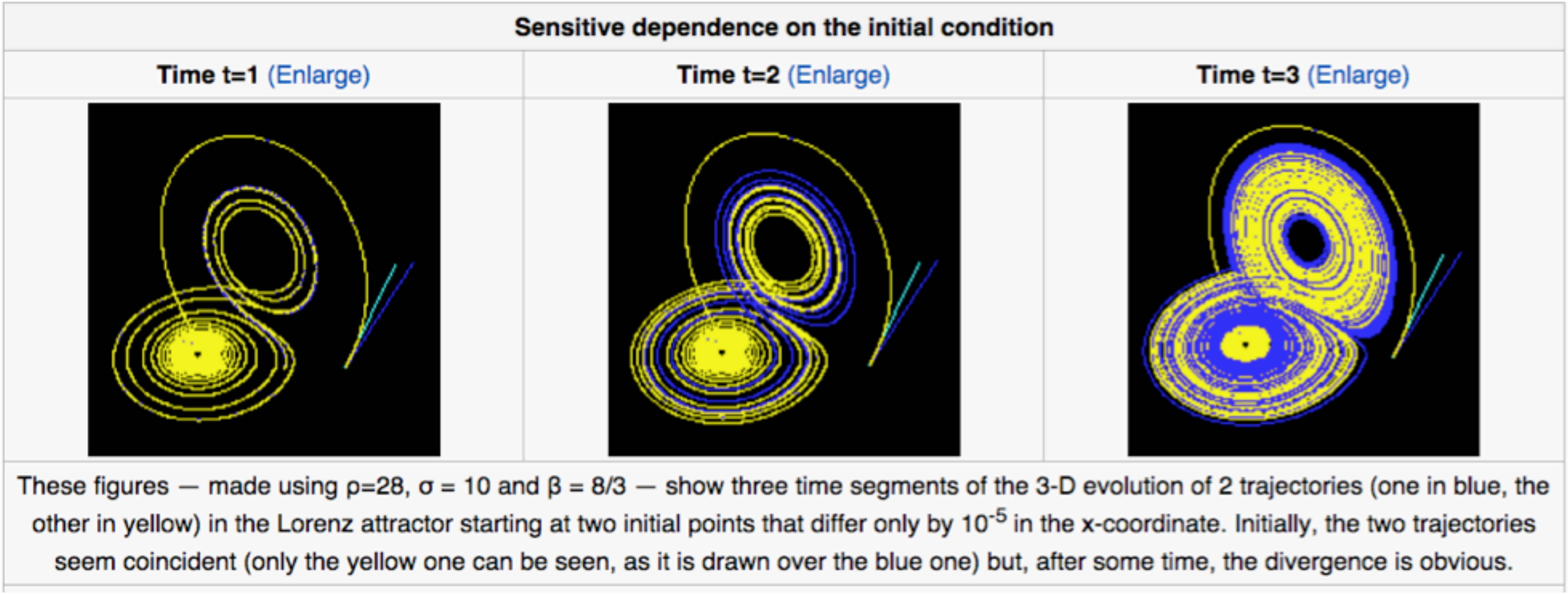
$$\frac{d[OH]}{dt} = \begin{array}{lll}
-k^{bi}_{2}[OH][O_3] & +k^{bi}_{3}[HO_2][O_3] & +k^{bi}_{10}[H][O_3] \\
-k^{bi}_{11}[OH][ClONO_2] & -k^{bi}_{12}[CH_4][OH] & +k^{bi}_{17}[HCHO][O(^3P)] \\
-k^{bi}_{23}[HCl][OH] & -k^{bi}_{25}[OH][H_2O_2] & -k^{bi}_{26}[H_2][OH] \\
+k^{bi}_{27}[O(^3P)][H_2O_2] & -k^{bi}_{29}[CO][OH] & -k^{bi}_{30}[HNO_3][OH] \\
+k^{bi}_{31}[NO][HO_2] & +2k^{bi}_{32}[H_2O][O(^1D)] & -k^{bi}_{33}[OH][HO_2] \\
-k^{bi}_{34}[OH][O(^3P)] & +k^{bi}_{35}[HO_2][O(^3P)] & +k^{bi}_{39}[H_2][O(^1D)] \\
+k^{bi}_{40}[CH_4][O(^1D)] & -k^{bi}_{41}[HCHO][OH] & -k^{bi}_{44}[OH][HO_2NO_2] \\
-k^{bi}_{46}[OH][CH_3OOH] & \boxed{-2k^{bi}_{47}[OH][OH]} & -k^{bi}_{48}[ClO][OH] \\
-k^{bi}_{49}[ClO][OH] & -k^{bi}_{50}[HOCl][OH] & +k^{bi}_{54}[Cl][HO_2] \\
-k^{bi}_{55}[N][OH] & +2k^{bi}_{59}[H][HO_2] & -k^{bi}_{68}[OH][HBr] \\
-k^{bi}_{69}[CH_3Br][OH] & +k^{bi}_{79}[H][NO_2] & \boxed{-2k^{tri}_{84}[OH][OH]} \\
-k^{tri}_{85}[OH][NO_2] & +j_{120}[H_2O] & +2j_{121}[H_2O_2] \\
+j_{126}[HO_2] & +j_{128}[HO_2NO_2] & +j_{129}[HOBr] \\
+j_{130}[HOCl] & +j_{131}[HNO_3] & +j_{132}[CH_3OOH] \\
+k^{cr}_{184}[M] & +k^{cr}_{187}[HNO_3] &
\end{array}$$

- Is a problem involving ODEs completely specified by its equations?

- Is a problem involving ODEs completely specified by its equations?

- NO! Even more crucial in determining how to attack the problem numerically is the nature of the boundary conditions. Boundary conditions are algebraic conditions on the values of the functions *y*$_i$ in:

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, \ldots, y_N), \qquad i = 1, \ldots, N$$



Sensitive dependence on the initial condition

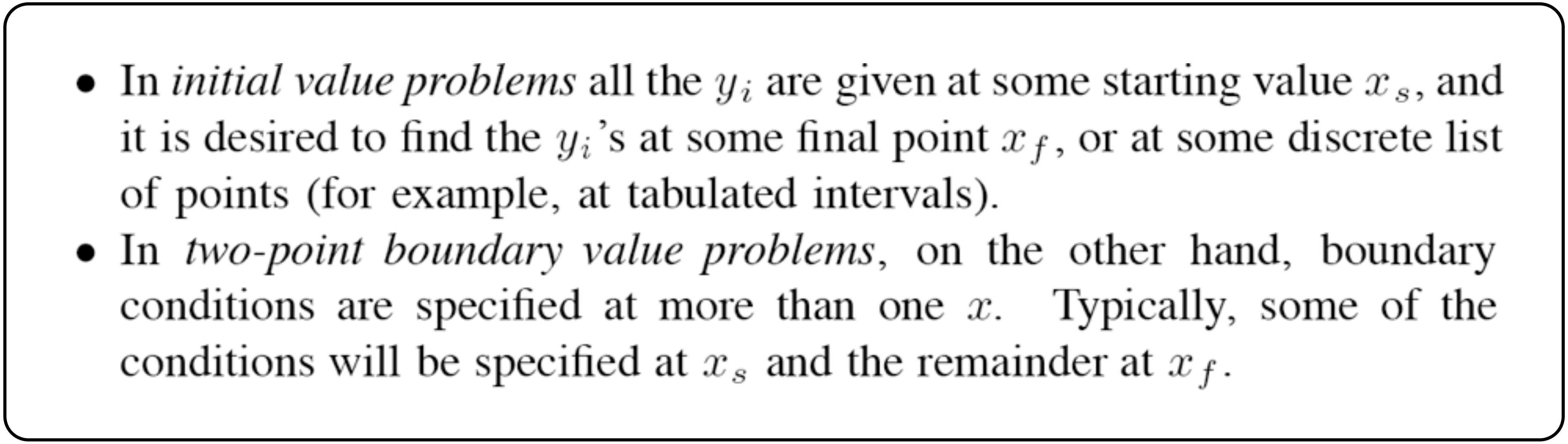
Time t=1 (Enlarge)	Time t=2 (Enlarge)	Time t=3 (Enlarge)

These figures — made using $\rho=28$, $\sigma = 10$ and $\beta = 8/3$ — show three time segments of the 3-D evolution of 2 trajectories (one in blue, the other in yellow) in the Lorenz attractor starting at two initial points that differ only by 10^{-5} in the x-coordinate. Initially, the two trajectories seem coincident (only the yellow one can be seen, as it is drawn over the blue one) but, after some time, the divergence is obvious.

- In general, the boundary conditions can be specified at discrete points, but do not hold between those points, i.e. are not preserved automatically by the differential equations.

- Boundary conditions may be as simple as requiring that certain variables have certain numerical values, or as complicated as a set of non-linear algebraic equations among the variables.

- Usually it is the nature of the boundary conditions that determines which numerical methods will be feasible. Boundary conditions divide into two broad categories.



- In *initial value problems* all the y_i are given at some starting value x_s, and it is desired to find the y_i's at some final point x_f, or at some discrete list of points (for example, at tabulated intervals).
- In *two-point boundary value problems*, on the other hand, boundary conditions are specified at more than one x. Typically, some of the conditions will be specified at x_s and the remainder at x_f.

The underlying idea of any routine for solving the initial value problem is always this: Rewrite the dy's and dx's in

as finite steps in Δy and Δx, and multiply the equations by Δx. This gives algebraic formulas for the change in the functions when the independent variable, x, is "stepped" by one "stepsize" Δx. In the limit of making the stepsize very small ($\Delta x \rightarrow 0$), a good approximation to the underlying differential equation is achieved.

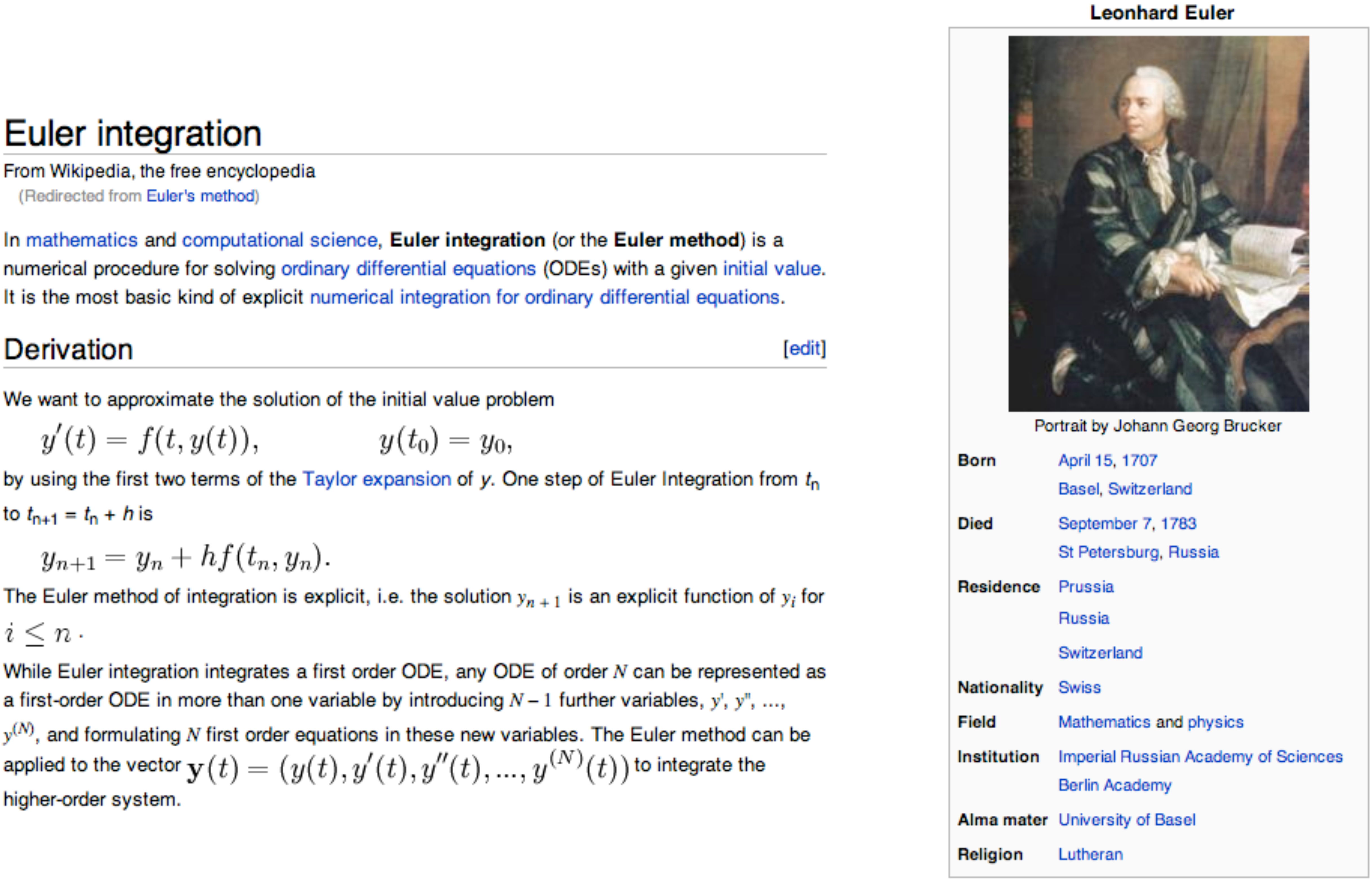
The underlying idea of any routine for solving the initial value problem is always this: Rewrite the dy's and dx's in

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, \ldots, y_N), \qquad i = 1, \ldots, N$$

as finite steps in Δy and Δx, and multiply the equations by Δx. This gives algebraic formulas for the change in the functions when the independent variable, x, is "stepped" by one "stepsize" Δx. In the limit of making the stepsize very small (Δx→0), a good approximation to the underlying differential equation is achieved.

Literal implementation of this procedure results in *Euler's* method, which is however not recommended for any practical use. *Euler's* method is conceptually important, however; one way or another practical methods all come down to the same idea:

Add small increments to your functions corresponding to derivatives (right-hand sides of the equations) multiplied by stepsizes.



Euler integration

From Wikipedia, the free encyclopedia

(Redirected from Euler's method)

In mathematics and computational science, **Euler integration** (or the **Euler method**) is a numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic kind of explicit numerical integration for ordinary differential equations.

Derivation

[edit]

We want to approximate the solution of the initial value problem

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0,$$

by using the first two terms of the Taylor expansion of y. One step of Euler Integration from t_n to $t_{n+1} = t_n + h$ is

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

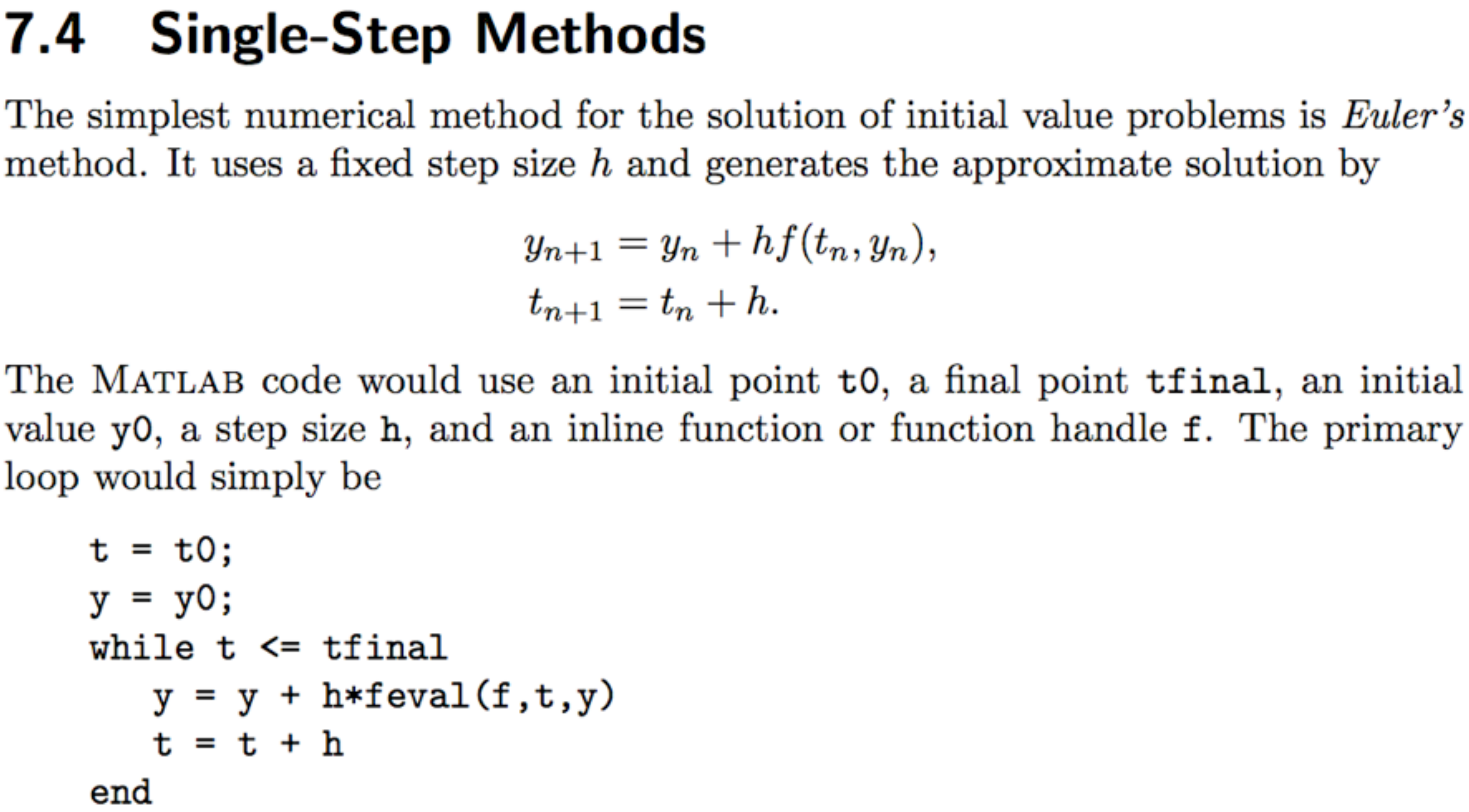
The Euler method of integration is explicit, i.e. the solution y_{n+1} is an explicit function of y_i for $i \leq n$.

While Euler integration integrates a first order ODE, any ODE of order N can be represented as a first-order ODE in more than one variable by introducing $N-1$ further variables, y', y'', ..., $y^{(N)}$, and formulating N first order equations in these new variables. The Euler method can be applied to the vector $\mathbf{y}(t) = (y(t), y'(t), y''(t), ..., y^{(N)}(t))$ to integrate the higher-order system.

Leonhard Euler

Portrait by Johann Georg Brucker

Born	April 15, 1707 Basel, Switzerland
Died	September 7, 1783 St Petersburg, Russia
Residence	Prussia Russia Switzerland
Nationality	Swiss
Field	Mathematics and physics
Institution	Imperial Russian Academy of Sciences Berlin Academy
Alma mater	University of Basel
Religion	Lutheran



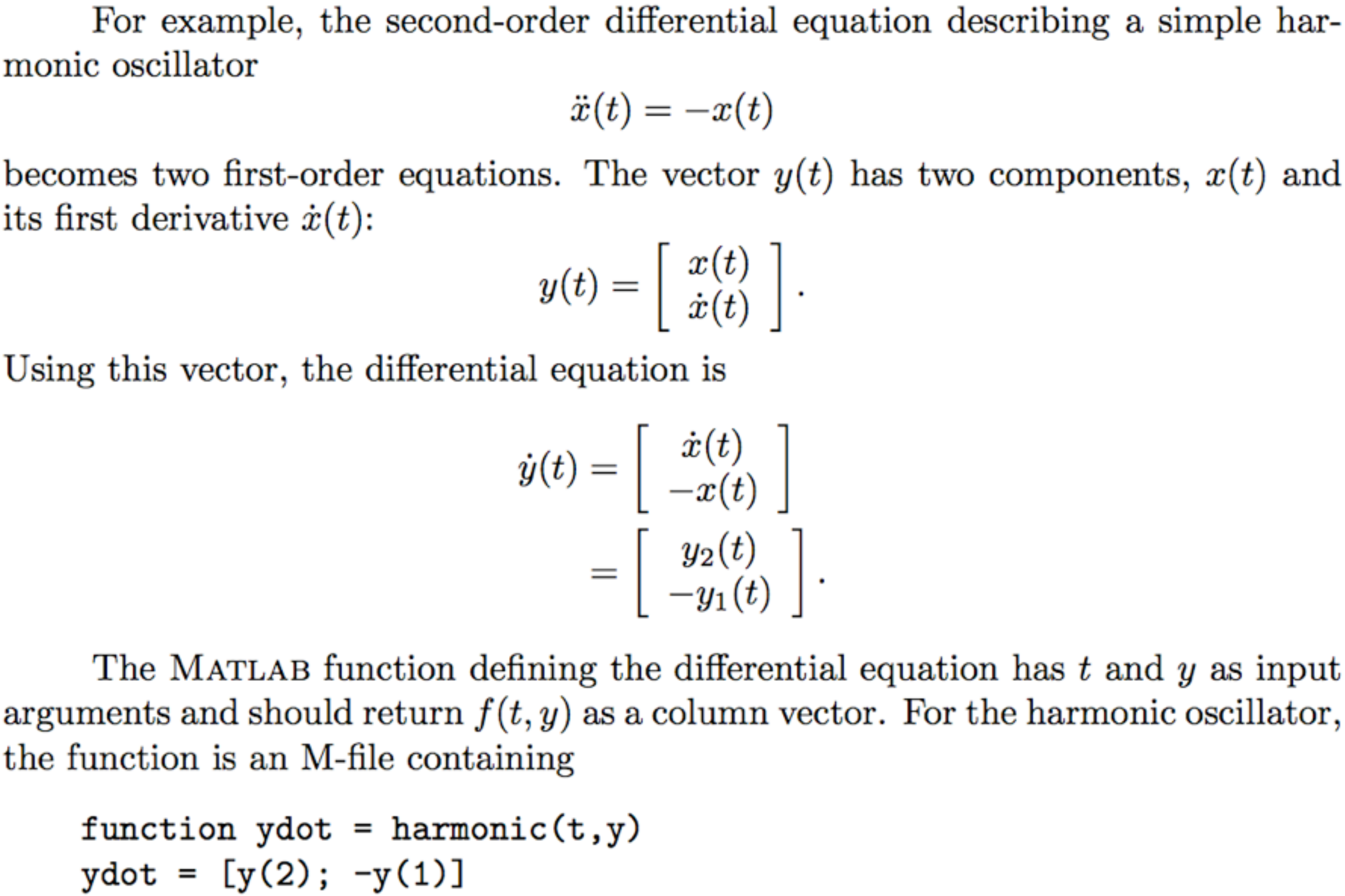
7.4 Single-Step Methods

The simplest numerical method for the solution of initial value problems is *Euler's* method. It uses a fixed step size h and generates the approximate solution by

$$\begin{aligned} y_{n+1} &= y_n + hf(t_n, y_n), \\ t_{n+1} &= t_n + h. \end{aligned}$$

The MATLAB code would use an initial point t0, a final point tfinal, an initial value y0, a step size h, and an inline function or function handle f. The primary loop would simply be

```
t = t0;
y = y0;
while t <= tfinal
   y = y + h*feval(f,t,y)
   t = t + h
end
```



For example, the second-order differential equation describing a simple harmonic oscillator

$$\ddot{x}(t) = -x(t)$$

becomes two first-order equations. The vector $y(t)$ has two components, $x(t)$ and its first derivative $\dot{x}(t)$:

$$y(t) = \left[\begin{array}{c} x(t) \\ \dot{x}(t) \end{array} \right].$$

Using this vector, the differential equation is

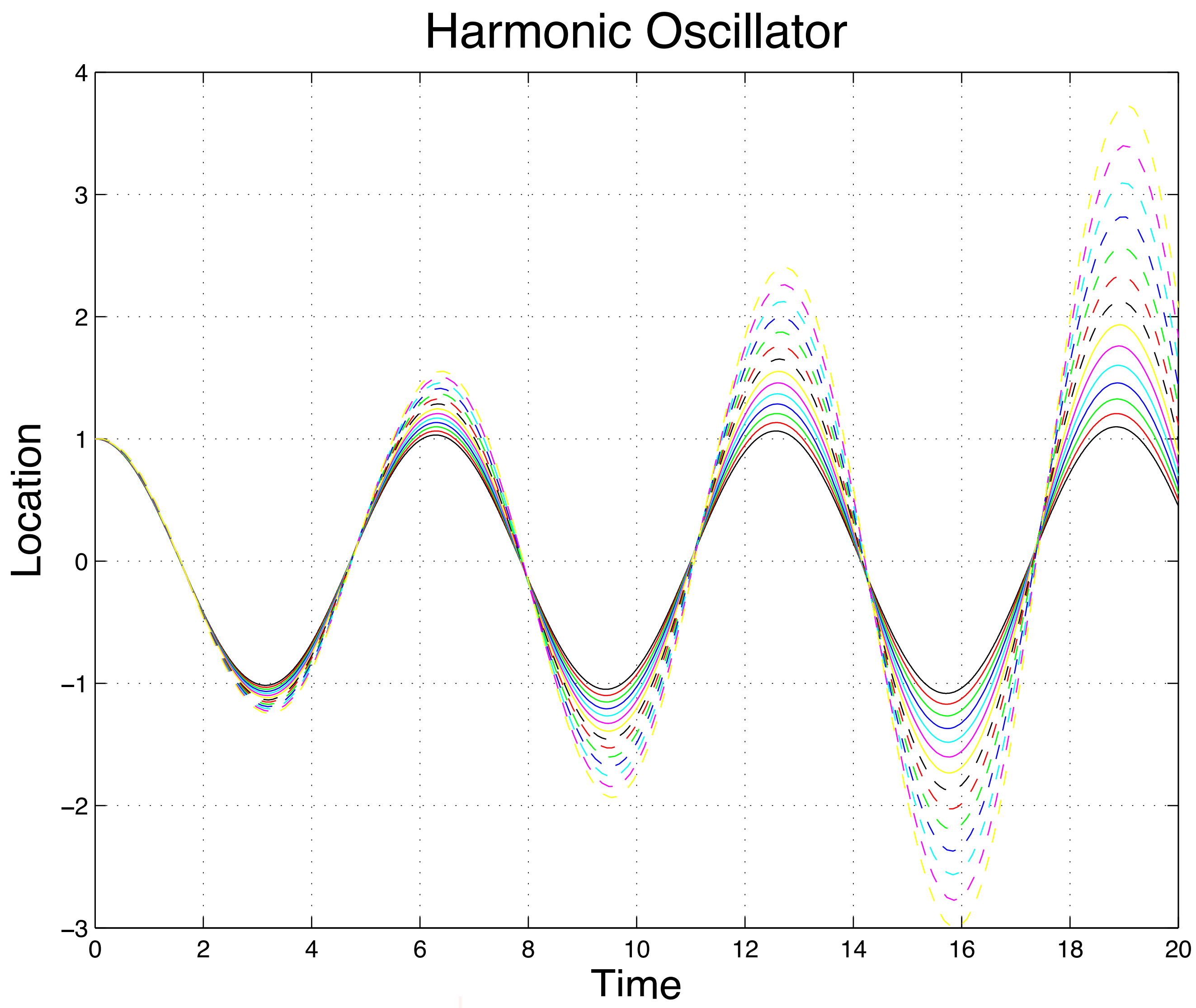
$$\begin{aligned} \dot{y}(t) &= \left[\begin{array}{c} \dot{x}(t) \\ -x(t) \end{array} \right] \\ &= \left[\begin{array}{c} y_2(t) \\ -y_1(t) \end{array} \right]. \end{aligned}$$

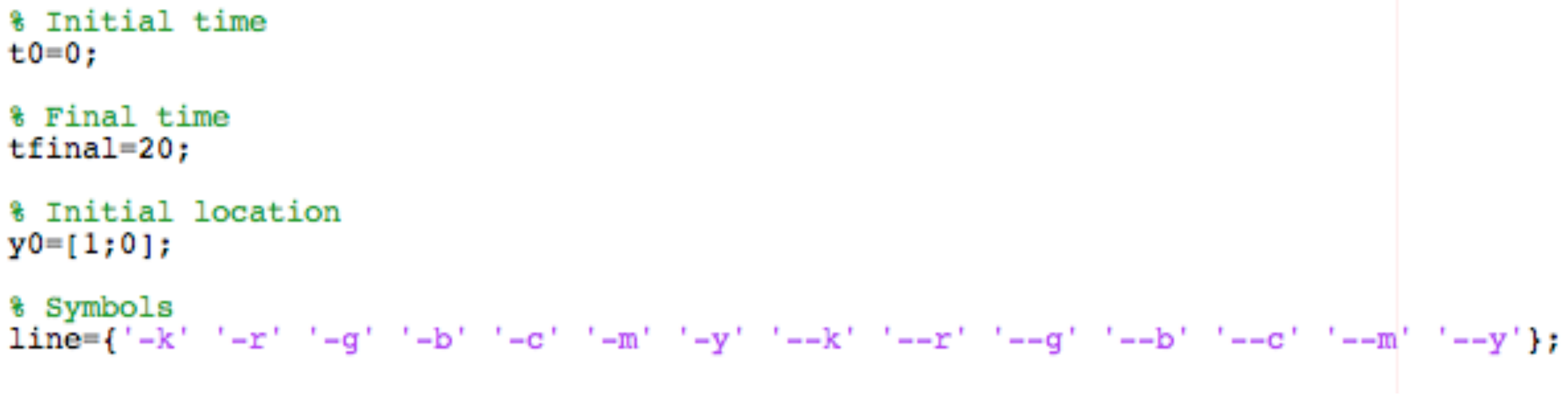
The MATLAB function defining the differential equation has t and y as input arguments and should return $f(t, y)$ as a column vector. For the harmonic oscillator, the function is an M-file containing

```
function ydot = harmonic(t,y)
ydot = [y(2); -y(1)]
```

Homework 4

- Starting from the conditions below write a Euler integration for the simple harmonic oscillator to produce the plot on the right.
- Why are the solutions diverging?

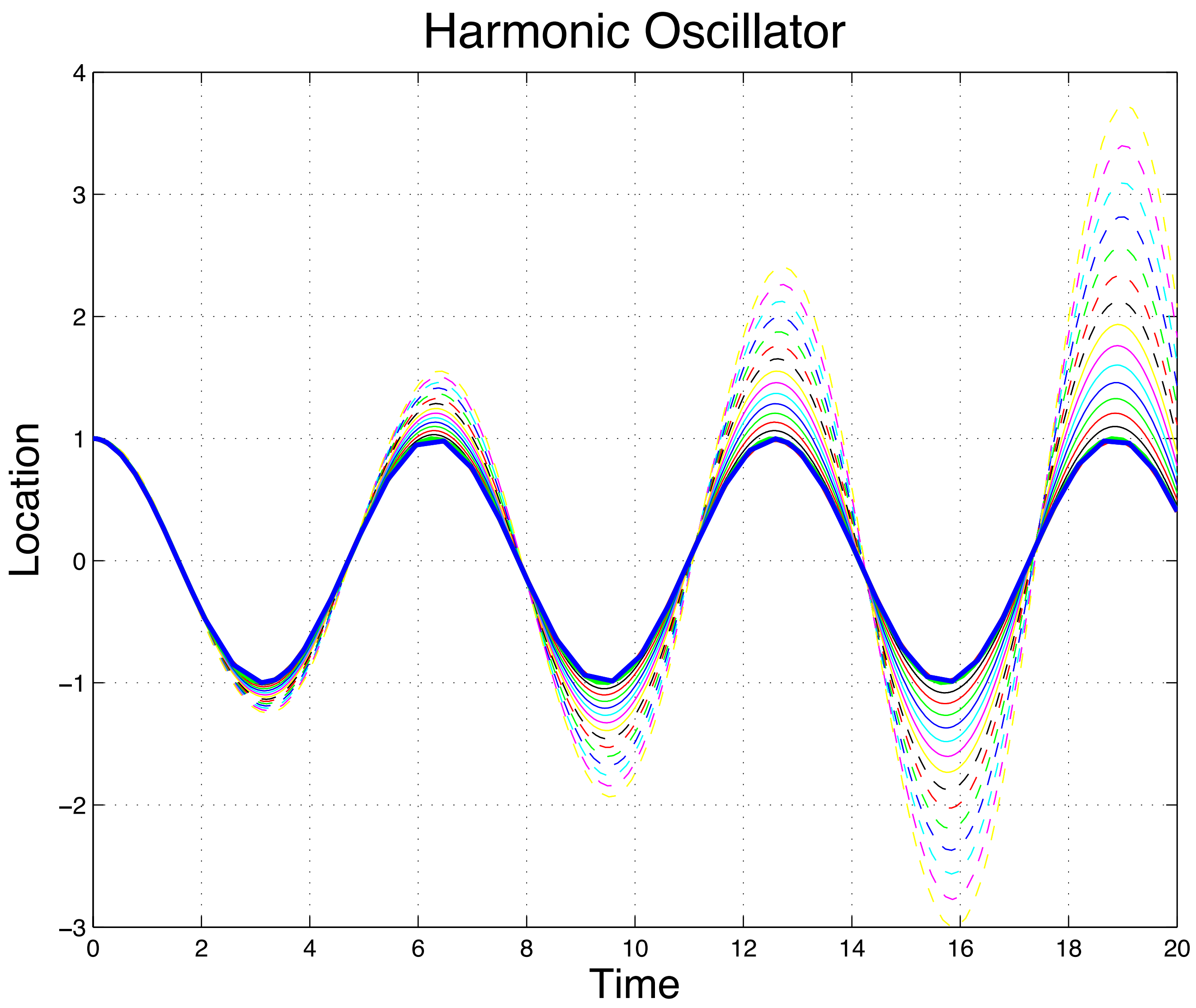


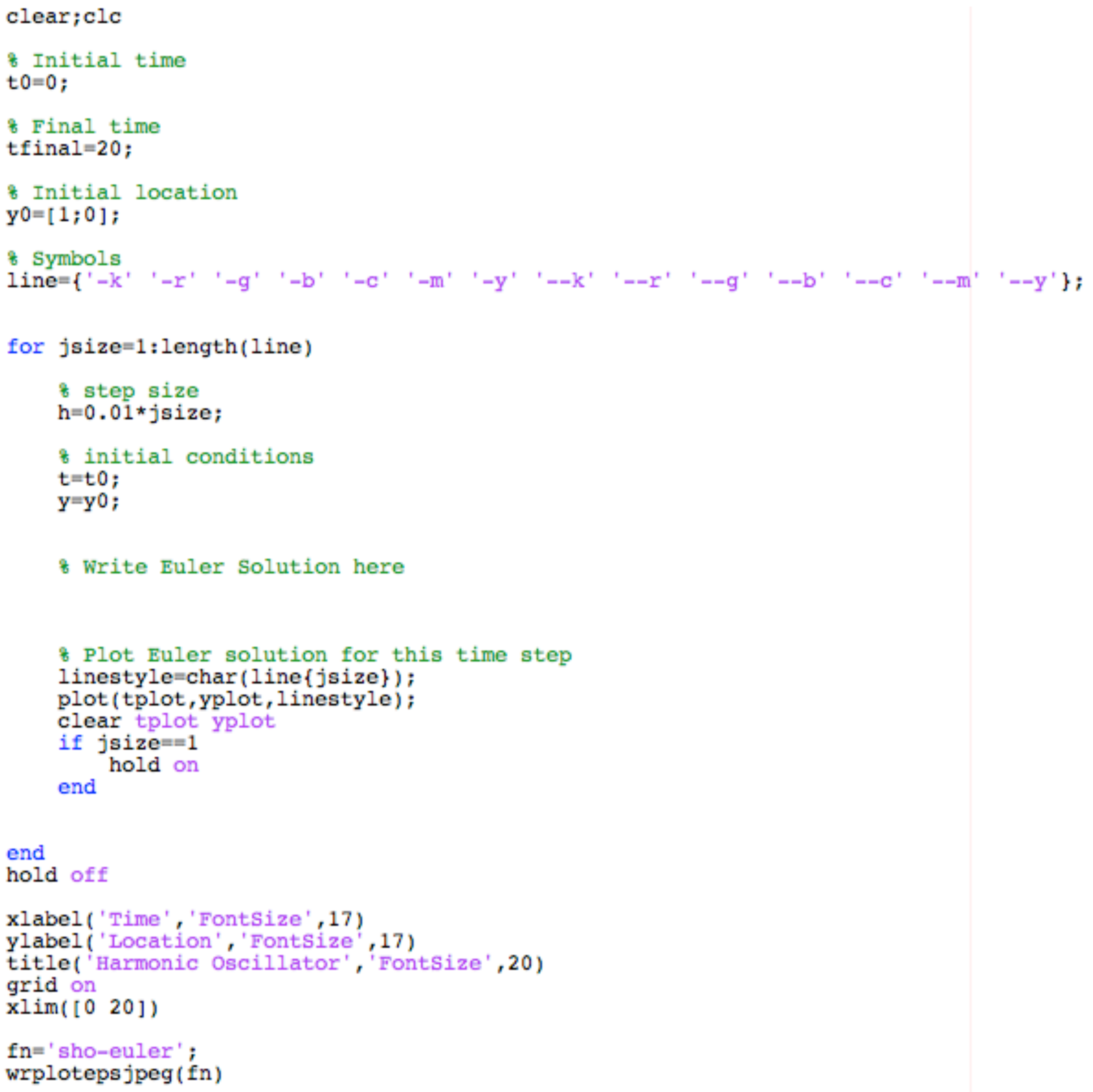


```
% Initial time
t0=0;

% Final time
tfinal=20;

% Initial location
y0=[1;0];

% Symbols
line={'-k' '-r' '-g' '-b' '-c' '-m' '-y' '--k' '--r' '--g' '--b' '--c' '--m' '--y'};
```

Homework 4

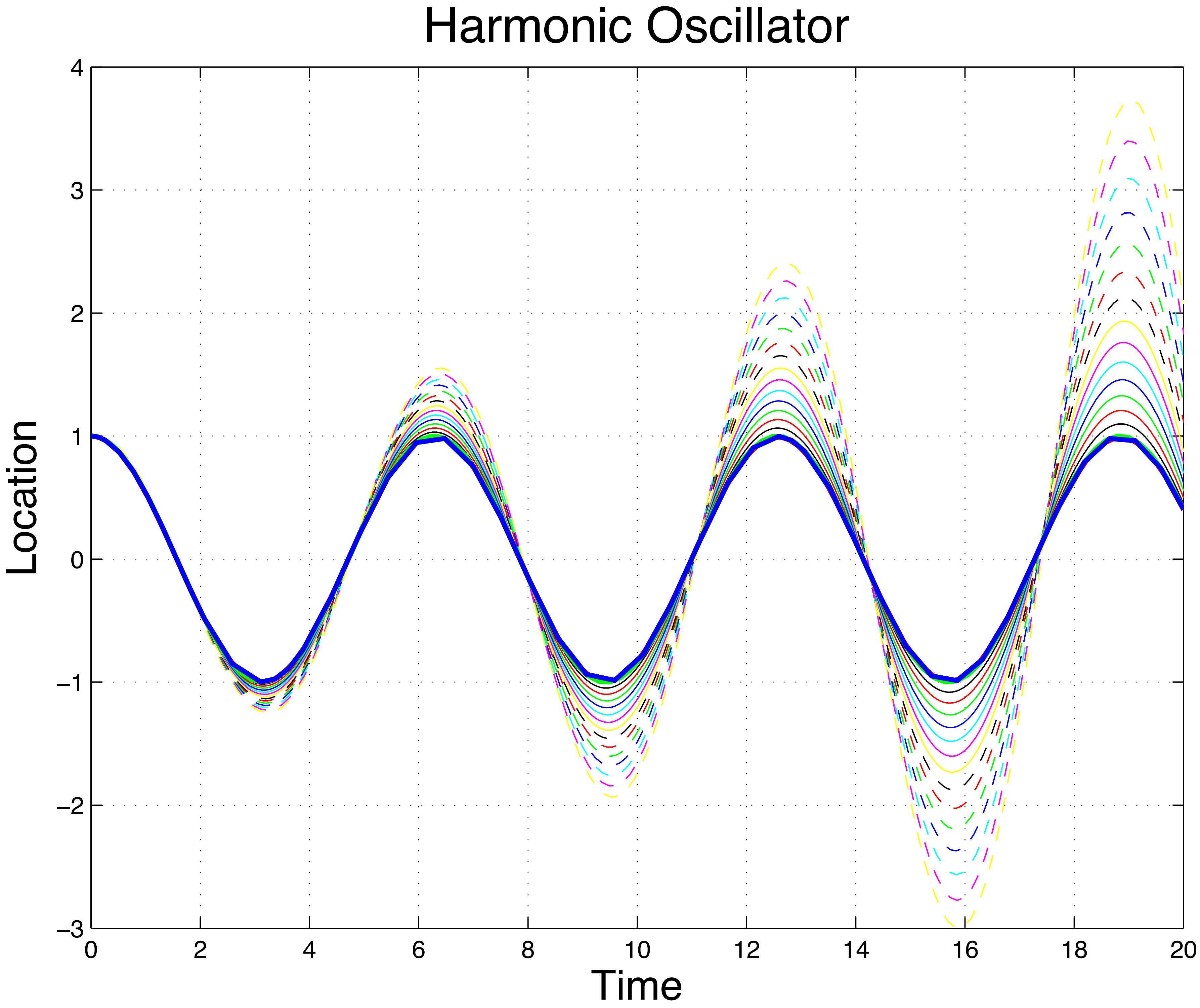
- Solve the same ODEs by calling ODE23, ODE45, ODE113 overlay on the Euler solutions like on the plot to the right.



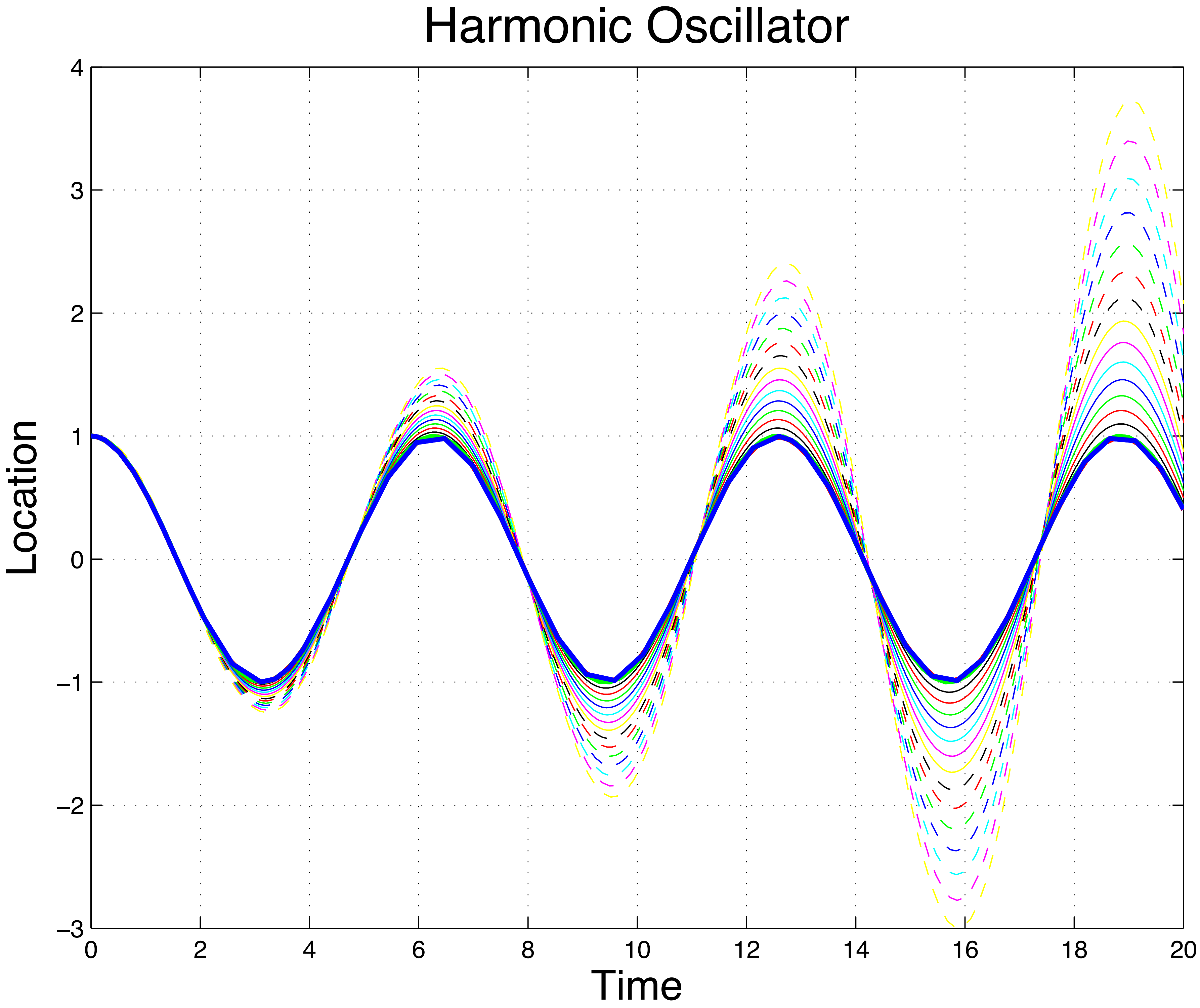


```
clear;clc

% Initial time
t0=0;

% Final time
tfinal=20;

% Initial location
y0=[1;0];

% Symbols
line={'-k' '-r' '-g' '-b' '-c' '-m' '-y' '--k' '--r' '--g' '--b' '--c' '--m' '--y'};


for jsize=1:length(line)

    % step size
    h=0.01*jsize;

    % initial conditions
    t=t0;
    y=y0;


    % Write Euler Solution here



    % Plot Euler solution for this time step
    linestyle=char(line{jsize});
    plot(tplot,yplot,linestyle);
    clear tplot yplot
    if jsize==1
        hold on
    end


end
hold off

xlabel('Time','FontSize',17)
ylabel('Location','FontSize',17)
title('Harmonic Oscillator','FontSize',20)
grid on
xlim([0 20])

fn='sho-euler';
wrplotepsjpeg(fn)
```

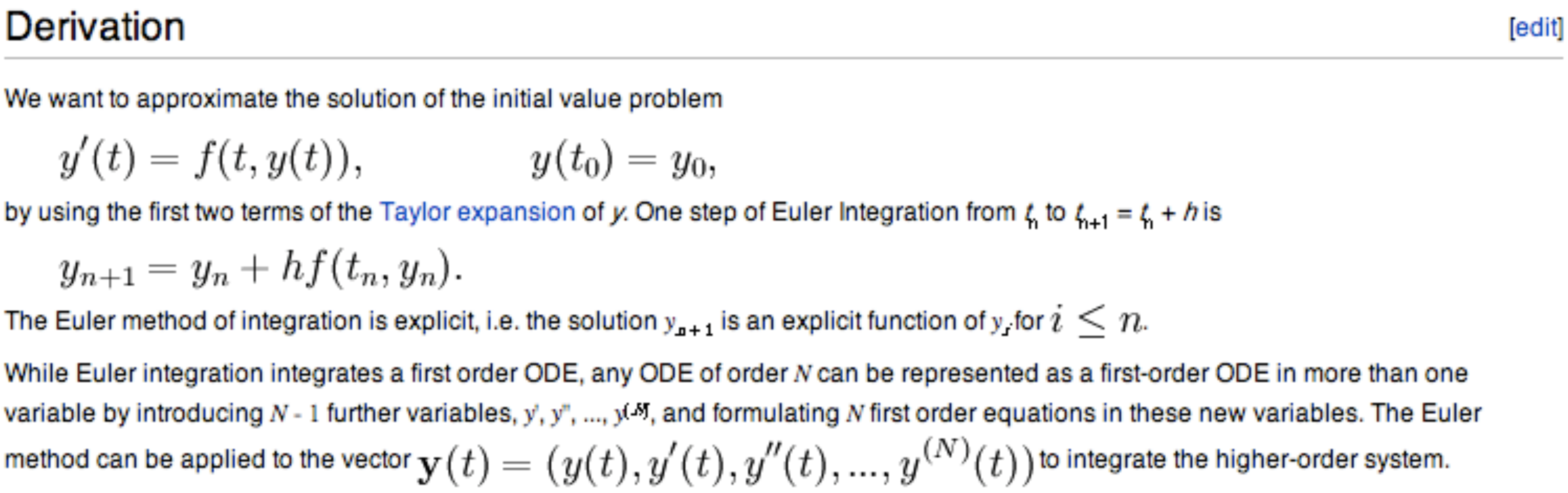
As the step size increased so did the error! Why?



As the step size increased so did the error! Why?



The slope changes over our time step, but in Euler's method we assume it is constant.



Derivation

[edit]

We want to approximate the solution of the initial value problem

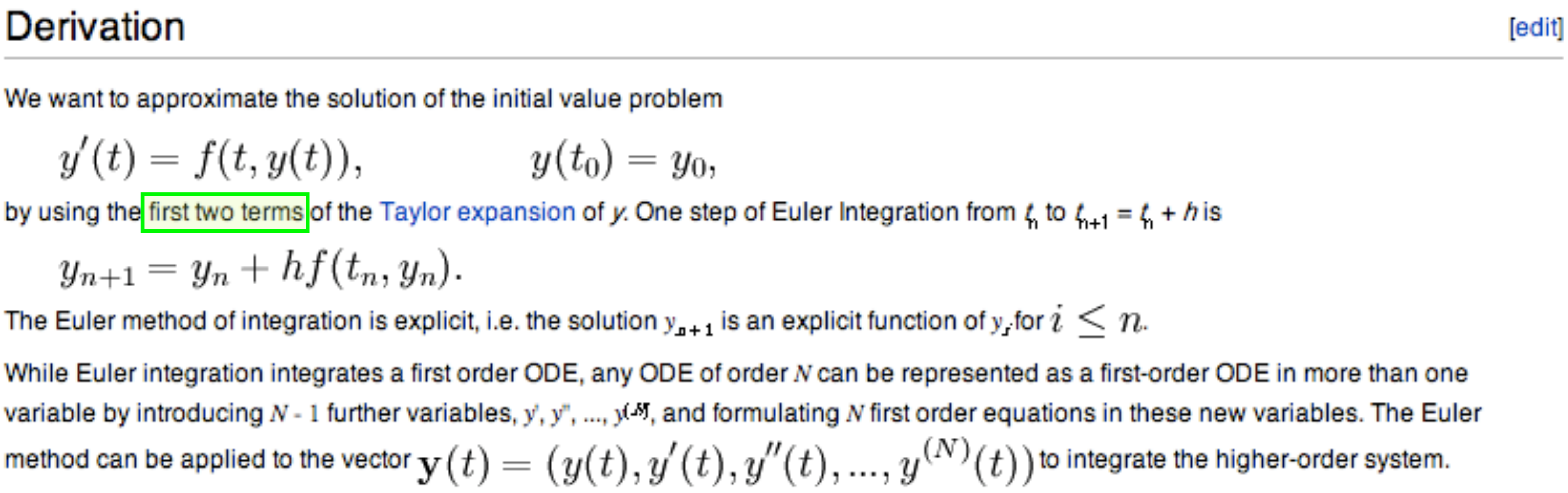
$$y'(t) = f(t, y(t)), \qquad\qquad y(t_0) = y_0,$$

by using the first two terms of the Taylor expansion of y. One step of Euler Integration from t_n to $t_{n+1} = t_n + h$ is

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

The Euler method of integration is explicit, i.e. the solution y_{n+1} is an explicit function of y_i for $i \leq n$.

While Euler integration integrates a first order ODE, any ODE of order N can be represented as a first-order ODE in more than one variable by introducing $N - 1$ further variables, y', y'', ..., $y^{(N)}$, and formulating N first order equations in these new variables. The Euler method can be applied to the vector $\mathbf{y}(t) = (y(t), y'(t), y''(t), ..., y^{(N)}(t))$ to integrate the higher-order system.



Derivation

[edit]

We want to approximate the solution of the initial value problem

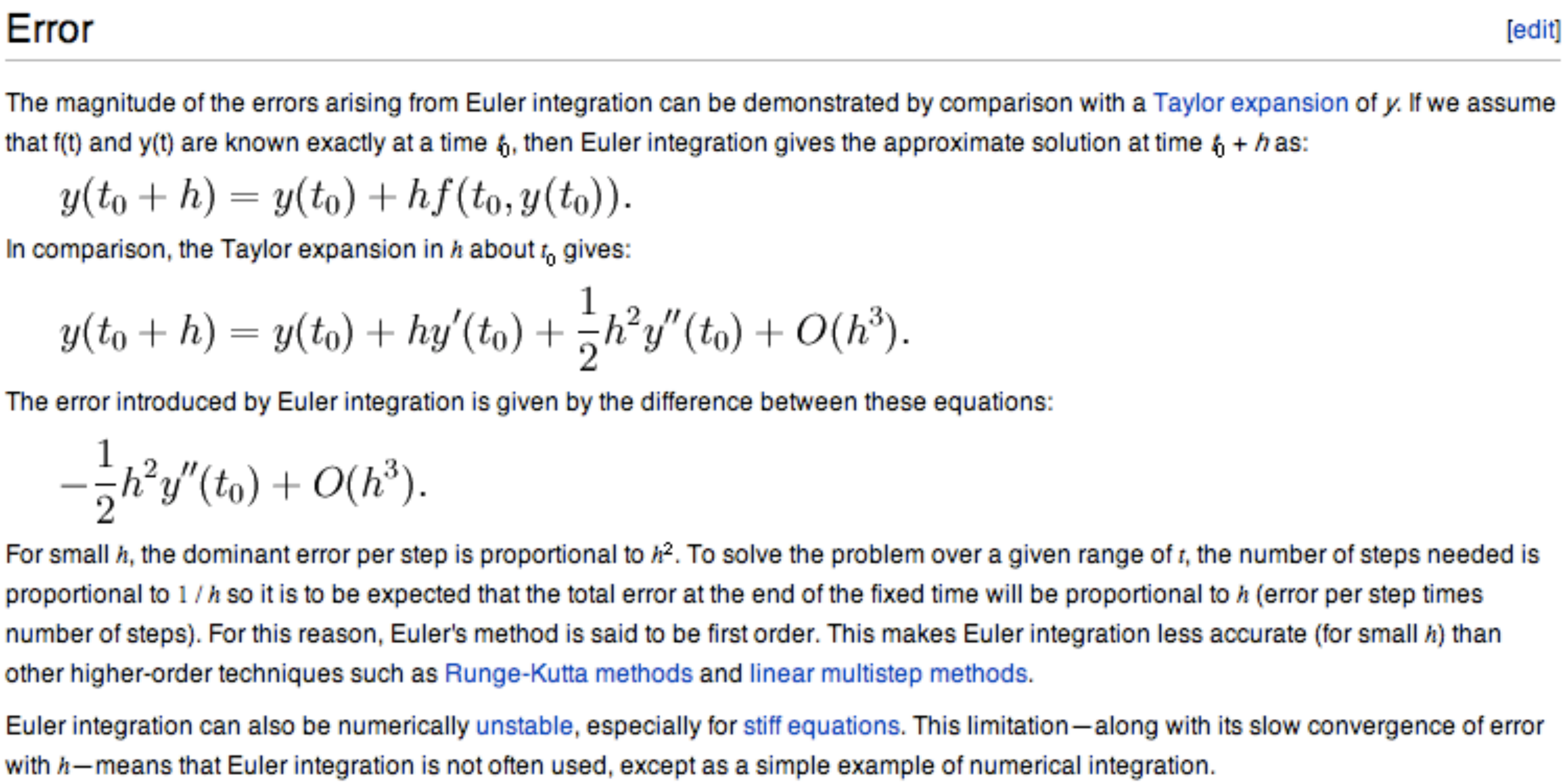
$$y'(t) = f(t, y(t)), \qquad\qquad y(t_0) = y_0,$$

by using the first two terms of the Taylor expansion of y. One step of Euler Integration from t_n to $t_{n+1} = t_n + h$ is

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

The Euler method of integration is explicit, i.e. the solution y_{n+1} is an explicit function of y_i for $i \leq n$.

While Euler integration integrates a first order ODE, any ODE of order N can be represented as a first-order ODE in more than one variable by introducing $N - 1$ further variables, y', y'', ..., $y^{(N)}$, and formulating N first order equations in these new variables. The Euler method can be applied to the vector $\mathbf{y}(t) = (y(t), y'(t), y''(t), ..., y^{(N)}(t))$ to integrate the higher-order system.



Error

[edit]

The magnitude of the errors arising from Euler integration can be demonstrated by comparison with a Taylor expansion of y. If we assume that f(t) and y(t) are known exactly at a time t_0, then Euler integration gives the approximate solution at time $t_0 + h$ as:

$$y(t_0 + h) = y(t_0) + hf(t_0, y(t_0)).$$

In comparison, the Taylor expansion in h about t_0 gives:

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + O(h^3).$$

The error introduced by Euler integration is given by the difference between these equations:

$$-\frac{1}{2}h^2y''(t_0) + O(h^3).$$

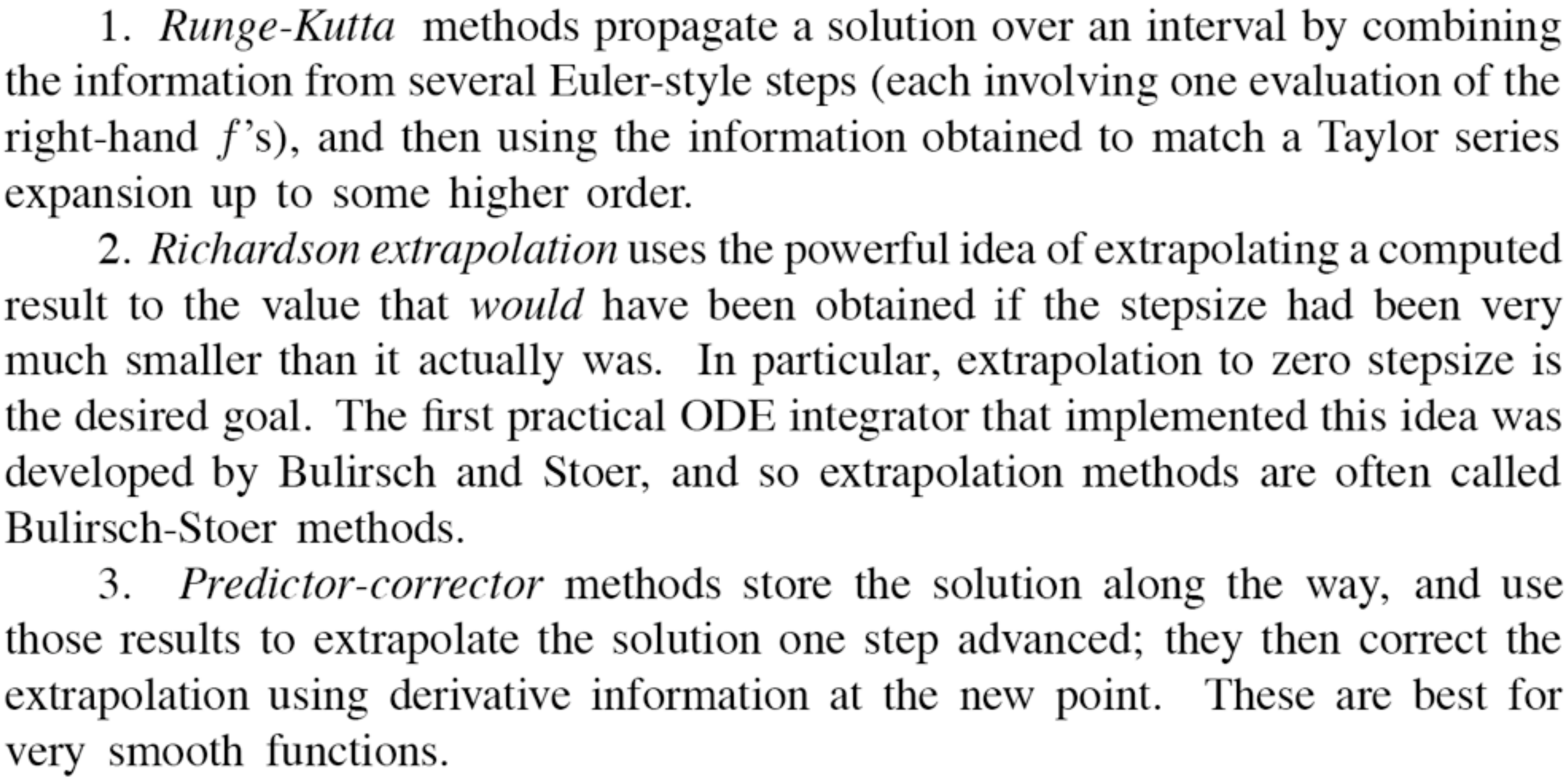
For small h, the dominant error per step is proportional to h^2. To solve the problem over a given range of t, the number of steps needed is proportional to $1 / h$ so it is to be expected that the total error at the end of the fixed time will be proportional to h (error per step times number of steps). For this reason, Euler's method is said to be first order. This makes Euler integration less accurate (for small h) than other higher-order techniques such as Runge-Kutta methods and linear multistep methods.

Euler integration can also be numerically unstable, especially for stiff equations. This limitation—along with its slow convergence of error with h—means that Euler integration is not often used, except as a simple example of numerical integration.

ODE Solvers

- There are three types of solvers:
  - Runge-Kutta Methods
  - Richardson extrapolation and in particular the Burlish-Stoer method
  - Predictor corrector methods

ODE Solvers

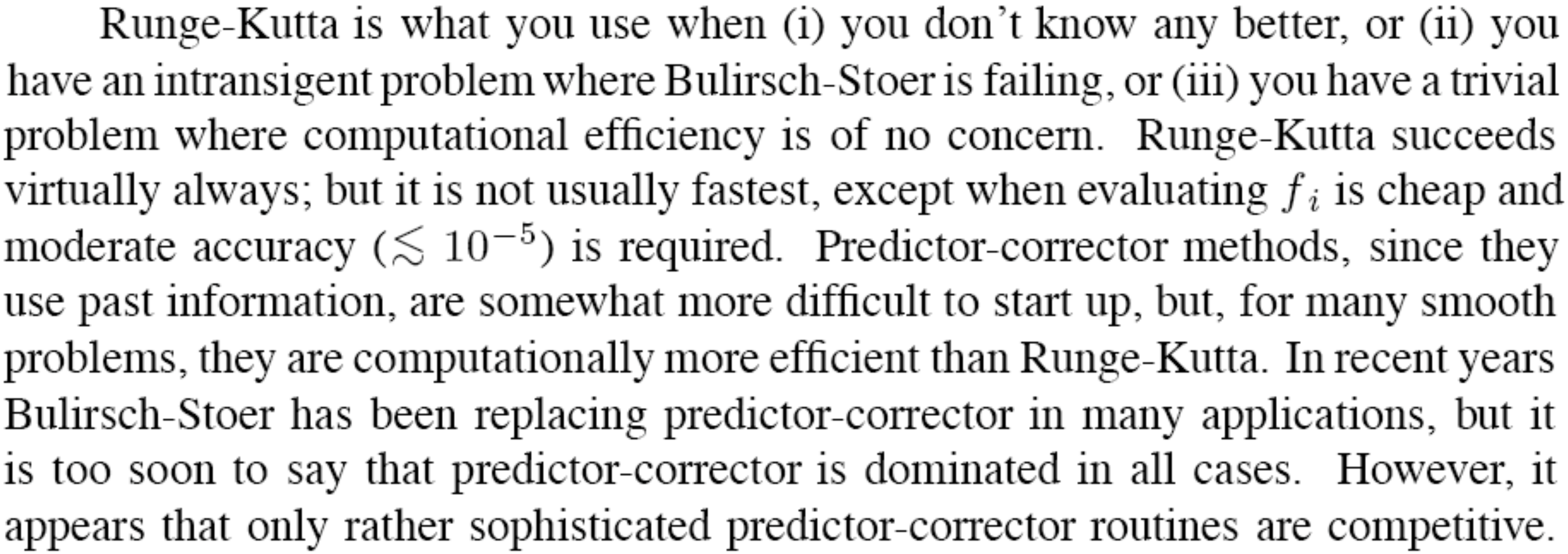


1. *Runge-Kutta* methods propagate a solution over an interval by combining the information from several Euler-style steps (each involving one evaluation of the right-hand f's), and then using the information obtained to match a Taylor series expansion up to some higher order.

2. *Richardson extrapolation* uses the powerful idea of extrapolating a computed result to the value that *would* have been obtained if the stepsize had been very much smaller than it actually was. In particular, extrapolation to zero stepsize is the desired goal. The first practical ODE integrator that implemented this idea was developed by Bulirsch and Stoer, and so extrapolation methods are often called Bulirsch-Stoer methods.

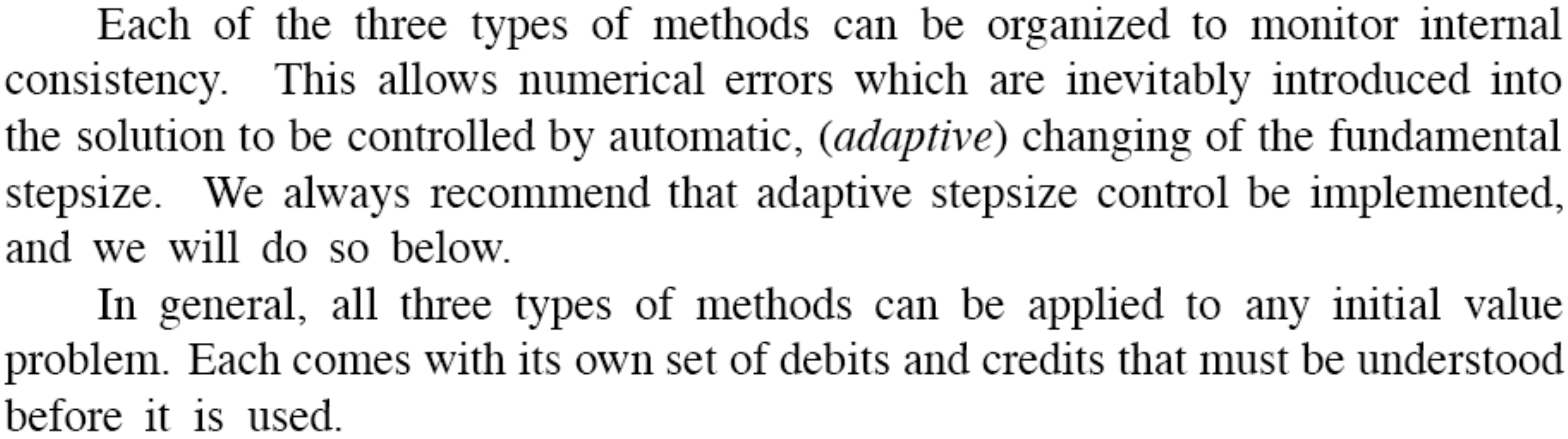
3. *Predictor-corrector* methods store the solution along the way, and use those results to extrapolate the solution one step advanced; they then correct the extrapolation using derivative information at the new point. These are best for very smooth functions.

Which method should I use?



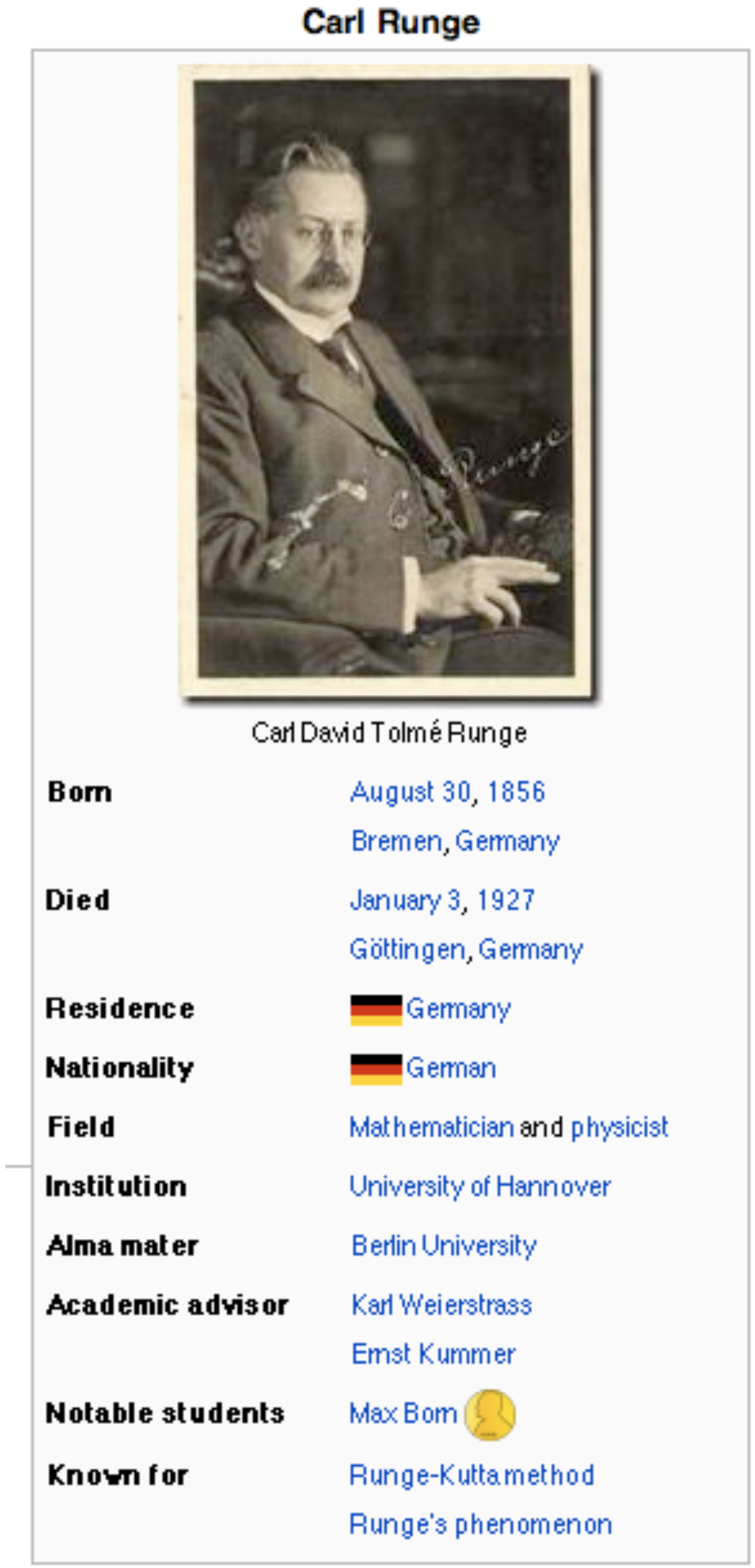
Runge-Kutta is what you use when (i) you don't know any better, or (ii) you have an intransigent problem where Bulirsch-Stoer is failing, or (iii) you have a trivial problem where computational efficiency is of no concern. Runge-Kutta succeeds virtually always; but it is not usually fastest, except when evaluating f_i is cheap and moderate accuracy ($\lesssim 10^{-5}$) is required. Predictor-corrector methods, since they use past information, are somewhat more difficult to start up, but, for many smooth problems, they are computationally more efficient than Runge-Kutta. In recent years Bulirsch-Stoer has been replacing predictor-corrector in many applications, but it is too soon to say that predictor-corrector is dominated in all cases. However, it appears that only rather sophisticated predictor-corrector routines are competitive.

Which method should I use?



Each of the three types of methods can be organized to monitor internal consistency. This allows numerical errors which are inevitably introduced into the solution to be controlled by automatic, (*adaptive*) changing of the fundamental stepsize. We always recommend that adaptive stepsize control be implemented, and we will do so below.

In general, all three types of methods can be applied to any initial value problem. Each comes with its own set of debits and credits that must be understood before it is used.



Carl Runge

Carl David Tolmé Runge

Born	August 30, 1856 Bremen, Germany
Died	January 3, 1927 Göttingen, Germany
Residence	Germany
Nationality	German
Field	Mathematician and physicist
Institution	University of Hannover
Alma mater	Berlin University
Academic advisor	Karl Weierstrass Ernst Kummer
Notable students	Max Born
Known for	Runge-Kutta method Runge's phenomenon

Martin Kutta

Born: 3 Nov 1867 in Pitschen, Upper Silesia (now Byczyna, Poland)
Died: 25 Dec 1944 in Fürstenfeldbruck, Germany

The formula for the Euler method is

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{16.1.1}$$

which advances a solution from x_n to $x_{n+1} \equiv x_n + h$. The formula is unsymmetrical: It advances the solution through an interval h, but uses derivative information only at the beginning of that interval (see Figure 16.1.1). That means (and you can verify by expansion in power series) that the step's error is only one power of h smaller than the correction, i.e $O(h^2)$ added to (16.1.1).

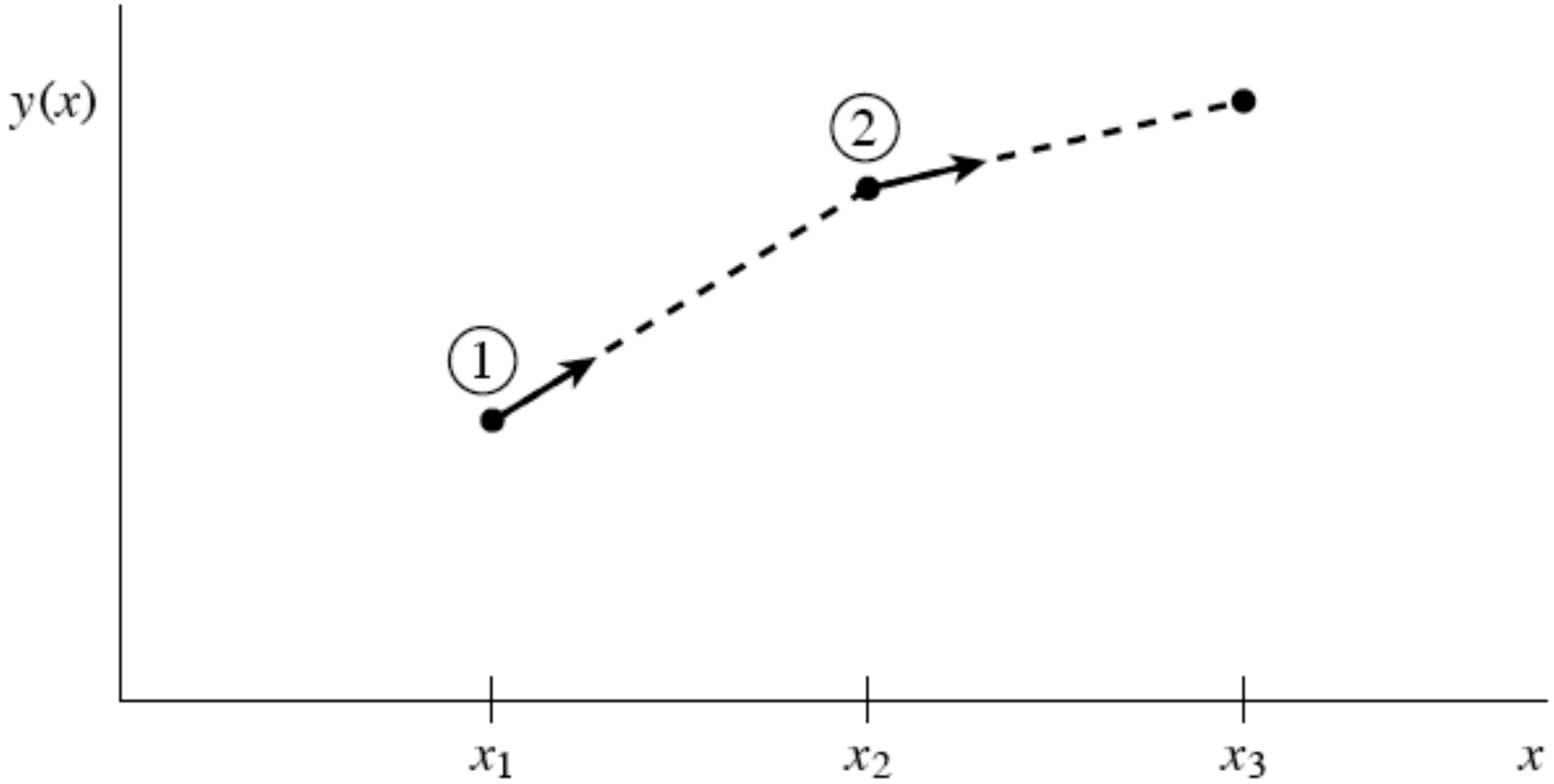


Figure 16.1.1. Euler's method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.

Consider, however, the use of a step like (16.1.1) to take a "trial" step to the midpoint of the interval. Then use the value of both x and y at that midpoint to compute the "real" step across the whole interval. Figure 16.1.2 illustrates the idea. In equations,

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned} \tag{16.1.2}$$

As indicated in the error term, this symmetrization cancels out the first-order error term, making the method *second order*. [A method is conventionally called nth order if its error term is $O(h^{n+1})$.] In fact, (16.1.2) is called the *second-order Runge-Kutta* or *midpoint* method.

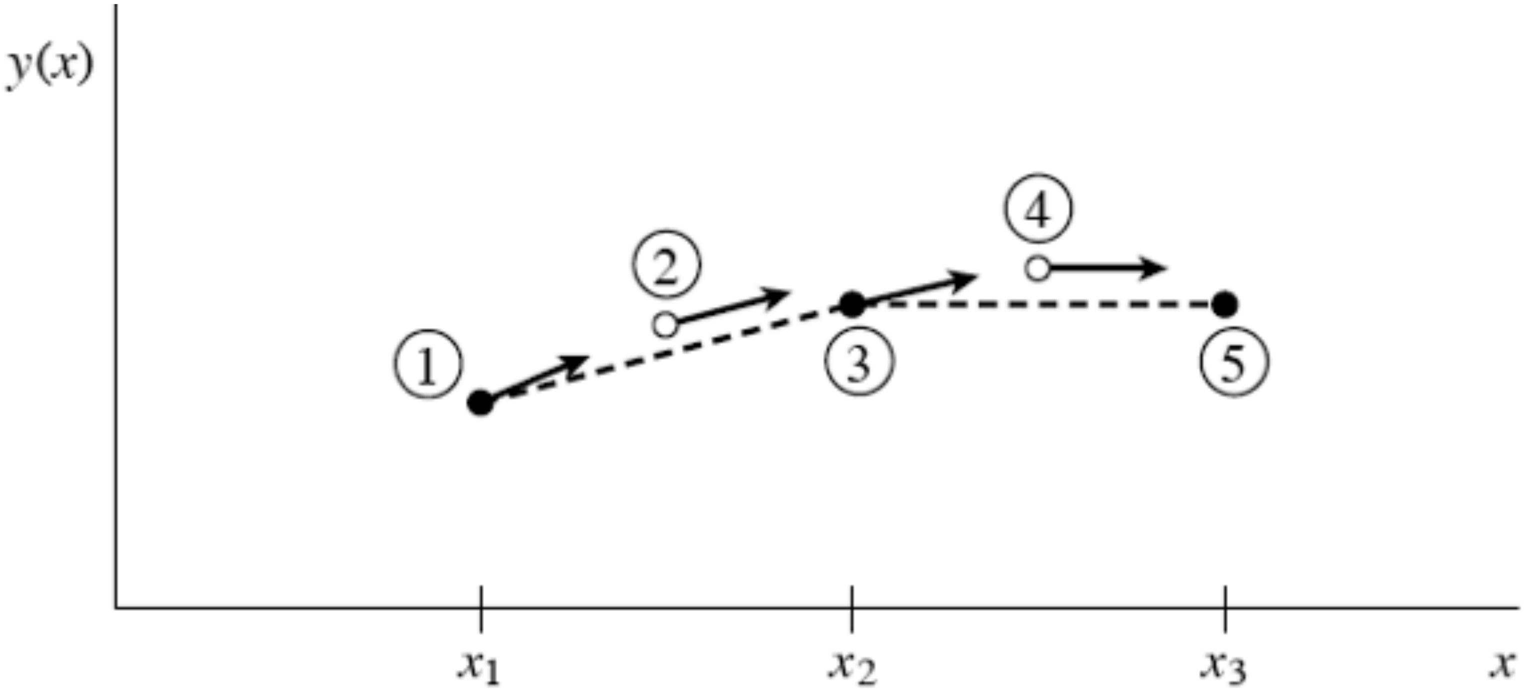


Figure 16.1.2. Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.

By far the most often used is the classical *fourth-order Runge-Kutta formula*, which has a certain sleekness of organization about it:

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \qquad (16.1.3)
\end{aligned}
$$

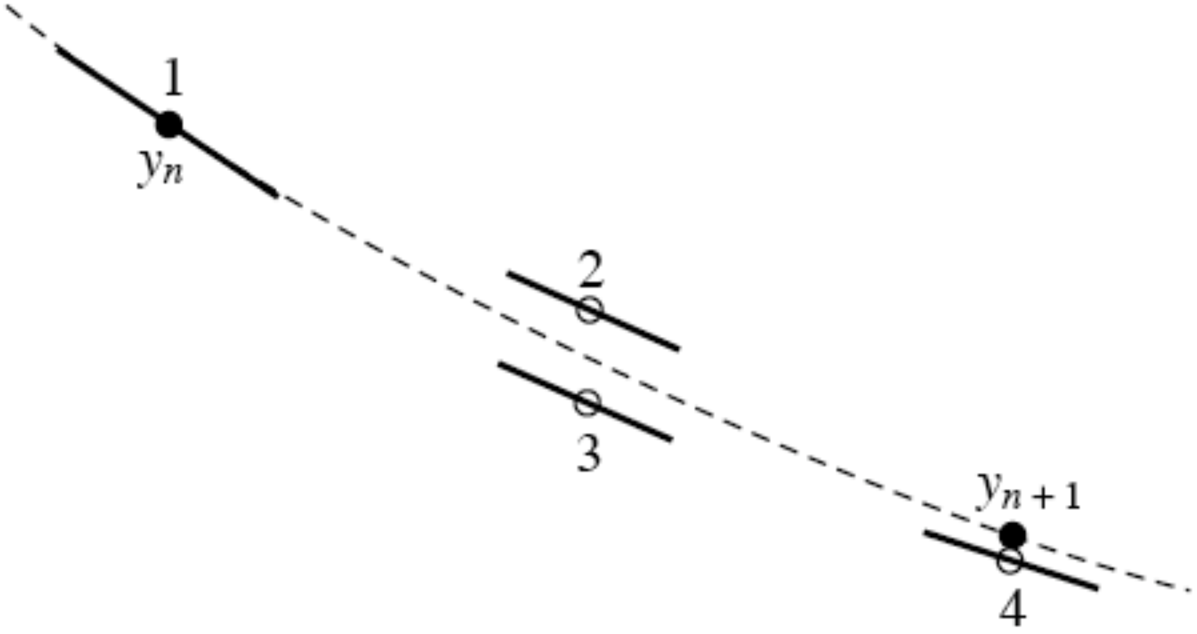


Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

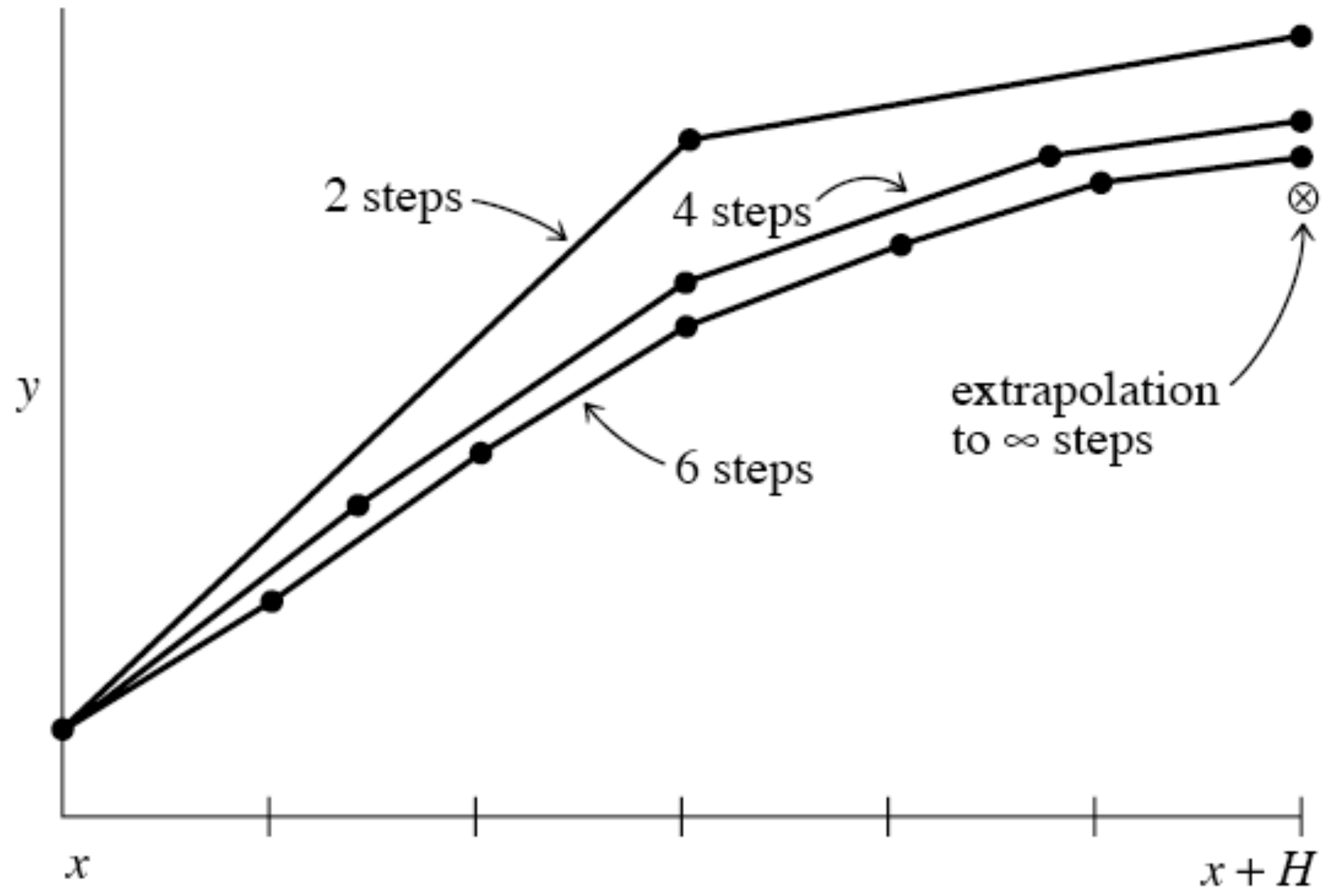
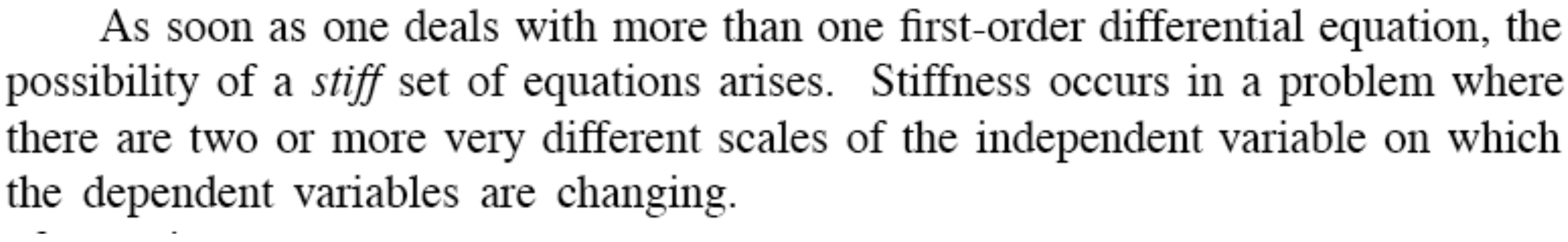
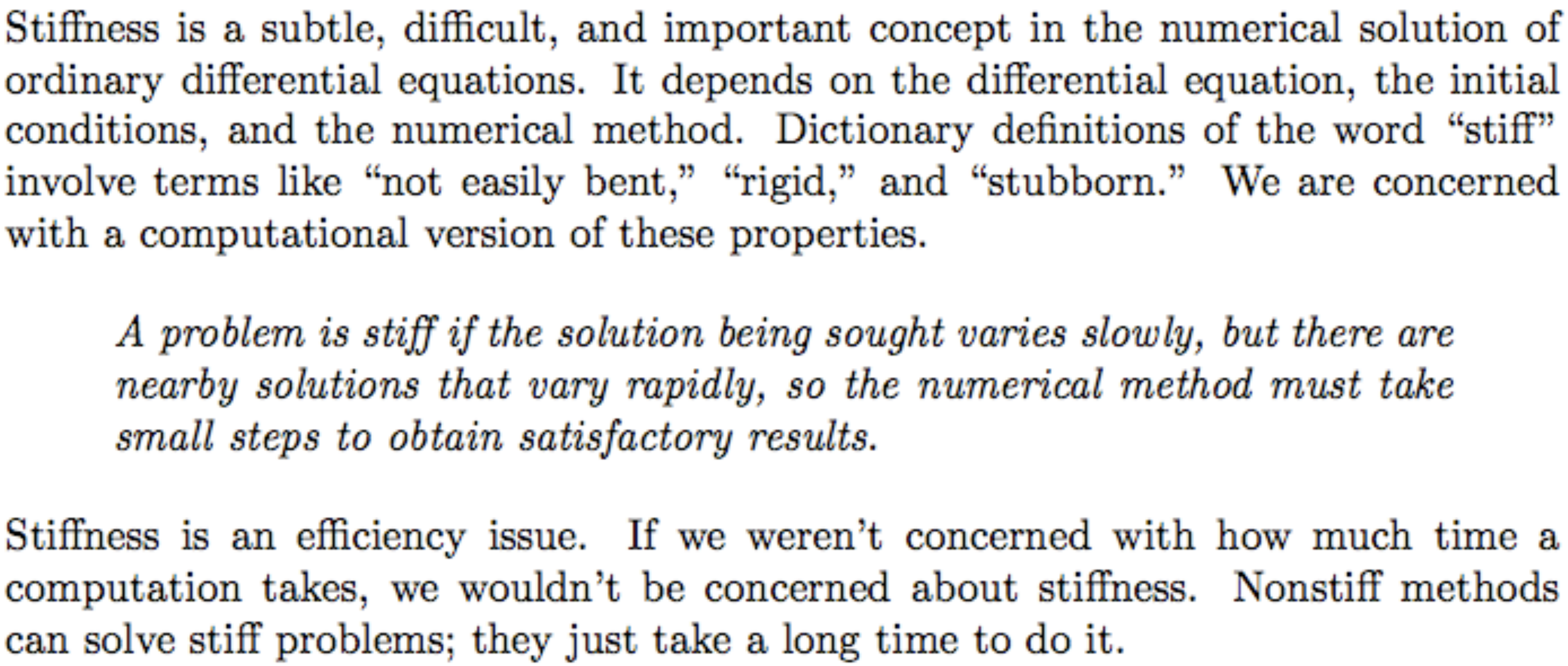


Figure 16.4.1. Richardson extrapolation as used in the Bulirsch-Stoer method. A large interval H is spanned by different sequences of finer and finer substeps. Their results are extrapolated to an answer that is supposed to correspond to infinitely fine substeps. In the Bulirsch-Stoer method, the integrations are done by the modified midpoint method, and the extrapolation technique is rational function or polynomial extrapolation.

Stiff Sets of Equations



As soon as one deals with more than one first-order differential equation, the possibility of a *stiff* set of equations arises. Stiffness occurs in a problem where there are two or more very different scales of the independent variable on which the dependent variables are changing.



Stiffness is a subtle, difficult, and important concept in the numerical solution of ordinary differential equations. It depends on the differential equation, the initial conditions, and the numerical method. Dictionary definitions of the word “stiff” involve terms like “not easily bent,” “rigid,” and “stubborn.” We are concerned with a computational version of these properties.

> *A problem is stiff if the solution being sought varies slowly, but there are nearby solutions that vary rapidly, so the numerical method must take small steps to obtain satisfactory results.*

Stiffness is an efficiency issue. If we weren’t concerned with how much time a computation takes, we wouldn’t be concerned about stiffness. Nonstiff methods can solve stiff problems; they just take a long time to do it.

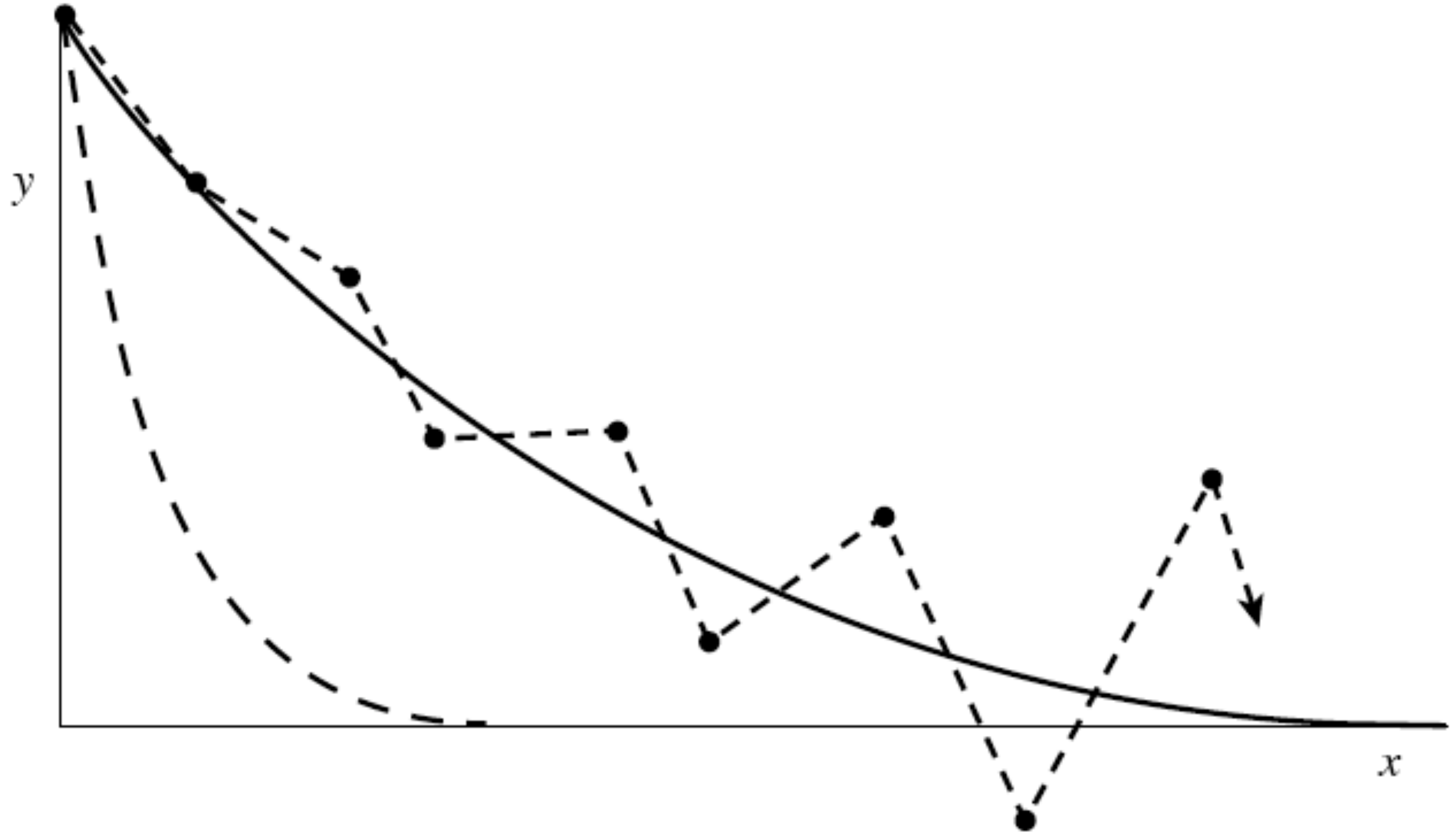


Figure 16.6.1. Example of an instability encountered in integrating a stiff equation (schematic). Here it is supposed that the equation has two solutions, shown as solid and dashed lines. Although the initial conditions are such as to give the solid solution, the stability of the integration (shown as the unstable dotted sequence of segments) is determined by the more rapidly varying dashed solution, even after that solution has effectively died away to zero. Implicit integration methods are the cure.

Exercise

- Look at Matlab help: `doc ode23`
- Do the three examples
- Open the document `odes.pdf` and read about the Lorenz attractor in section 7.8
- Look at the Lorenz attractor example `lorenzgui.m`
- Read about stiff systems in section 7.9
- Read chapter 17 of Numerical Recipes