3.2 Initial Value Problems

Many problems in applied mathematics lead to Ordinary Differential Equations (ODEs). In the simplest case one seeks a differentiable function $y = y(x)$ of one real variable x, whose derivative $y'(x)$ is to satisfy an equation of the form $y'(x) = f(x, y(x))$, or more briefly,

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

one then speaks of an ordinary differential equation of first order. In general there are infinitely many different functions y which are solutions of an ODE. Through additional requirements one can single out certain solution from the set of all solutions. Thus, in an Initial-Value Problem, one seeks a solution y of an ODE which for given (x_0, y_0) satisfies an Initial Condition of the form

$$y(x_0) = y_0 \,.$$

System of ODEs

More generally, one also considers system of n ordinary differential equations

$$\begin{cases} y_1'(x) = f_1(x, y_1(x), y_2(x), \ldots, y_n(x)) \\ y_2'(x) = f_2(x, y_1(x), y_2(x), \ldots, y_n(x)) \\ \qquad\qquad\vdots \\ y_n'(x) = f_n(x, y_1(x), y_2(x), \ldots, y_n(x)) \end{cases}$$

with initial conditions

$$y_1(x_0) = y_{10}, \quad y_2(x_0) = y_{20}, \quad \ldots \quad y_n(x_0) = y_{n0} \,.$$

High order ODEs

There are ODEs of m^{th} order of the form

$$y^{(m)}(x) = f(x,\ y(x),\ y'(x), \ldots, y^{(m-1)}(x))$$

The initial conditions should be

$$y^{(i)}(x_0) = y_{i0} \qquad\qquad i = 0, 1, \ldots, m-1$$

Transformation

By introducing auxiliary functions

$$z_1(x) = y(x), \quad z_2(x) = y'(x), \ \ldots \ z_m(x) = y^{(m-1)}(x)$$

The higher order ODE can be transformed into an equivalent system of first-order differential equations.

$$\begin{cases} z_1' & = z_2 \\ z_2' & = z_3 \\ \vdots & \\ z_{m-1}' & = z_m \\ z_m' & = f(x, z_1, z_2, \ldots, z_m) \end{cases}$$

And then the initial conditions are

$$z_1(x_0) = y_{00}, \quad z_2(x_0) = y_{10}, \; \ldots \; z_m(x_0) = y_{m-1,0}$$

Example : Orbit of a Planet. Let m, M denote the mass of a planet and the sun respectively. According to the Gravitation Law, there is a gravitation

$$\vec{F} = -k\frac{Mm}{|\vec{r}|^2}\,\frac{\vec{r}}{|\vec{r}|}$$

acting on the planet, where $\vec{r}$ is the vector connecting the sun to the planet.

Put the sun on the origin of coordinates, then $\vec{r} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ represents the displacement of the planet, and satisfies the Newton Second Law of Motion:

$$\begin{aligned} m\,\frac{d^2\vec{r}}{dt^2} &= \vec{F} \\ &= -k\frac{Mm}{|\vec{r}|^3}\,\vec{r} \end{aligned}$$

Hence, we obtain an ODE system of second order

$$\begin{cases} x''(t) = -kM(x^2+y^2)^{-\frac{3}{2}}x\,, \\ y''(t) = -kM(x^2+y^2)^{-\frac{3}{2}}y\,. \end{cases} \tag{3.9}$$

The initial conditions are of the form

$$\begin{cases} x(t_0) = x_0 \\ y(t_0) = y_0 \end{cases} \qquad \begin{cases} x'(t_0) = v_{x0} \\ y'(t_0) = v_{y0} \end{cases}$$

Example : Trajectory Consider the trajectory of a shell. Assume the resistance of air is proportional to the square of the velocity. Thus the trajectory of a shell is governed by the ODE

$$m\,\frac{d^2\vec{r}}{dt^2} = -mg\,\vec{e_2} - k\left|\frac{d\vec{r}}{dt}\right|\frac{d\vec{r}}{dt}\,. \tag{3.10}$$

where $\vec{r}(t)$ represents the displacement of the shell with mass m and $\vec{e_2}$ the unit vector on the y-axis (the direction perpendicular to the earth).

3.2.1 Euler's Method

For solving the initial value problem

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

the simplest method ariese by approximating $y'(x)$ by the finite difference quotient

$$y'(x) \approx \frac{1}{h}[y(x+h) - y(x)]$$

Using this approximation at the initial point, i.e. for $x = x_0$, gives

$$y(x_0 + h) \approx y(x_0) + h\, f(x_0,\, y(x_0))$$

Let $x_1 = x_0 + h$. Since $y(x_0)$ is known from initial condition $y_0 = y(x_0)$, then $y_1 = y_0 + h\, f(x_0,\, y_0)$ gives an approximation to the solution at x_1. The proess can be continued. Let $x_k = x_0 + kh,\ k = 0, 1, \ldots$ and define the approximations y_{k+1} to the solution at x_{k+1} by

$$y_{k+1} = y_k + h\, f(x_k,\, y_k), \qquad k = 0, 1, \ldots \tag{3.11}$$

The above algorithm is called Euler's method.

Discretization Error

Another derivation of Euler's method is by Taylor expansion.

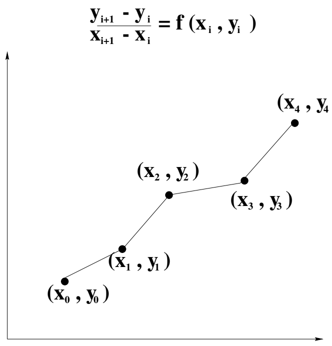
$$y(x) = y(x_k) + y'(x_k)(x - x_k) + \frac{1}{2}y''(\xi_k)(x - x_k)^2, \qquad \xi_k \text{ is between } x_k \text{ and } x\,.$$

Dropping the remainder term we have

$$y(x) \approx y(x_k) + (x - x_k) f(x_k,\, y(x_k))$$

Taking $x = x_{k+1}$ gives

$$y(x_{k+1}) \approx y(x_k) + h\, f(x_k,\, y(x_k))\,.$$



Euler's Method

It can be seen that, the remainder term in Taylor expansion is of order $O(h^2)$. It menas that if we use an accurate $y(x_k)$ to compute y_{k+1} by

$$y_{k+1} = y(x_k) + h\, f(x_k,\, y(x_k))$$

then the error of $y_{k+1} - y(x_{k+1})$ is of order $O(h^2)$. But now there is an accumulative error propagated from y_1 to y_k. Hence, the order must be lower than $O(h^2)$.

Global Discretization Error

Let $a = x_0, \quad x_N = x_0 + Nh = b$ and $x_i = a + ih, \quad i = 0, 1, \ldots$. Assume $y(x_i)$ and y_i are the exact solution and approximation obtained by Euler's method respectively. Then the Golbal Discretization Error is defined as

$$e(h) = \max_{1 \le i \le N} |y_i - y(x_i)|$$

We wish to know whether and how rapidly $e(h) \to 0$ as $h \to 0$ (Note that then $N = \dfrac{(b-a)}{h} \to \infty$).

Theorem 3.1 If the function $f(x, y)$ has a bounded partial derivative with respect to its second variable and if the solution has a bounded second derivative, then the Euler approximation converges to the exact solution as $h \to 0$ and $e(h) = O(h)$.

Proof: Let $e_k = y(x_k) - y_k$, $\left|\dfrac{\partial f}{\partial y}\right| \le K$ and $|y''| \le M_2$

Since
$$\begin{aligned} y(x_k) &= y(x_{k-1}) + h\, f(x_{k-1},\, y(x_{k-1})) + \frac{h^2}{2} y''(\xi_{k-1}), \\ y_k &= y_{k-1} + h\, f(x_{k-1},\, y_{k-1}), \end{aligned}$$
then
$$\begin{aligned} y(x_k) - y_k &= y(x_{k-1}) - y_{k-1} + h[f(x_{k-1},\, y(x_{k-1})) - f(x_{k-1},\, y_{k-1})] \\ &\quad + \frac{h^2}{2} y''(\xi_k). \end{aligned}$$

Because $[f(x_{k-1},\, y(x_{k-1})) - f(x_{k-1},\, y_{k-1})] = \dfrac{\partial f}{\partial y}(x_{k-1},\, \zeta_{k-1})[y(x_{k-1}) - y_{k-1}]$,
where ζ_{k-1} is between y_{k-1} and $y(x_{k-1})$,

we have
$$\begin{aligned} |e_k| &\le |e_{k-1}| + h\, K |e_{k-1}| + \frac{M_2}{2} h^2 \\ &= (1 + h\, K)|e_{k-1}| + \frac{M_2}{2} h^2 \end{aligned}$$

Apply the above inequality recursively,

$$
\begin{aligned}
|e_k| &\le (1+h\,K)\left\{(1+h\,K)|e_{k-2}| + \frac{M_2}{2}h^2\right\} + \frac{M_2}{2}h^2 \\
&= (1+h\,K)^2|e_{k-2}| + (1+h\,K)\frac{M_2}{2}h^2 + \frac{M_2}{2}h^2 \\
&\le (1+h\,K)^3|e_{k-3}| + (1+h\,K)^2\frac{M_2}{2}h^2 + (1+h\,K)\frac{M_2}{2}h^2 + \frac{M_2}{2}h^2 \\
&\le \cdots \\
&\le (1+h\,K)^k|e_0| + \left[(1+h\,K)^{k-1} + (1+h\,K)^{k-2} + \cdots + 1\right]\frac{M_2}{2}h^2 \\
&= \frac{M_2}{2}h^2\sum_{j=0}^{k-1}(1+h\,K)^j \\
&= \frac{(1+h\,K)^k - 1}{h\,K}\,\frac{M_2}{2}h^2 \\
&\le \frac{(1+h\,K)^N - 1}{h\,K}\,\frac{M_2}{2}h^2
\end{aligned}
$$

Since $\quad (1+h\,K) \le e^{h\,K} \quad$ then $\quad (1+h\,K)^N \le e^{N\,h\,K} = e^{(b-a)K}$

Hence, for all $0 \le k \le N$

$$
|e_k| \le C\,h, \qquad \text{where } C = \frac{(e^{(b-a)K} - 1)M_2}{2K}.
$$

The above theorem implies that Euler's method is of first-order accuracy. If we halve the step size, h, it is expected that the error will decrease by about a factor of 2.

Example : Consider the initial problem

$$
\begin{cases} y' &= y \\ y(0) &= 1 \end{cases}
$$

The exact solution is $y(x) = e^x$. We compute the solution at $x = 1$ by Euler's method using $h = 1,\ 1/2,\ 1/4,\ 1/8,\ 1/16$. It is seen from Table 3.1 that the error decreases by about a factor of 2 when the mesh size h decreases.

Higher-Order Methods by Taylor Expansion

The conceptually simplest approach to higher-order methods is to use more terms in the Taylor expansion. Compared with Euler's method, one more term is taken so that

$$
y(x_{k+1}) \approx y(x_k) + h\,y'(x_k) + \frac{h^2}{2}y''(x_k)
$$

h	Computed Value	Error	Error Ratio
1	2.000	0.718	
1/2	2.250	0.468	0.65
1/4	2.441	0.277	0.59
1/8	2.566	0.152	0.55
1/16	2.638	0.080	0.53

Table 3.1: *Error in Euler's method.*

where the remainder of $O(h^3)$ has been dropped.
As before, $y'(x_k)$ can be replaced by $f(x_k, y(x_k))$, but we now need to replace $y''(x_k)$ also. It follows from

$$y''(x) = \frac{d}{dx} f(x, y(x)) = f_x(x, y) + f_y(x, y)y'(x)\,,$$

that

$$y''(x_k) = f_x(x_k, y(x_k)) + f_y(x_k, y(x_k)) \cdot f(x_k, y(x_k))\,.$$

Using this to replace $y''(x_k)$ leads to the method

$$y_{k+1} = y_k + h\,f(x_k, y_k) + \frac{h^2}{2}[f_x(x_k, y_k) + f_y(x_k, y_k)f(x_k, y_k)]$$

which can be shown the global discretization error of $O(h^2)$.

3.2.2 Runge-Kutta Methods

Instead of computing the partial derivatives of $f(x, y)$, we could also obtain highter-order methods by making more evaluations of $f(x, y)$ at each step. This class of methods known as Runge-Kutta methods are more efficient than the above by Taylor expansion. The simplest such method is

$$y_{k+1} = y_k + \frac{h}{2}[f(x_k, y_k) + f(x_{k+1}, y_k + h\,f(x_k, y_k)]\,. \tag{3.12}$$

where we have replaced $f(x_k, y_k)$ in Euler's method by an average of $f(x, y)$ evaluated at (x_k, y_k) and $(x_{k+1}, y_k + h\,f(x_k, y_k))$, the later one is just the point obtained by Euler's method. For the method, it can be shown that $e(h) = O(h^2)$, so it is called the second-order Runge-Kutta method.

The most famous of the Runge Kutta methods is the fourth-order method given by

$$\begin{cases} y_{k+1} &= y_k + \frac{h}{6}(F_1 + 2F_2 + 2F_3 + F_4) \\ F_1 &= f(x_k, y_k), \qquad F_2 = f(x_k + \frac{h}{2},\, y_k + \frac{h}{2}F_1) \\ F_3 &= f(x_k + \frac{h}{2},\, y_k + \frac{h}{2}F_2), \qquad F_4 = f(x_{k+1},\, y_k + hF_3) \end{cases} \tag{3.13}$$

Here, the $f(x_k, y_k)$ in Euler's method has been replaced by a weighted average of f evaluated at four different points. Methods of order p require $p+1$ evaluations of f for $5 \le p \le 7$ and $p+2$ evaluations for $p \ge 8$. This is in contrast to only p evaluations if $p \le 4$.

It is interesting to note that if we apply the above mentioned methods to the initial problem

$$\begin{cases} y' &= f(x) \\ y(a) &= 0 \end{cases}$$

then, Euler's method is the composite rectangle rule; the second-order Runge-Kutta method is the composite trapezoidal rule; and the fourth-order Runge-Kutta method is the composite Simpson's rule.

Example : Consider again the initial problem

$$\begin{cases} y' &= y \\ y(0) &= 1 \end{cases}$$

We now compute the solution at $x = 1$ by fourth-order Runge-Kutta method. The exact solution $y(1) = e = 2.71828182846$.

h	Computed Value	Error	Error Ratio
1	2.70833333333	0.99485^{-2}	
1/2	2.71734619141	0.93564^{-3}	0.094
1/4	2.71820993920	0.71889^{-4}	0.077
1/8	2.71827684442	0.49840^{-5}	0.069
1/16	2.71828150034	0.32812^{-6}	0.066
1/32	2.71828180741	0.21048^{-7}	0.064
1/64	2.71828182713	0.13327^{-8}	0.063

Table 3.2: *Error in Fourth-Order Runge-Kutta Method.*

Rounding Error

As the error estimate shows, the discretization error goes to zero as h goes to zero. However, the smaller h is, the more steps is needed to go to attain a fixed point and then the larger will be the rounding error. In practice, for a fixed word length in the computer arithmetic, there will be a size of h below which the rounding error will become the dominant contribution to the overall error. This minimum h is very difficult to ascertain in advance but usually can be checked out from some numerical phenomenon, for example, the violation of monotone of the data sequence computed. The minimum h will vary from method to method and problem to problem.

```
              Forward Euler, dy/dt=y, y(0)=1
            solution at t=1 (single precision)
     h              y(t)              yh               y-yh
 .50000000E+00   .27182817E+01   .22500000E+01   .46828175E+00
 .25000000E+00   .27182817E+01   .24414063E+01   .27687550E+00
 .12500000E+00   .27182817E+01   .25657845E+01   .15249729E+00
 .62500000E-01   .27182817E+01   .26379285E+01   .80353260E-01
 .31250000E-01   .27182817E+01   .26769896E+01   .41292191E-01
 .15625000E-01   .27182817E+01   .26973457E+01   .20936012E-01
 .78125000E-02   .27182817E+01   .27077391E+01   .10542631E-01
 .39062500E-02   .27182817E+01   .27129931E+01   .52886009E-02
 .19531250E-02   .27182817E+01   .27156358E+01   .26459694E-02
 .97656250E-03   .27182817E+01   .27169552E+01   .13265610E-02
 .48828125E-03   .27182817E+01   .27176132E+01   .66852570E-03
 .24414063E-03   .27182817E+01   .27179534E+01   .32830238E-03
 .12207031E-03   .27182817E+01   .27181191E+01   .16260147E-03
 .61035156E-04   .27182817E+01   .27181976E+01   .84161758E-04
 .30517578E-04   .27182817E+01   .27182360E+01   .45776367E-04
 .15258789E-04   .27182817E+01   .27182436E+01   .38146973E-04
 =============================================================
 .76293945E-05   .27182817E+01   .27182052E+01   .76532364E-04
 .38146973E-05   .27182817E+01   .27178402E+01   .44155121E-03
 .19073486E-05   .27182817E+01   .27170022E+01   .12795925E-02
 .95367432E-06   .27182817E+01   .27137330E+01   .45487881E-02
 .47683716E-06   .27182817E+01   .26949396E+01   .23342133E-01
```

```
              4th order Runge-Kutta, dy/dt=y, y(0)=1
              solution at t=1 (single precision)
     h              y(t)              yh               y-yh
.10000000E+01   .27182817E+01   .27083335E+01   .99482536E-02
.50000000E+00   .27182817E+01   .27173462E+01   .93555450E-03
.25000000E+00   .27182817E+01   .27182102E+01   .71525574E-04
.12500000E+00   .27182817E+01   .27182767E+01   .50067902E-05
.62500000E-01   .27182817E+01   .27182813E+01   .47683716E-06
=============================================================
.31250000E-01   .27182817E+01   .27182822E+01  -.47683716E-06
.15625000E-01   .27182817E+01   .27182820E+01  -.23841858E-06
.78125000E-02   .27182817E+01   .27182834E+01  -.16689301E-05
.39062500E-02   .27182817E+01   .27182829E+01  -.11920929E-05
.19531250E-02   .27182817E+01   .27182825E+01  -.71525574E-06
```

.97656250E-03	.27182817E+01	.27182789E+01	.28610229E-05
.48828125E-03	.27182817E+01	.27182808E+01	.95367432E-06
.24414063E-03	.27182817E+01	.27182782E+01	.35762787E-05
.12207031E-03	.27182817E+01	.27182817E+01	.00000000E+00
.61035156E-04	.27182817E+01	.27182791E+01	.26226044E-05
.30517578E-04	.27182817E+01	.27182746E+01	.71525574E-05
.15258789E-04	.27182817E+01	.27182662E+01	.15497208E-04
.76293945E-05	.27182817E+01	.27182157E+01	.66041946E-04
.38146973E-05	.27182817E+01	.27178450E+01	.43678284E-03
.19073486E-05	.27182817E+01	.27170050E+01	.12767315E-02

3.2.3 Multistep Methods

Another approach to higher-order methods utilizies information already computed and does not require additional evalutaions of $f(x,y)$. One of the simplest such methods is

$$y_{k+1} = y_k + \frac{h}{2}[3f(x_k, y_k) - f(x_{k-1}, y_{k-1})] \tag{3.14}$$

for which $e(h) = O(h^2)$, and is known as the second-order Adams-Bashforth method. Note that the method requires only the evaluation of $f(x_k, y_k)$ at each step, the value $f(x_{k-1}, y_{k-1})$ being known from the previous step.

We now consider the general construction of Adams-Bashforth methods. Let $y_k, y_{k-1}, \ldots, y_{k-N}$ be the computed approximations to the solution at $x_k, x_{k-1}, \ldots, x_{k-N}$. Let $f_i = f(x_i, y_i)$ and let $p(x)$ be the interpolating polynomial of degree N that satisfies

$$p(x_i) = f_i, \qquad i = k, k-1, \ldots, k-N\,.$$

We may then consider $p(x)$ to be an approximation to $f(x, y(x))$. Since the solution $y(x)$ satisfies

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} y'(x)dx = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx \int_{x_k}^{x_{k+1}} p(x)dx$$

then we obtain the so-called Adams-Bashforth (AB) methods as following

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} p(x)dx\,.$$

The following lists a few special cases of the Adams-Bashforth methods:

- $N = 0$: $p(x) = f_k$ gives Euler's method.
- $N = 1$: $p(x) = p_1(x) = f_k - \dfrac{(x - x_k)}{h}(f_k - f_{k-1})$, which leads to

  $$y_{k+1} = y_k + \frac{h}{2}(2f_k - f_{k-1}),$$

  is second-order Adams-Bashforth method mentioned above.

- $N = 2$: $p_2(x) = p_1(x) + \dfrac{(x - x_k)(x - x_{k-1})}{2h^2}(f_k - 2f_{k-1} + f_{k-2})$, which leads to
$$y_{k+1} = y_k + \frac{h}{12}(23f_k - 16f_{k-1} + 5f_{k-2})$$
is the third-order Adams-Bashforth method.
- $N = 3$: $p_3(x) = p_2(x) - \dfrac{(x - x_k)(x - x_{k-1})(x - x_{k-2})}{3!h^3}(f_k - 3f_{k-1} + 3f_{k-2} - f_{k-3})$, which leads to
$$y_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3})$$
is a fourth-order AB method.

In principle, we can continue the preceding process to obtain Adams-Bashforth methods of arbitrarily high-order, but the formulas become increasingly complex as N increases. The Adams-Bashforth methods are called multistep methods since prior values of the data are used. This is in contrast to the Runge-Kutta methods which use no prior data and are called one-step methods.

Starting values

Multistep methods can not start by theirselves. For example, consider the fourth-order Adams-Bashforth method. The initial value y_0 is given, but for $k = 0$, the information is needed at x_{-1}, x_{-2}, x_{-3} which doesn't exist. The method needs "help" getting started. We can not use the fourth-order multistep method until $k \geq 3$.

A common policy is to use a one-step method, such as a Runge-Kutta method of the same order of accuracy at some starting steps.

Comparison

Adams-Bashforth methods of arbitrary order require only one evaluation of $f(x, y)$ at each step, the "cost" is lower than that of Runge-Kutta methods. But they can not self-start and need the help of Runge-Kutta methods. Runge-Kutta methods are much easier to change step-size, hence they are more suitable for use in an adaptive algorithm.

Implicit and Predictor-Corrector Methods

The Adams-Bashforth methods can be unstable due to the fact they are obtained by integrating the interpolating polynomial outside the interval of the data that defines the polynomial. This can be remedied by using data at x_{k+1} to determine the interpolating polynomial.

Consider $f(x, y(x))$ approximated by $p(x)$ which is the linear function on $[x_k, x_{k+1}]$ that interpolates (x_k, f_k) and (x_{k+1}, f_{k+1}), then from
$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} y'(x)dx = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx \int_{x_k}^{x_{k+1}} p(x)dx$$
we obtain a formula
$$y_{k+1} = y_k + \frac{h}{2}(f_{k+1} + f_k)$$

Similarly, if $p(x)$ is the cubic interpolating polynomial for the points (x_{k+1}, f_{k+1}), (x_k, f_k), (x_{k-1}, f_{k-1}) and (x_{k-2}, f_{k-2}), then the formula is

$$y_{k+1} = y_k + \frac{h}{24}[9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}]$$

This kind of formula is called the Adams-Moulton method. The former one is of second order and the latter one is of fourth order.

Now, note that in the above formulas, f_{k+1} is not known since we need y_{k+1} to evaluate $f_{k+1} = f(x_{k+1}, y_{k+1})$, but y_{k+1} is not get known. Hence the Adams-Moulton methods define y_{k+1} only implicitly. Thus, the Adams-Moulton methods are called Implicit whereas the Adams-Bashforth methods are called Explicit since no equation needs to be solved to obtain y_{k+1} at each step.

In principle, we could attempt to solve the nonlinear algebraic equations for y_{k+1} by Newton's method or other methods, discussed in next chapter. However, another use of implicit methods is to combine an explicit formula with an implicit one to form a Predictor-Corrector method. For example, the second order Adams-Moulton methods define y_{k+1} as the solution of the following nonlinear equation

$$y_{k+1} = y_k + \frac{h}{2}[f(x_{k+1}, y_{k+1}) + f(x_k, y_k)]$$

Instead of interative method, we get a "predicted" value $y_{k+1}^{(p)}$ by the second order explicit Adams-Bashforth formula

$$y_{k+1}^{(p)} = y_k + \frac{h}{2}(3f_k - f_{k-1})$$

and then compute $f_{k+1}^{(p)} = f(x_{k+1}, y_{k+1}^{(p)})$ and substitute

$$y_{k+1} = y_k + \frac{h}{2}[f_{k+1}^{(p)} + f_k]$$

The method is entirely explicit.

A commonly used predictor-corrector method is the combination of the fourth-order explicit and implicit Adams methods:

$$\begin{cases} y_{k+1}^{(p)} &= y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}) \\ f_{k+1}^{(p)} &= f(x_{k+1}, y_{k+1}^{(p)}) \\ y_{k+1} &= y_k + \frac{h}{24}(9f_{k+1}^{(p)} + 19f_k - 5f_{k-1} + f_{k-2}) \end{cases}$$

The Adams-Bashforth formula is used as a "predictor" and the Adams-Moulton formula is used as a "corrector".

We could also take additional corrector steps. It just equal an iterative method to solve the nonlinear equation.

3.2.4 Numerical Solutions of Systems of ODEs

We now indicate how the methods that have been discussed can be used for systems of equations. Write the system in the vector form

$$Y'(x) = F(x, Y(x))$$

where $Y(x)$ and $F(x,y)$ represent the vectors

$$Y(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{bmatrix} \qquad F(x,y) = \begin{bmatrix} f_1(x, y_1, \ldots, y_n) \\ f_2(x, y_1, \ldots, y_n) \\ \vdots \\ f_n(x, y_1, \ldots, y_n) \end{bmatrix}$$

Then, Euler's method can be written as

$$Y^{(k+1)} = Y^{(k)} + h\, F(x_k, Y^{(k)})$$

In component form, for example $n = 2$, this would be

$$\begin{aligned} y_1^{(k+1)} &= y_1^{(k)} + h\, f_1(x_k, y_1^{(k)}, y_2^{(k)}) \\ y_2^{(k+1)} &= y_2^{(k)} + h\, f_2(x_k, y_1^{(k)}, y_2^{(k)}) \end{aligned}$$

Clearly, the succinct vector notation is advantageous, especially for a large number of equations. Similiarly, the second-order Runge-Kutta method can be written in vector form as

$$Y^{(k+1)} = Y^{(k)} + \frac{h}{2}[F(x_k, Y^{(k)} + F(x_{k+1}, Y^{(k)} + h\, F(x_k, Y^{(k)})]$$

and the second-order Adams-Bash-forth method can be written as

$$Y^{(k+1)} = Y^{(k)} + \frac{h}{2}[3F(x_k, Y^{(k)}) - F(x_{k-1}, Y^{(k-1)})]$$

3.2.5 Stability

Unstable Solutions

One potential difficulty in solving initial value problems is that the solution may be unstable. Consider the second-order initial problem

$$\begin{cases} y'' = 10y' - 11y \\ y(0) = 1, \quad y'(0) = -1 \end{cases} \tag{3.15}$$

The solution is $y(x) = e^{-x}$. Now suppose the initial conditions are changes to be

$$y(0) = 1 + \epsilon, \qquad\qquad y'(0) = -1$$

Then the solution is changed correspondingly to be

$$y(x) = (1 + \frac{11}{12}\epsilon)e^{-x} + \frac{\epsilon}{12}e^{11x}$$

Therefore, for any $\epsilon > 0$, no matter how small, the second term cause the solution to tend to infinity as $x \to \infty$. In this case, we say that the solution is unstable, since arbitrarily small changes in the initial condition can produce arbitrarily large changes in the solution as $x \to \infty$.

One would also say that this problem is Ill-Conditioned; it is extremely difficult to solve numerically because rounding and discretization error will cause the same effect as changing the initial conditions and the approximate solution will tend to diverge.

Unstable Methods

We now turn to possible unstabilities in the numerical method. It is second-order accurate using central divided difference to approximate the derivative, i.e.,

$$\frac{y(x+h) - y(x-h)}{2h} = y'(x) + O(h^2)$$

This gives rise to the following numerical method

$$y_{n+1} = y_{n-1} + 2h\, f(x_n, y_n)$$

We now apply the method to the problem

$$\begin{cases} y' & = -2y + 1 \\ y(0) & = 1 \end{cases}$$

whose exact solution is

$$y(x) = \frac{1}{2}e^{-2x} + \frac{1}{2}.$$

The above solution is stable: since if the initial condition is changed to $y(0) = 1 + \epsilon$, the solution becomes

$$y(x) = (\frac{1}{2} + \epsilon)e^{-2x} + \frac{1}{2}$$

and the change is $\epsilon\, e^{-2x}$.

The mentioned method applied to the problem is

$$\begin{aligned} y_{n+1} &= y_{n-1} + 2h(-2y_n + 1) \\ &= -4h\, y_n + y_{n-1} + 2h, \end{aligned}$$

Since it is a two-step method, except $y_0 = 1$, we need to supply y_1 to start the process. Assume that we take y_1 to be the exact solution at $x = h$:

$$x = h: \quad y_1 = \frac{1}{2}e^{-2h} + \frac{1}{2}\,.$$

For any fixed $h > 0$, we will see that

$$\lim_{n\to\infty} |y_n| = \infty\,,$$

where sequence y_n is generated by the method. In contrast, for the exact solution $y(x)$, $\lim_{x\to\infty} y(x) = \frac{1}{2}$ Thus, the method although second-order accurate, exhibits unstable behaviour.

A representation of approximate y_n

$$y_n = c_1\lambda_1^n + c_2\lambda_2^n + \frac{1}{2}$$

where λ_1 and λ_2 are roots of equation

$$\lambda^2 + 4h\lambda - 1 = 0$$

$$\Rightarrow \begin{cases} \lambda_1 = \sqrt{1+4h} - 2h \\ \\ \lambda_2 = -\sqrt{1+4h} - 2h \end{cases}$$

constants c_1 and c_2 are solutions of linear equations

$$\begin{cases} c_1 + c_2 + \frac{1}{2} = y_0 = 1 \\ \\ c_1\lambda_1 + c_2\lambda_2 + \frac{1}{2} = y_1 = \frac{1}{2}e^{-2h} + \frac{1}{2} \end{cases}$$

Because $|\lambda_2| > 1, \quad \lim_{n\to\infty} |y_n| = \infty$.

Stability of Methods

All of the multistep methods described so far are special cases of what are called linear multistep methods of the form

$$y_{k+1} = \sum_{i=1}^{m} \alpha_i y_{k+1-i} + h\sum_{i=0}^{m} \beta_i f_{k+1-i}$$

where $f_j = f(x_j, y_j)$ and m is some fixed integer. If $\beta_0 = 0$, the method is explicit, and if $\beta_0 \neq 0$, then the method is implicit.

In all of the Adams methods, $\alpha_1 = 1$ and $\alpha_i = 0,\ i > 1$; in the Adams-Bashforth methods, $\beta_0 = 0$; in the Adams-Moulton methods, $\beta_0 \neq 0$. Associated with a multistep method is the polynomial

$$p(\lambda) = \lambda^m - \alpha_1\lambda^{m-1} \cdots \alpha_{m-1}\lambda - \alpha_m$$

The method is stable provided that all roots λ_i of $p(\lambda)$ satisfy $|\lambda_i| \leq 1$ and only one $|\lambda_i| = 1$.

For a m-step Adams methods, $p(\lambda) = \lambda^m - \lambda^{m-1}$, so one root is 1 and other $(m-1)$ roots are zero and hence they are stable. It is possible to extend the above framework so as to include Runge-Kutta methods, and these methods are also stable.

For the method

$$y_{n+1} = y_{n-1} + 2h\, f(x_n, y_n)$$

the polynomial is

$$p(\lambda) = \lambda^2 - 1$$

with roots ± 1; hence the method is not stable.

It should be noted that from the stability theory, a stable method will give accurate results for sufficiently small h. But, on the other hand a stable method can exhibit unstable behaviour if h is too large. Although, in principle, h can be taken sufficiently small to overcome this difficulty, the computing time may become prohibitive. This is the situation with ODEs that are known as Stiff.

Stiff Equations

An ODE is called stiff if its solution consists of a steady-state part that does not grow significantly, together with a transient part that decays rapidly to zero. Stiff problems occur in a wide variety of applications, including the study of spring and damping systems, the analysis of control systems, and problems in chemical kinetic.

As a simple example, consider the problem

$$y' = -100y + 100 \qquad y(0) = 2. \tag{3.16}$$

The exact solution of the problem is

$$y(x) = e^{-100x} + 1$$

It is clear that the solution is stable since if we change the initial condition to $y(0) = 2 + \epsilon$, then the solution changes by $\epsilon\, e^{-100x}$. Euler's method applied to this problem is

$$\begin{aligned} y_n &= y_{n-1} + h(-100y_{n-1} + 100) \\ &= (1 - 100h)y_{n-1} + 100h \end{aligned}$$

Applying the corresponding formulas for $y_{n-1}, y_{n-2}, \ldots y_1$, it is possible to obtain

$$y_n = (1 - 100h)^n + 1 \tag{3.17}$$

Note that $y(x)$ decrease very rapidly from $y(0) = 2$ to its limiting value of 1; for example, $y(0.1) \approx 1 + 5 \times 10^{-5}$. Initially, therefore, we expect to require a small step size h to compute the solution accurately. Beyond, say, $x = 0.1$, the solution varies slowly, so intuitively, we would expect to obtain sufficient accuracy using a relatively large h. But this is not correct for the Euler's method. We see from the formula (3.17) that $|1 - 100h| > 1$ when $h > 0.02$ and the approximation y_n grows at each step and shows an unstable behaviour. The quantity $(1 - 100h)^n$ is a good approximation for very small h, but becomes a poor approximation as h becomes as large as 0.02. Even though the exponential term e^{-100x} contributes virtually nothing to the solution after $x = 0.1$, Euler's method still requires that we approximate it sufficiently accurately to maintain stability. This is the typical problem with stiff equations: the solution contains a component (the transient part that decays rapidly to zero) that contributes very little to the solution, but the used methods require that it be approximated accurately.

The general approach to stiff problems is to use implicit methods. The simplest implicit method is

$$y_{n+1} = y_n + h\,f(x_{n+1}, y_{n+1})$$

which is known as the Backward Euler Method. It is of the same form as Euler's method except that f is evaluated at (x_{n+1}, y_{n+1}) rather than at (x_n, y_n), hence, the method is implicit. Using this method to the discussed problem, we obatin

$$y_{n+1} = y_n + h(-100y_{n+1} + 100)$$

which can be put in the form

$$y_{n+1} = \frac{(y_n + 100h)}{(1 + 100h)}$$

From the formula and initial condition, we obtain

$$y_n = \frac{1}{(1 + 100h)^n} + 1$$

For this approximation, we can see that there is no unstable behaviour regardless of the size of h.

The above is a simple example for demonstration. In practice, stiff problems are usually nonlinear equations or systems of nonlinear equations. Hence when using implicit methods, the solution of a system of nonlinear equations would be needed at each step. This is costly in computer time, but the effective handling of stiff equations requires that some kind of implicitness be brought into the numerical method.

Problem 3.4 Transform the ODEs in (3.9) and (3.10) into first order ODE systems.

Problem 3.5 Apply Euler's method (3.11) to the initial value problem $y' = -y, 0 \le x \le 1, y(0) = 1$, with $h = 0.25$. Compare your answers to the exact solution $y(x) = e^{-x}$. Repeat for $h/2$ and $h/4$. Next, apply second-order Runge-Kutta method (3.12) to this problem and compare with Euler's method.

Problem 3.6 Write a computer program to carry out the second-order Adams-Bashforth method (3.14). Use the second-order Runge-Kutta method to supply the missing starting value y_1. Apply your program to the problems of Excercise 3.5 and compare your results with the Euler and second-order Runge-Kutta methods.

Problem 3.7 By letting $z = y'$, show that the problem (3.15) is equivalent to the first-order system $y' = z, z' = 10z + 11y$, with initial condition $y(0) = 1$ and $z(0) = -1$. Attempt to solve this system numerically by any of the methods of this chapter and discuss your results.

Problem 3.8 Consider the method

$$y_{n+1} = y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}),$$

which is known as *Milne's method*. Discuss whether this method is stable and strongly stable.

Problem 3.9 Write a program to carry out Euler's method on (3.16) for different values of h both less than and greater than 0.02. Discuss your results.

Problem 3.10 The function $y(x) = e^{-x}$ is the solution of the problem

$$y'' = y, \qquad y(0) = 1, \quad y'(0) = -1.$$

Is this solution stable?