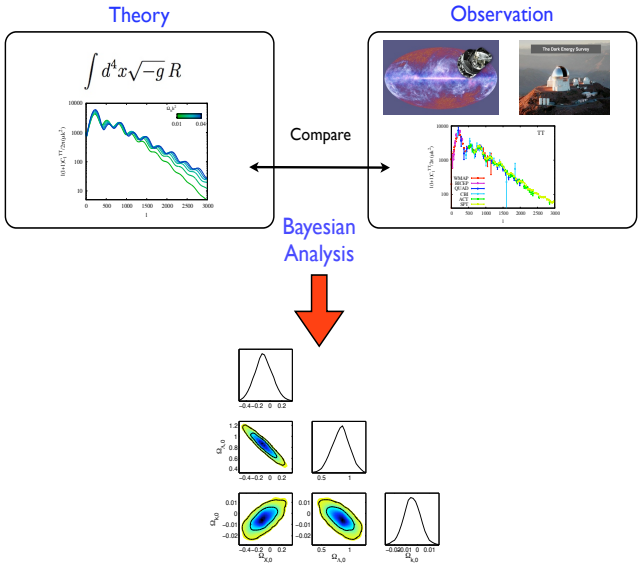
Numerical Methods



José-Alberto Vázquez

ICF-UNAM / Kavli-Cambridge

In progress

August 12, 2021

1

Ordinary Differential Equations

1.0.1 Euler's Method

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0.$$

$$\frac{dy}{dt} = f(t,y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

We make the stipulation that the mesh points are equally distributed throughout the interval $[a,b]$.

$$t_i = a + ih \qquad i = 0, 1, 2, ..., N$$

The common distance between the points $h = (b-a)/N$ is called the step size.
We use Taylor's theorem to derive Euler's method.

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2}y''(\xi_i)$$

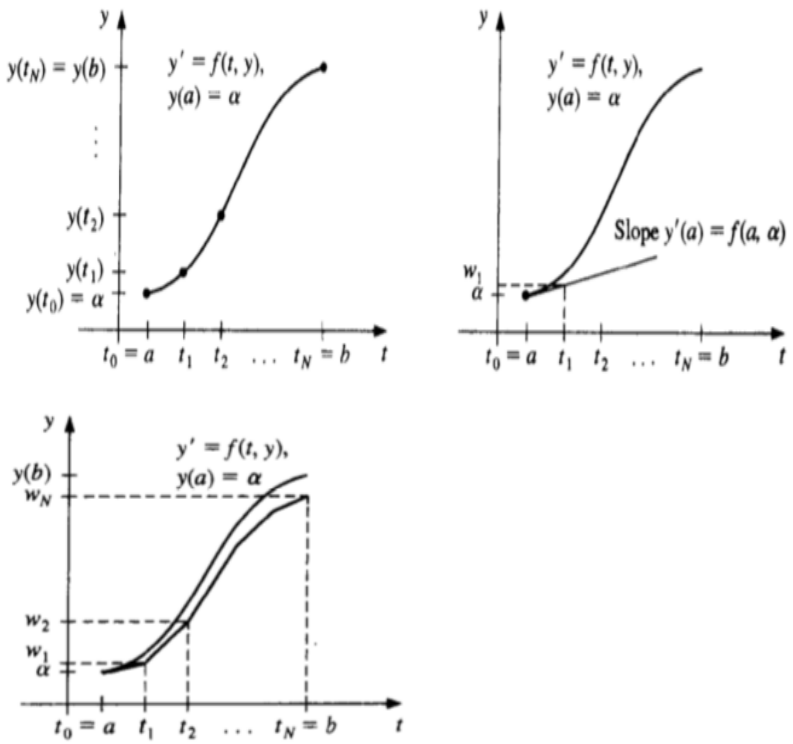
Since $h = t_{i+1} - t_i$ and $y(t)$ satisfies the differential equation

$$y_{t+i} = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}y''(\xi_i)$$

Euler's method constructs $w_i \approx y(t_i)$ by deleting the remainder term.

$$w_0 = \alpha \tag{1.1}$$

$$w_{i+1} = w_i + hf(t_i, w_i), \qquad \text{for } i = 0, 1, ...N-1. \tag{1.2}$$



1.0.2 Extending Euler

Since Euler's method was derived by using Taylor's theorem with $n = 1$, we extend this technique to n.

$$\frac{dy}{dt} = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha$$

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i) + g(\xi)$$

Substituting

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}f'(t_i, y(t_i)) + \cdots + \frac{h^n}{n!}f^{(n-1)}(t_i, y(t_i)) + g(\xi)$$

1.0.3 Taylor method of order n

$$w_0 = \alpha \tag{1.3}$$

$$w_{i+1} = w_i + hT^{(n)}(t_i, w_i) \tag{1.4}$$

where

$$T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}f'(t_i, w_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, w_i)$$

If Taylor's method of order n is used to approximate the solution to

$$y'(t) = f(t, y(t)), \quad a \le t \le b, \quad y(a) = \alpha,$$

with step size h and if $y \in C^{n+1}[a, b]$, then the local truncation error is $O(h^n)$.

1.1 Runge-Kutta Methods

The disadvantage of Taylor is requiring the computation and evaluation of the derivatives of $f(t, y)$. Runge-Kutta methods have the high-order local truncation error while eliminating the need to compute and evaluate derivatives of $f(t, y)$.

$$f(t, y) = P_n(t, y) + R_n(t, y),$$

where

$$\begin{aligned} P_n(t, y) = f(t_0, y_0) &+ \left[(t - t_0)\frac{\partial f}{\partial t}(t_0, y_0) + (y - y_0)\frac{\partial f}{\partial y}(t_0, y_0)\right] \\ &+ \left[\frac{(t - t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0, y_0) + (t - t_0)(y - y_0)\frac{\partial^2 f}{\partial t \partial y}(t_0, y_0)\right. \\ &\left. + \frac{(y - y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0, y_0)\right] + \cdots \\ &+ \left[\frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}(t - t_0)^{n-j}(y - y_0)^j\frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0, y_0)\right] \end{aligned}$$

and

$$R_n(t, y) = \frac{1}{(n+1)!}\sum_{j=0}^{n+1}\binom{n+1}{j}(t - t_0)^{n+1-j}(y - y_0)^j\frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi, \mu).$$

The first step in deriving a Runge-Kutta is to determine $a_1 f(t+\alpha_1, y+\beta_1)$ that approximates

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y)$$

Since,

$$f'(t,y) = \frac{df}{dt}(t,y) = \frac{\partial f}{\partial t}(t,y) + \frac{\partial f}{\partial y}(t,y)y'(t) \quad \text{and} \quad y'(t) = f(t,y)$$

this implies

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t}(t,y) + \frac{h}{2}\frac{\partial f}{\partial y}(t,y)f(t,y)$$

Expanding $f(t+\alpha_1, y+\beta_1)$ in Taylor

$$a_1 f(t+\alpha_1, y+\beta_1) = a_1 f(t,y) + a_1\alpha_1 \frac{\partial f}{\partial t}(t,y) + a_1\beta_1 \frac{\partial f}{\partial y}(t,y) + a_1 R_1 \tag{1.5}$$

Matching the coefficients of f, gives

$$a_1 = 1, \quad a_1\alpha_1 = \frac{h}{2}, \quad a_1\beta_1 = \frac{h}{2} f(t,y)$$

Therefore

$$a_1 = 1, \quad \alpha_1 = \frac{h}{2}, \quad \beta_1 = \frac{h}{2} f(t,y)$$

So,

$$T^{(2)}(t,y) = f\left(t + \frac{h}{2}, y + \frac{h}{2} f(t,y)\right) - R_1(\mathcal{O}(h^2))$$

1.1.1 Midpoint Method

$$w_0 = \alpha \tag{1.6}$$

$$w_{i+1} = w_i + hf\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right) \tag{1.7}$$

The most appropriate four-parameter form for approximating

$$T^{(3)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y) + \frac{h^2}{6} f''(t,y)$$

is

$$a_1 f(t,y) + a_2 f(t+\alpha_2, y+\delta_2 f(t,y))$$

1.1.2 Modified Euler Method

Given the flexibility of choosing parameters, $a_1 = a_2 = \frac{1}{2}$ and $\alpha_2 = \delta_2 = h$

$$w_0 = \alpha \tag{1.8}$$
$$w_{i+1} = w_i + \frac{h}{2}[f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))] \tag{1.9}$$

1.1.3 Heun's Method

$a_1 = \frac{1}{4}, a_2 = \frac{3}{4}$ and $\alpha_2 = \delta_2 = \frac{2}{3}h$

$$w_0 = \alpha \tag{1.10}$$
$$w_{i+1} = w_i + \frac{h}{4}[f(t_i, w_i) + 3f(t_i + \frac{2}{3}h, w_i + \frac{2}{3}hf(t_i, w_i))] \tag{1.11}$$

Both are classified as Runge-Kutta methods of order two.

1.1.4 RK-4

$$w_0 = \alpha,$$
$$k_1 = hf(t_i, w_i),$$
$$k_2 = hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1\right),$$
$$k_3 = hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2\right),$$
$$k_4 = hf(t_{i+1}, w_i + k_3),$$
$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

Figure 1.1: fig1

1.2 Adaptative Runge-Kutta Methods

Methods that have been presented so far employ a constant step size. For a significant number of problems, this can represent a serious limitation.

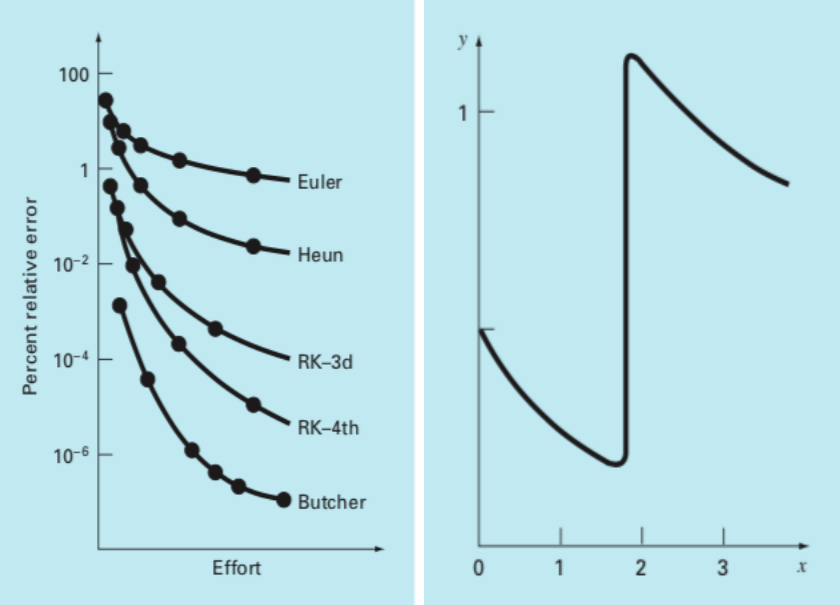


Figure 1.2: figure

Adaptive methods that adjust step size can give great advantage because they 'adapt' to the solution's trajectory. Implementation of such approaches requires that an estimate of the local truncation error (LTE) be obtained at each step, which serves as a basis for adjusting step size.

(Remember the Richardson interpolation). The local truncation error (LTE) is estimated as the difference between two predictions using different- order Runge-Kutta methods.

$$q = \left(\frac{\varepsilon h}{2|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/4} = 0.84\left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/4}.$$

$$\tilde{w}_{i+1} = w_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6,$$

to estimate the local error in a Runge-Kutta method of order four given by

$$w_{i+1} = w_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5,$$

where

$$k_1 = hf(t_i, w_i),$$
$$k_2 = hf\left(t_i + \frac{h}{4}, w_i + \frac{1}{4}k_1\right),$$

Figure 1.3: fig 2

$$k_3 = hf\left(t_i + \frac{3h}{8}, w_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right),$$
$$k_4 = hf\left(t_i + \frac{12h}{13}, w_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right),$$
$$k_5 = hf\left(t_i + h, w_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right),$$
$$k_6 = hf\left(t_i + \frac{h}{2}, w_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right).$$

1.3 Multi-step methods

The methods of Euler, Heun, and Runge-Kutta that have been presented so far are called single-step methods, because they use only the information from one previous point to compute the successive point After several points have been found, it is feasible to use several prior points in the calculation. This is the basis of multistep methods.

A desirable feature of multistep methods is that the local truncation error (LTE) can be determined and a correction term can be included, which improves the accuracy of the answer at each step. Also, it is possible to determine if the step size is small enough to obtain an accurate value for y_{i+1}, yet large enough so that unnecessary and time-consuming calculations are eliminated.

1.3.1 Derivation

Integrate

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

fron x_i to x_{i+2} to get

$$\begin{aligned} \int_{x_i}^{x_{i+2}} y'(x)dx &= \int_{x_i}^{x_{i+2}} f(x,y)dx && (1.12)\\ y(x_{i+2}) - y(x_i) &= && (1.13)\\ y_{i+2} - y_i &= && (1.14) \end{aligned}$$

Now the step-size $h = x_{i+2} - x_{i+1} = x_{i+1} - x_i$, gives $x_{i+2} - x_i = 2(x_{i+1} - x_i) = 2h$.

if we approximate the integral by Simpson's 1/3 rule

$$\int_{x_i}^{x_{i+2}} f(x,y)dx = \frac{x_{i+2} - x_i}{6}(f(x_i, y_i) + 4f(x_{i+1}, y_{i+1}) + f(x_{i+2}, y_{i+2})).$$

This is a two-step method rather than a one-step method.

1.3.2 General Form of Linear Multi-step Methods (LMMs)

Given a sequence of equally spaced step levels $x_i = x_0 + ih$ with step size h, the general k-step LMM can be written as

$$\sum_{j=0}^{k} \alpha_j y_{i+j} = h \sum_{j=0}^{k} \beta_j f_{i+j}$$

with $f_i \equiv f(x_i, y_i)$, and parameters α_j and β_j. Given the approximate solution up to x_{i+k-1} we obtain the approximate solution y_{i+k} at the new step level x_{i+k}

$$\alpha_k\, y_{i+k} = \beta_k\, f(x_{i+k}, y_{i+k}) - \underbrace{\sum_{j=0}^{k-1} \alpha_j\, y_{i+j} + h \sum_{j=0}^{k-1} \beta_j\, f_{i+j}}_{\text{all previously computed}}$$

If $\beta_k = 0$ then the scheme is **explicit** since y_{i+k} can be evaluated directly without the need to solve y_{i+k}. If $\beta_k \neq 0$ the scheme is **implicit** since we need to solve y_{i+k} each step.
Note that to get started, the k-step LMM needs to the first k step levels of the approximate solution, $y_0, y_1, ...y_{k-1}$ to be specified. The ODE IVPs only give so something extra has to be done.

1.3.3 Adams-Bashforth Formulas

Rewrite a forward Taylor series expansion

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2!} f_i' + \frac{h^3}{3!} f_i'' + \cdots$$

and a 2nd-order backward expansion

$$f_{i-1} = f_i - hf_i' + \frac{h^2}{2!} f_i'' + R_3$$

can be used to approximate the derivative, and we get the 2nd-order Adams-Bashforth formula

$$y_{i+1} = y_i + \frac{3h}{2} f_i - \frac{h}{2!} f_{i-1} + \frac{5h^3}{12} f_i'' + \mathcal{O}(h^4)$$

Higher-order Adams-Bashforth formulas can be developed by substituting higher-difference approximations into eqn

$$y_{i+1} = y_i + h \sum_{k=0}^{n-1} \beta_k f_{i-k} + \mathcal{O}(h^{n+1})$$

Using the same technique as Adams-Bashforth (but using the backward Taylor series expansion around x_{i+1})

Order	β_0	β_1	β_2	β_3	β_4	β_5
1	1					
2	3/2	-1/2				
3	23/12	-16/12	5/12			
4	55/24	-59/24	37/24	-9/24		
5	1901/720	-2774/720	2616/720	-1274/720	251/720	
6	4277/720	-7923/720	9982/720	-7298/720	2877/720	-475/720

$$y_{i+1} = y_i + \frac{h}{2}(f_i + f_{i+1}) - \frac{h^3}{12}f_i'' + \mathcal{O}(h^4)$$

and in general

$$y_{i+1} = y_i + h\sum_{k=0}^{n-1}\beta_k f_{i+1-k} + \mathcal{O}(h^{n+1})$$

1.3.4 Milne's method

Milne's method is based on Newton-Cotes integration formulas and uses the three-point Newton-Cotes open formula as a predictor

$$y_{i+1}^0 = y_{i-3}^m + \frac{4h}{3}(2f_i^m - f_{i-1}^m + 2f_{i-2}^m)$$

and the three-point Newton-Cotes closed formula (Simpson's 1/3 rule) as a corrector

$$y_{i+1}^j = y_{i-1}^m + \frac{h}{3}(f_{i-1}^m + 4f_i^m + f_{i+1}^{j-1})$$

where j is an index representing the number of iterations of the modifier.

Order	β_0	β_1	β_2	β_3	β_4	β_5
2	1/2	1/2				
3	5/12	8/12	-1/12			
4	9/24	19/24	-5/24	1/24		
5	251/720	646/720	-264/720	106/720	-19/720	
6	475/1440	1427/1440	-798/1440	482/1440	-173/1440	27/1440

Adams-Bashforth-Moulton Method

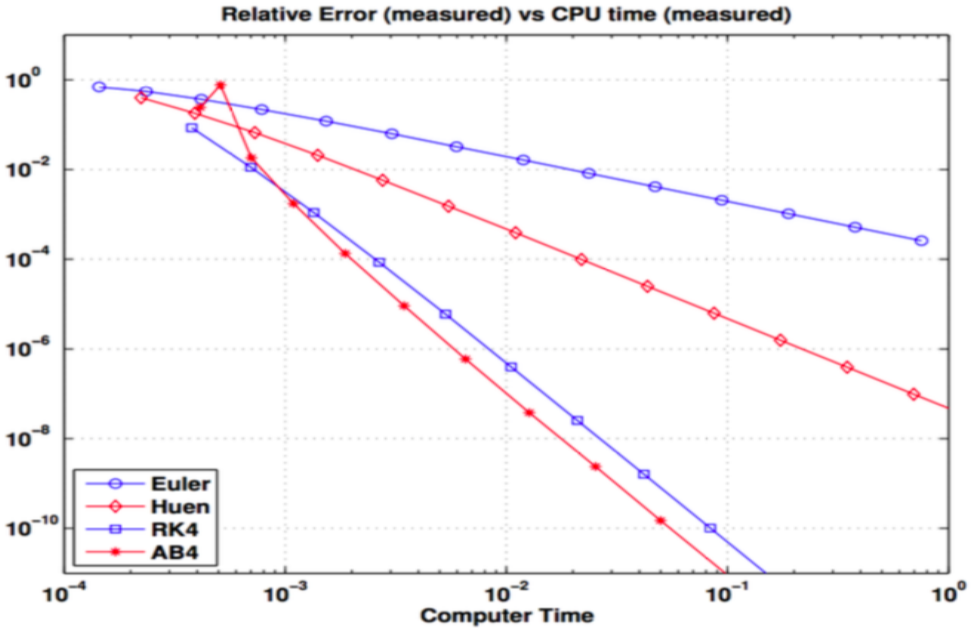
This method is a popular multistep method that uses the 4th-order Adams-Bashforth formula as the predictor

$$y_{i+1}^0 = y_i^m + \frac{h}{24}\left(55f_i^m - 59f_{i-1}^m + 37f_{i-2}^m - 9f_{i-3}^m\right) \quad \textbf{(10.20)}$$

and the 4th-order Adams-Moulton formula as the corrector

$$y_{i+1}^j = y_i^m + \frac{h}{24}\left(9f_{i+1}^{j-1} + 19f_i^m - 5f_{i-1}^m + f_{i-2}^m\right) \quad \textbf{(10.21)}$$

1.4 Variable Step-Size Multistep Methods



1.5 Higher-Order Equation and Systems of Differential Equations

An mth-order system of first-order initial-value problem has the form

$$\frac{du_1}{dt} = f_1(t, u_1, u_2, \ldots, u_m), \tag{1.15}$$

$$\frac{du_2}{dt} = f_2(t, u_1, u_2, \ldots, u_m), \tag{1.16}$$

$$. \tag{1.17}$$

$$\frac{du_m}{dt} = f_m(t, u_1, u_2, \ldots, u_m). \tag{1.18}$$

$$\tag{1.19}$$

for $a \leq t \leq b$, with the initial conditions

$$u_1(a) = \alpha_1, u_2(a) = \alpha_2, \ldots, u_m(a) = \alpha_m$$

Let an integer $N > 0$ be chosen and set $h = (b - a)/N$. Partition the interval $[a, b]$ into N subintervals with the mesh points

$$t_j = a + jh, \qquad \text{for each } j = 0, 1, \ldots, N.$$

For the initial conditions, set

$$w_{1,0} = \alpha_1, w_{2,0} = \alpha_2, \ldots, w_{m,0} = \alpha_m$$

Suppose that the values $w_{1,j}, w_{2,j}, \ldots, w_{m,j}$ have been computed. We obtain $w_{1,j+1}, w_{2,j+1}, \ldots, w_{m,j+1}$ by first calculating

$$k_{1,i} = hf_i(t_j, w_{1,j}, w_{2,j}, \ldots, w_{m,k}) \quad \text{for each } i = 1, 2, \ldots, m \tag{1.20}$$

$$k_{2,i} = hf_i\left(t_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{1,1}, w_{2,j} + \frac{1}{2}k_{1,2}, \ldots, w_{m,k} + \frac{1}{2}k_{1,m}\right) \tag{1.21}$$

$$k_{3,i} = hf_i\left(t_j + \frac{h}{2}, w_{1,j} + \frac{1}{2}k_{2,1}, w_{2,j} + \frac{1}{2}k_{2,2}, \ldots, w_{m,k} + \frac{1}{2}k_{2,m}\right) \tag{1.22}$$

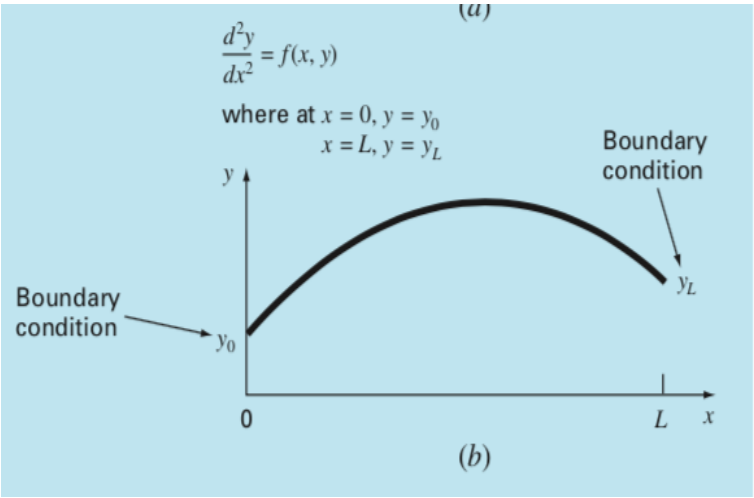
$$k_{4,i} = hf_i\left(t_j + h, w_{1,j} + k_{3,1}, w_{2,j} + k_{3,2}, \ldots, w_{m,k} + k_{3,m}\right) \tag{1.23}$$

$$\tag{1.24}$$

and then

$$w_{i,j+1} = w_{i,j} + \frac{1}{6}(k_{1,j} + 2k_{2,i} + 2k_{3,i} + k_{4,i})$$

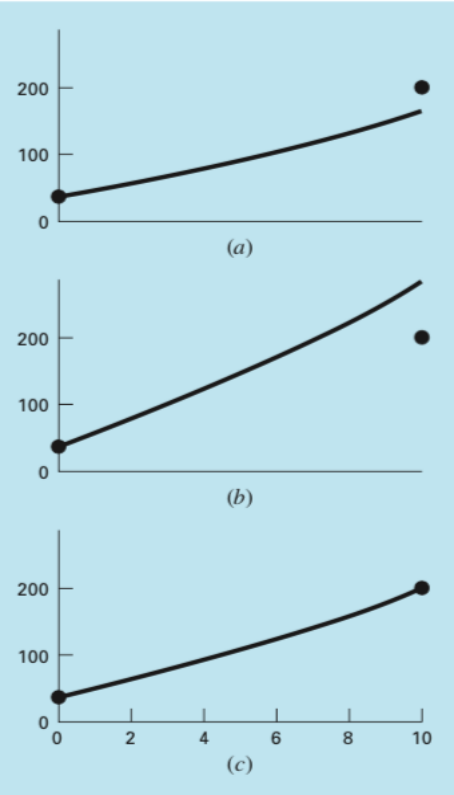
for each $i = 1, 2, \ldots, m$.



(py: Double-pendulum, Lorentz)

1.6 The shooting Method

The shooting method is based on converting the boundary-value problem into an equivalent initial-value problem. A trial-and-error approach is then implemented to solve the initial-value version.


200
100
0
(a)
200
100
0
(b)
200
100
0
0
2
4
6
8
10
(c)