The discussion given here is intended to help the reader in a valuation of existing codes and a comparison with the entire range of Runge–Kutta and Runge–Kutta–Nyström methods available today. In Sect. 4.4 a subset of these methods is compared with multistep and extrapolation codes to provide a more general assessment of the performance of different types of integration methods.

4.2 Multistep Methods

The Runge–Kutta methods discussed so far may be characterized as single-step methods. No use is made of function values calculated in earlier steps, which means that all integration steps are completely independent of one another. This feature allows a compact implementation of single-step methods and makes them particularly easy to use. Since a new stepsize may be used in each step, single-step methods are well suited for differential equations with rapid changes in the function to be integrated.

One may, however, think of a completely different approach that tries to reduce the total number of function evaluations as much as possible by storing values from previous steps. This leads to the concept of multistep methods which are most efficient for differential equations defined by complicated functions with a lot of arithmetic operations.

The development of multistep integration methods in the 19th and early 20th centuries is closely linked with the work of astronomers who utilized them for an accurate description of solar system bodies. Among these are J. C. Adams, who is most famous for his contribution to the discovery of Neptune, F. R. Moulton, and Ph. H. Cowell, who accurately predicted the motion of Halley's comet before its 1910 return.

4.2.1 Introduction

In order to illustrate the basic principles of multistep methods, it is assumed for the moment that one has already obtained approximate values $\boldsymbol{\eta}_j$ of the solution $\boldsymbol{y}(t_j)$ at equidistant times $t_j = t_0 + jh$ for $j = 0, 1, \ldots, i$. Integrating both sides of the differential equation

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}) \tag{4.41}$$

with respect to t from t_i to t_{i+1} one obtains the equivalent expression

$$\boldsymbol{y}(t_{i+1}) = \boldsymbol{y}(t_i) + \int_{t_i}^{t_i+h} \boldsymbol{f}(t, \boldsymbol{y}(t))dt \quad . \tag{4.42}$$

The integral cannot, however, be evaluated as it is, since it depends itself on the unknown solution $\boldsymbol{y}(t)$ of the differential equation. To circumvent this difficulty, one replaces the integrand by a polynomial $\boldsymbol{p}(t)$ that interpolates some of the values

$$\boldsymbol{f}_j = \boldsymbol{f}(t_j, \boldsymbol{\eta}_j) \tag{4.43}$$

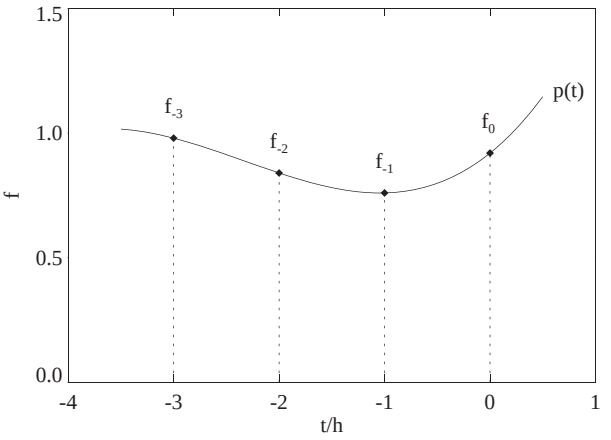


Fig. 4.3. Interpolation of four function values by a third-order polynomial

at previous times t_j that are already known according to the initial assumption. This results in

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{\eta}_i + \int_{t_i}^{t_i+h} \boldsymbol{p}(t)dt \tag{4.44}$$

and the increment function of a multistep method is therefore given by

$$\boldsymbol{\Phi} = \frac{1}{h}\int_{t_i}^{t_i+h} \boldsymbol{p}(t)dt \quad . \tag{4.45}$$

As an example a third-order polynomial is considered (see Fig. 4.3), which is defined by the four function values $\boldsymbol{f}_{i-3}$, $\boldsymbol{f}_{i-2}$, $\boldsymbol{f}_{i-1}$ and $\boldsymbol{f}_i$ at times t_{i-3}, t_{i-2}, t_{i-1} and t_i. This polynomial may be written as

$$\boldsymbol{p}(t) = \boldsymbol{a}_0 + \boldsymbol{a}_1\sigma + \boldsymbol{a}_2\sigma^2 + \boldsymbol{a}_3\sigma^3 \tag{4.46}$$

with $\sigma(t) = (t - t_i)/h$. This yields the simple expression

$$\boldsymbol{\Phi} = \int_0^1 (\boldsymbol{a}_0 + \boldsymbol{a}_1\sigma + \boldsymbol{a}_2\sigma^2 + \boldsymbol{a}_3\sigma^3)d\sigma = \boldsymbol{a}_0 + \boldsymbol{a}_1/2 + \boldsymbol{a}_2/3 + \boldsymbol{a}_3/4 \tag{4.47}$$

for the increment function. Substituting the coefficients

$$\begin{aligned}
\boldsymbol{a}_0 &= (\phantom{-2\boldsymbol{f}_{i-3}\ +9}6\boldsymbol{f}_i)/6 \\
\boldsymbol{a}_1 &= (-2\boldsymbol{f}_{i-3}\ +9\boldsymbol{f}_{i-2}-18\boldsymbol{f}_{i-1}+11\boldsymbol{f}_i)/6 \\
\boldsymbol{a}_2 &= (-3\boldsymbol{f}_{i-3}+12\boldsymbol{f}_{i-2}-15\boldsymbol{f}_{i-1}\ +6\boldsymbol{f}_i)/6 \\
\boldsymbol{a}_3 &= (-1\boldsymbol{f}_{i-3}\ +3\boldsymbol{f}_{i-2}\ -3\boldsymbol{f}_{i-1}\ +1\boldsymbol{f}_i)/6
\end{aligned} \tag{4.48}$$

finally leads to the 4th-order Adams–Bashforth formula

$$\boldsymbol{\Phi}_{\mathrm{AB4}} = \frac{1}{24}(-9\boldsymbol{f}_{i-3} + 37\boldsymbol{f}_{i-2} - 59\boldsymbol{f}_{i-1} + 55\boldsymbol{f}_i) \quad , \tag{4.49}$$

which may be used to calculate the approximate solution

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{\eta}_i + h\boldsymbol{\Phi}_{\mathrm{AB4}} \tag{4.50}$$

of the differential equation at $t_{i+1} = t_i + h$. Repeated application of the Adams–Bashforth formula then yields the solution of the differential equation for subsequent times $t_i + jh$.

In order to start the integration scheme the first four values $\boldsymbol{f}_0$, $\boldsymbol{f}_1$, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$ or, equivalently, $\boldsymbol{\eta}_0$, $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$ and $\boldsymbol{\eta}_3$ are required. They may, for example, be obtained from t_0 and $\boldsymbol{\eta}_0$ using three steps of a fourth or higher-order Runge–Kutta method with sufficient accuracy.

4.2.2 Adams–Bashforth Methods

The simple procedure described in the introductory section may easily be extended to derive general multistep methods of arbitrary order.

For this purpose one makes use of Newton's formula for a polynomial $\boldsymbol{p}_m^i$ of order $m-1$ that interpolates m points

$$(t_{i-m+1}, \boldsymbol{f}_{i-m+1}) \ , \quad \ldots \ , \quad (t_i, \boldsymbol{f}_i)$$

with equidistant nodes t_i. This polynomial is given by the compact expression

$$\boldsymbol{p}_m^i(t) = \boldsymbol{p}_m^i(t_i + \sigma h) = \sum_{j=0}^{m-1} (-1)^j \binom{-\sigma}{j} \nabla^j \boldsymbol{f}_i \quad , \tag{4.51}$$

where the binomial coefficient stands for

$$\binom{-\sigma}{j} = \frac{(-\sigma)(-\sigma-1)\ldots(-\sigma-j+1)}{j!} \tag{4.52}$$

if $j > 0$ and is equal to 1 for $j = 0$. The *backward differences* of $\boldsymbol{f}_i$ are recursively defined by

$$\begin{aligned} \nabla^0 \boldsymbol{f}_i &= \boldsymbol{f}_i \\ \nabla \boldsymbol{f}_i &= \boldsymbol{f}_i - \boldsymbol{f}_{i-1} \\ \nabla^n \boldsymbol{f}_i &= \nabla^{n-1} \boldsymbol{f}_i - \nabla^{n-1} \boldsymbol{f}_{i-1} \end{aligned} \tag{4.53}$$

and may be computed from the given function values as outlined in Fig. 4.4.

Using this notation the increment function of the mth-order Adams–Bashforth multistep method may now be written as

$$\boldsymbol{\Phi}_{\mathrm{AB}m} = \frac{1}{h} \int_{t_i}^{t_i+h} \boldsymbol{p}_m^i(t)\, dt = \sum_{j=0}^{m-1} \gamma_j \nabla^j \boldsymbol{f}_i \tag{4.54}$$

with stepsize independent coefficients

$$\gamma_j = (-1)^j \int_0^1 \binom{-\sigma}{j} d\sigma \quad . \tag{4.55}$$

$$
\begin{array}{ccccccccccc}
\boldsymbol{f}_{i-4} & \cdots & & & & & & & & & \\
 & \searrow & & & & & & & & & \\
\boldsymbol{f}_{i-3} & \rightarrow & \nabla^1 \boldsymbol{f}_{i-3} & \cdots & & & & & & & \\
 & \searrow & & \searrow & & & & & & & \\
\boldsymbol{f}_{i-2} & \rightarrow & \nabla^1 \boldsymbol{f}_{i-2} & \rightarrow & \nabla^2 \boldsymbol{f}_{i-2} & \cdots & & & & & \\
 & \searrow & & \searrow & & \searrow & & & & & \\
\boldsymbol{f}_{i-1} & \rightarrow & \nabla^1 \boldsymbol{f}_{i-1} & \rightarrow & \nabla^2 \boldsymbol{f}_{i-1} & \rightarrow & \nabla^3 \boldsymbol{f}_{i-1} & \cdots & & & \\
 & \searrow & & \searrow & & \searrow & & \searrow & & & \\
\boldsymbol{f}_{i} & \rightarrow & \nabla^1 \boldsymbol{f}_{i} & \rightarrow & \nabla^2 \boldsymbol{f}_{i} & \rightarrow & \nabla^3 \boldsymbol{f}_{i} & \rightarrow & \nabla^4 \boldsymbol{f}_{i} & \cdots
\end{array}
$$

Fig. 4.4. Backward difference table for polynomial interpolation

Table 4.4. Coefficients of Adams–Bashforth methods in backwards difference notation

j	0	1	2	3	4	5	6	7	8
γ_j	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$	$\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Numerical values of $\gamma_0 \ldots \gamma_8$ are given in Table 4.4. They may be obtained from a simple recurrence relation (see e.g. Hairer et al. 1987):

$$\gamma_j = 1 - \sum_{k=0}^{j-1} \frac{1}{j+1-k} \gamma_k \quad . \tag{4.56}$$

The local truncation error of the Adams–Bashforth method decreases with the order m and may be estimated by comparing two methods of order m and $m+1$:

$$e_{\mathrm{AB}m} = |\boldsymbol{y}(t_i + h) - \boldsymbol{\eta}_{\mathrm{AB}m}| \approx |\boldsymbol{\eta}_{\mathrm{AB}m+1} - \boldsymbol{\eta}_{\mathrm{AB}m}| = h|\gamma_m \nabla^m \boldsymbol{f}_i| \quad . \tag{4.57}$$

Since $\nabla^m \boldsymbol{f} / h^m$ is an approximation of the m-th derivative of $\boldsymbol{f}$, the truncation error may also be expressed as

$$e_{\mathrm{AB}m} \approx h^{m+1} |\gamma_m \boldsymbol{f}_i^{(m)}| = h^{m+1} |\gamma_m \boldsymbol{y}_i^{(m+1)}| \tag{4.58}$$

which shows that the order of the Adams–Bashforth method is equal to the number (m) of nodes ($t_{i-m+1} \ldots t_i$).

Substituting the definition of backwards differences into (4.54), the increment function may also be written in terms of the function values $\boldsymbol{f}_j$:

$$\boldsymbol{\Phi}_{\mathrm{AB}m} = \beta_{m1} \boldsymbol{f}_{i-m+1} + \ldots + \beta_{mm} \boldsymbol{f}_i = \sum_{j=1}^{m} \beta_{mj} \boldsymbol{f}_{i-m+j} \quad . \tag{4.59}$$

This formulation of the increment function, which has already been used in our introductory example, avoids the computation of the backwards differences and is therefore more convenient and efficient as long as methods of fixed order are considered. There are, however, several applications where (4.54) is preferable,

Table 4.5. Coefficients of Adams–Bashforth methods up to order $m = 8$.

j	1	2	3	4	5	6	7	8
β_{1j}	1							
β_{2j}	$\frac{-1}{2}$	$\frac{3}{2}$						
β_{3j}	$\frac{5}{12}$	$\frac{-16}{12}$	$\frac{23}{12}$					
β_{4j}	$\frac{-9}{24}$	$\frac{37}{24}$	$\frac{-59}{24}$	$\frac{55}{24}$				
β_{5j}	$\frac{251}{720}$	$\frac{-1274}{720}$	$\frac{2616}{720}$	$\frac{-2774}{720}$	$\frac{1901}{720}$			
β_{6j}	$\frac{-475}{1440}$	$\frac{2877}{1440}$	$\frac{-7298}{1440}$	$\frac{9982}{1440}$	$\frac{-7923}{1440}$	$\frac{4277}{1440}$		
β_{7j}	$\frac{19087}{60480}$	$\frac{-134472}{60480}$	$\frac{407139}{60480}$	$\frac{-688256}{60480}$	$\frac{705549}{60480}$	$\frac{-447288}{60480}$	$\frac{198721}{60480}$	
β_{8j}	$\frac{-36799}{120960}$	$\frac{295767}{120960}$	$\frac{-1041723}{120960}$	$\frac{2102243}{120960}$	$\frac{-2664477}{120960}$	$\frac{2183877}{120960}$	$\frac{-1152169}{120960}$	$\frac{434241}{120960}$

since the use of backwards differences allows a straightforward estimation of the local truncation error and an easy change of the order from one step to the next.

The new coefficients β_{mj} – which are no longer independent of the order m – may be obtained from the γ_j's using the relation

$$\beta_{mj} = (-1)^{m-j} \sum_{l=m-j}^{m-1} \gamma_l \binom{l}{m-j} \tag{4.60}$$

for $j = 1, \ldots, m$ (Grigorieff 1977). Explicit values are given in Table 4.5.

4.2.3 Adams–Moulton and Predictor–Corrector Methods

In the mth-order Adams–Bashforth method the polynomial $\boldsymbol{p}(t)$ is defined by m function values up to and including $\boldsymbol{f}_i$ at time t_i. The integration is, however, performed over the subsequent interval $t_i \ldots t_{i+1}$ where the approximation cannot be expected to be very good.

Another type of multistep method – known as the Adams–Moulton method – therefore uses the polynomial $\boldsymbol{p}_m^{i+1}(t)$ which interpolates m function values at time steps t_{i-m+2} and t_{i+1}:

$$\boldsymbol{p}_m^{i+1}(t) = \boldsymbol{p}_m^{i+1}(t_i + \sigma h) = \sum_{j=0}^{m-1} (-1)^j \binom{-\sigma+1}{j} \nabla^j \boldsymbol{f}_{i+1} \quad . \tag{4.61}$$

Upon integration this yields the Adams–Moulton formula

$$\boldsymbol{\Phi}_{\mathrm{AM}m} = \frac{1}{h} \int_{t_i}^{t_i+h} \boldsymbol{p}_m^{i+1}(t)dt = \sum_{j=0}^{m-1} \gamma_j^* \nabla^j \boldsymbol{f}_{i+1} \tag{4.62}$$

with coefficients

$$\gamma_j^* = (-1)^j \int_0^1 \binom{-\sigma+1}{j} d\sigma \tag{4.63}$$

that are given in Table 4.6. Further values may be calculated from the recurrence relation

$$\gamma_j^* = -\sum_{k=0}^{j-1} \frac{1}{j+1-k} \gamma_k^* \quad . \tag{4.64}$$

Table 4.6. Coefficients of Adams–Moulton methods in backwards difference notation

j	0	1	2	3	4	5	6	7	8
γ_j^*	1	$\frac{-1}{2}$	$\frac{-1}{12}$	$\frac{-1}{24}$	$\frac{-19}{720}$	$\frac{-3}{160}$	$\frac{-863}{60480}$	$\frac{-275}{24192}$	$\frac{-33953}{3628800}$

The order of the Adams–Moulton method is equal to m and is, therefore, the same as that of an Adams–Bashforth method involving m grid points for the interpolating polynomial. The local truncation error of the Adams–Moulton method, which is given by

$$e_{\mathrm{AM}m} \approx h^{m+1} |\gamma_m^* \boldsymbol{y}_i^{(m+1)}| \quad , \tag{4.65}$$

is smaller, however, than that of an Adams–Bashforth method of equal order, since the error constant $|\gamma_m^*|$ is smaller than $|\gamma_m|$.

As with the Adams–Bashforth methods, the backwards differences may be substituted to yield a formulation that depends on $\boldsymbol{f}_j$ only:

$$\boldsymbol{\Phi}_{\mathrm{AM}m} = \beta_{m1}^* \boldsymbol{f}_{i-m+2} + \ldots + \beta_{mm}^* \boldsymbol{f}_{i+1} = \sum_{j=1}^{m} \beta_{mj}^* \boldsymbol{f}_{i+1-m+j} \quad . \tag{4.66}$$

The coefficients β_{mj}^* (see Table 4.7) of the mth-order method may be obtained from the γ_j^* using the relation

$$\beta_{mj}^* = (-1)^{m-j} \sum_{l=m-j}^{m-1} \gamma_l^* \binom{l}{m-j} \tag{4.67}$$

for $j = 1, \ldots, m$ (Grigorieff 1977).

Since the increment function of the Adams–Moulton method depends on

$$\boldsymbol{f}_{i+1} = \boldsymbol{f}(t_{i+1}, \boldsymbol{\eta}_{i+1}) \quad , \tag{4.68}$$

it is not possible to calculate an explicit solution at t_{i+1} from

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{\eta}_i + h\boldsymbol{\Phi}_{\mathrm{AM}m} \quad . \tag{4.69}$$

Table 4.7. Coefficients of Adams–Moulton methods up to order $m = 8$.

j	1	2	3	4	5	6	7	8
β_{2j}	$\frac{1}{2}$	$\frac{1}{2}$						
β_{3j}	$\frac{-1}{12}$	$\frac{8}{12}$	$\frac{5}{12}$					
β_{4j}	$\frac{1}{24}$	$\frac{-5}{24}$	$\frac{19}{24}$	$\frac{9}{24}$				
β_{5j}	$\frac{-19}{720}$	$\frac{106}{720}$	$\frac{-264}{720}$	$\frac{646}{720}$	$\frac{251}{720}$			
β_{6j}	$\frac{27}{1440}$	$\frac{-173}{1440}$	$\frac{482}{1440}$	$\frac{-798}{1440}$	$\frac{1427}{1440}$	$\frac{475}{1440}$		
β_{7j}	$\frac{-863}{60480}$	$\frac{6312}{60480}$	$\frac{-20211}{60480}$	$\frac{37504}{60480}$	$\frac{-46461}{60480}$	$\frac{65112}{60480}$	$\frac{19087}{60480}$	
β_{8j}	$\frac{1375}{120960}$	$\frac{-11351}{120960}$	$\frac{41499}{120960}$	$\frac{-88547}{120960}$	$\frac{123133}{120960}$	$\frac{-121797}{120960}$	$\frac{139849}{120960}$	$\frac{36799}{120960}$

The Adams–Moulton formula is therefore called an "implicit" method and some iterative procedure is required to solve for $\boldsymbol{\eta}_{i+1}$.

To avoid this difficulty, an Adams–Bashforth method of order m is usually combined with an Adams–Moulton method of order m or $m+1$ in a so-called predictor–corrector or PECE-algorithm which consists of four steps:

1. In the first step – the *P*redictor step – an initial estimate of the solution at t_{i+1} is calculated from the Adams–Bashforth formula

$$\boldsymbol{\eta}^{\mathrm{p}}_{i+1} = \boldsymbol{\eta}_i + h\boldsymbol{\Phi}_{\mathrm{AB}} \quad . \tag{4.70}$$

2. The result is used in the *E*valuation step to find the corresponding function value

$$\boldsymbol{f}^{\mathrm{p}}_{i+1} = \boldsymbol{f}(t_{i+1}, \boldsymbol{\eta}^{\mathrm{p}}_{i+1}) \quad . \tag{4.71}$$

3. In the third step – called the *C*orrector – the Adams–Moulton formula is applied to find an improved value

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{\eta}_i + h\boldsymbol{\Phi}_{\mathrm{AM}}(\boldsymbol{f}^{\mathrm{p}}_{i+1}) \quad . \tag{4.72}$$

4. The final *E*valuation step yields the updated function value

$$\boldsymbol{f}_{i+1} = \boldsymbol{f}(t_{i+1}, \boldsymbol{\eta}_{i+1}) \tag{4.73}$$

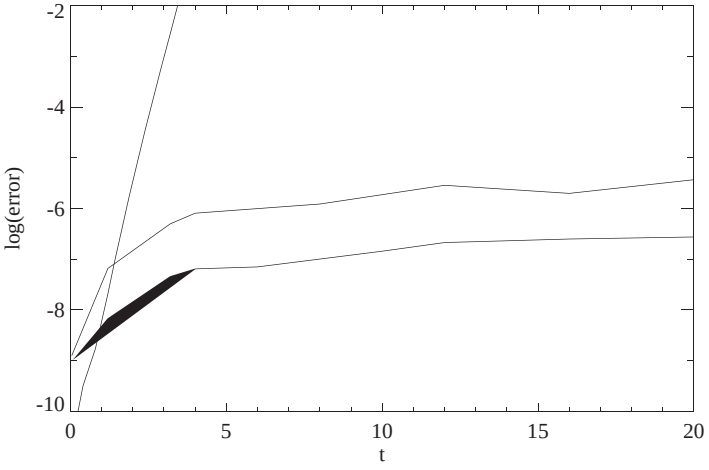
which may then be used for the start of the next integration step.

In principle the third and fourth step would have to be repeated until convergence is achieved to find the exact solution of the Adams–Moulton formula, but since each such iteration costs another function evaluation, this would not be worth the effort. A single corrector step is enough to assure that the order of the combined Adams–Bashforth–Moulton method is equal to that of the implicit method, even though the local truncation error is slightly larger (cf. Grigorieff 1977).

The justification for using the somewhat complicated predictor–corrector algorithm, lies in the *stability* of multistep methods at large stepsizes. Due to the truncation at a fixed order and a limited computing accuracy, the individual steps of the numerical integration are always affected by small local errors. An analysis of the way in which these errors are propagated from one step to the next shows that the errors may grow exponentially for large stepsizes. In order to avoid this unfavorable behavior and to guarantee stability, the stepsize may not exceed a certain limit that depends on the method and the differential equation to be solved.

Low-order methods are generally more stable even for large stepsizes. Due to their modest accuracy, small steps have to be used anyway and stability is often not a serious problem. When using high-order multistep methods, however, stability can pose stringent limits on the allowed stepsize. Even steps that yield a sufficiently small truncation error may then be too large since the propagation of local errors could result in an unbounded growth of the global integration error.

The implicit Adams–Moulton methods behave much better in this respect than the explicit Adams–Bashforth methods and even the approximate solution of the Adams–Moulton formula in the PECE algorithm improves the stability considerably. This is the main reason why it is generally recommended to use one additional function evaluation for the corrector step. Another advantage is that the local truncation error can be reduced by using the corrector. The doubled expense for a single integration step can at least partially be compensated for by larger stepsizes.



AB8 method, but the growth of the global integration error clearly shows the onset of instability. When combined with a corrector, the resulting Adams–Bashforth–Moulton method (ABM8) is stable, however, even at twice the stepsize.

As a compromise between a cheap predictor-only method and a more stable PECE method, a PECE* algorithm (cf. Long et al. 1989) may be useful for the treatment of perturbed satellite orbits. Here E* stands for a *pseudo-evaluate* step, which means that some simplifications are made in the final evaluation step. According to (4.4) the evaluation of $\boldsymbol{f}$ involves the computation of the acceleration

$$\boldsymbol{a}(t, \boldsymbol{r}, \dot{\boldsymbol{r}}) = -\frac{GM_\oplus}{r^3}\boldsymbol{r} + \boldsymbol{p}(t, \boldsymbol{r}, \dot{\boldsymbol{r}}) \quad , \tag{4.74}$$

where the dominant first term arises from the central gravity field of the Earth and $\boldsymbol{p}$ means the sum of all perturbations. Since the perturbations are much smaller than the central force, one makes a small error only if one does not recompute them after the corrector step. In a pseudo-evaluate step only the dominant term of the total acceleration is therefore updated with the coordinates obtained in the corrector step, while the perturbations are taken from the predictor step:

$$\boldsymbol{a}^*(t, \boldsymbol{r}, \dot{\boldsymbol{r}}) = -\frac{GM_\oplus}{r^3}\boldsymbol{r} + \boldsymbol{p}(t, \boldsymbol{r}^{\mathrm{p}}, \dot{\boldsymbol{r}}^{\mathrm{p}}) \quad . \tag{4.75}$$

Since the computation of the perturbations is much more time-consuming than that of the central acceleration term, the PECE* method increases the stability at almost no additional cost.

Irrespective of the use of a stabilizing corrector step one should not arbitrarily increase the order of a multistep method in an attempt to increase the stepsize and the accuracy. For the requirements of typical orbit computations orders in the range from 8 to 12 can usually be recommended.

4.2.4 Interpolation

The multistep methods of Adams' type may be extended in a straightforward manner to provide a solution at intermediate output points. For this purpose it is assumed that a PECE step of size h has been used to advance from t_i to t_{i+1} and that one is interested in the approximate solution at some time $t = t_i + \sigma h$, where $0 < \sigma < 1$. From a total of $m+1$ points

$$(t_{i-m+1}, \boldsymbol{f}_{i-m+1}) \ , \ \ldots \ , \ (t_i, \boldsymbol{f}_i) \ , \ (t_{i+1}, \boldsymbol{f}^{\mathrm{p}}_{i+1})$$

one may construct the interpolating polynomial

$$\boldsymbol{p}^{*,i+1}_{m+1}(t) = \boldsymbol{p}^{*,i+1}_{m+1}(t_i + \sigma h) = \sum_{j=0}^{m} (-1)^j \binom{-\sigma+1}{j} \nabla^j \boldsymbol{f}^*_{i+1} \tag{4.76}$$

that was already used in the implicit Adams–Moulton formula of the corrector step. Here the $*$ denotes that $\boldsymbol{f}^{\mathrm{p}}_{i+1}$ is a predictor value while all other $\boldsymbol{f}_j$ result from the

final (corrector) values of previous steps. Inserting this polynomial into the integral form of the differential equation leads to

$$\boldsymbol{\eta}(t_i + \sigma h) = \boldsymbol{\eta}_i + \int_{t_i}^{t_i+\sigma h} \boldsymbol{p}_{m+1}^{*,i+1}(t)dt \tag{4.77}$$

or

$$\boldsymbol{\eta}(t_i + \sigma h) = \boldsymbol{\eta}_i + h \sum_{j=0}^{m} \hat{\gamma}_j(\sigma) \nabla^j \boldsymbol{f}_{i+1}^{*} \tag{4.78}$$

with

$$\hat{\gamma}_j(\sigma) = \int_0^{\sigma} (-1)^j \binom{-s+1}{j} ds \quad . \tag{4.79}$$

The coefficients $\hat{\gamma}_j$ depend on σ and must therefore be calculated separately for each desired output value. Appropriate recurrence relations may be found e.g. in Shampine & Gordon (1975). Since the differential equations that are usually treated by multistep methods are characterized by complicated functions $\boldsymbol{f}$, the computational effort for the interpolation coefficients is negligible in most applications.

It should be noted that the interpolant given here is continuous at the grid points t_i, by definition, but that the same is not true for the first derivative. For a more detailed discussion of smooth interpolants we refer to Watts & Shampine (1986) and Higham (1989).

4.2.5 Variable Order and Stepsize Methods

In the derivation of the Adams–Bashforth and Adams–Moulton methods it has so far been assumed that the solution of the differential equation is calculated with a constant stepsize, i.e. on a series of equidistant time points. This concept has to be modified whenever the behavior of the solution requires changes of the stepsize during the integration.

The easiest way to realize a variable stepsize consists of stopping the integration and calculating new starting values for another stepsize (e.g. with a Runge–Kutta method) whenever the current stepsize has to be modified. Alternatively one may use interpolation formulas like those described above to find a new set of starting values. Both approaches are feasible when stepsize changes are rare events, i.e. when a constant stepsize can be used for most of the integration.

A more flexible solution is obtained by generalizing the Adams formulas of the previous sections. In the case of arbitrary stepsizes the mth-order predictor formula for the computation of the solution at t_{i+1} may be written as

$$\boldsymbol{\eta}_{i+1} = \boldsymbol{\eta}_i + (t_{i+1} - t_i) \cdot \sum_{j=0}^{m-1} g_j(i) \boldsymbol{\phi}_j(i) \quad . \tag{4.80}$$

Here the factors

$$g_j(i) = \frac{1}{t_{i+1} - t_i} \int_{t_i}^{t_{i+1}} \prod_{l=0}^{j-1} \frac{t - t_{i-l}}{t_{i+1} - t_{i-l}} dt \tag{4.81}$$

correspond to the coefficients γ_j of the fixed stepsize formula, while the

$$\boldsymbol{\phi}_j(i) = \prod_{l=0}^{j-1} (t_{i+1} - t_{i-l}) \cdot \boldsymbol{f}[t_i, \ldots, t_{i-j}] \tag{4.82}$$

replace the backward differences $\nabla^j \boldsymbol{f}_i$ (see e.g. Hairer et al. 1987). The expressions $\boldsymbol{f}[t_i, \ldots, t_{i-j}]$ are known as ***divided differences*** and result from the use of Newton's formula for a general interpolation polynomial. They are recursively defined by

$$\begin{aligned} \boldsymbol{f}[t_i] &= \boldsymbol{f}_i \\ \boldsymbol{f}[t_i, t_{i-1}] &= \frac{\boldsymbol{f}_i - \boldsymbol{f}_{i-1}}{t_i - t_{i-1}} \\ \boldsymbol{f}[t_i, t_{i-1}, t_{i-2}] &= \frac{\boldsymbol{f}[t_i, t_{i-1}] - \boldsymbol{f}[t_{i-1}, t_{i-2}]}{t_i - t_{i-2}} \\ &\ldots \end{aligned} \tag{4.83}$$

For constant stepsize h

$$\boldsymbol{f}[t_i, \ldots, t_{i-j}] = \frac{1}{h^j} \nabla^j \boldsymbol{f}_i \quad . \tag{4.84}$$

The coefficients $g_j(i)$ and $\boldsymbol{\phi}_j(i)$ as well as those of a corresponding corrector formula may be calculated from recurrence relations which are essential for an efficient implementation of variable order methods (see e.g. Shampine & Gordon 1975).

For the selection of order and stepsize the error for the order currently in use is estimated as well as the expected error for adjacent orders. At the same time a new stepsize is calculated based on the current error estimate and the current order. Evaluating this information a new order or stepsize can be chosen. Since changes in the stepsize require an increased effort for the computation of the coefficients $g_j(i)$ the stepsize is changed only if the recommended stepsize is larger or smaller than the present one by a factor of at least two.

A great advantage of variable order and stepsize methods is the fact that they do not require a starting procedure. Starting from order one and a very small initial stepsize, both order and stepsize may be increased up to an optimum value within a few steps. This makes variable order and stepsize codes particularly easy to use. Among the various implementations the following are mentioned:

- DVDQ – developed at the Jet Propulsion Laboratory by Krogh (1969, 1974) – is one of the earliest variable order and stepsize multistep codes. DVDQ has, for example, been used for the numerical integration of the solar system ephemeris DE102 (Newhall et al., 1983).

- DE/DEABM is one of the most popular methods of its kind. The code and its theoretical background are explained in detail in the textbook of Shampine & Gordon (1975). The original code DE has since been improved to meet the needs of program libraries (Shampine & Watts 1979) and is available under the name DEABM now. Further amendments of the interpolation routines are reported in Watts & Shampine (1986).

- VOAS is a variable order and stepsize multistep code by Sedgwick (1973). Aside from a different implementation its characteristics and performance are similar to DE/DEABM.

- The predictor–corrector method of Hall & Watts – implemented as D02CJF in the NAG Fortran library – provides interpolation for dense output similar to DE/DEABM.

4.2.6 Stoermer and Cowell Methods

In the discussion of Runge–Kutta methods, Nyström methods have been introduced that are especially designed for the direct integration of second-order differential equations. Corresponding multistep methods that are known as Stoermer and Cowell methods may be derived by an extension of the concept of Adams methods. For this purpose the differential equation

$$\ddot{\boldsymbol{r}} = \boldsymbol{a}(t, \boldsymbol{r}) \tag{4.85}$$

is integrated twice to form the equivalent integral equation

$$\boldsymbol{r}(t_i + h) = \boldsymbol{r}_i + h\dot{\boldsymbol{r}}_i + \int_{t_i}^{t_i+h} \int_{t_i}^{t} \boldsymbol{a}(\tau, \boldsymbol{r}(\tau)) d\tau dt \quad . \tag{4.86}$$

Using partial integration the double integral can be replaced by a single integral:

$$\begin{aligned} \int_{t_i}^{t_i+h} 1 \cdot \int_{t_i}^{t} \boldsymbol{a}(\tau, \boldsymbol{r}(\tau)) d\tau \, dt &= \left(t \cdot \int_{t_i}^{t} \boldsymbol{a}(\tau, \boldsymbol{r}(\tau)) d\tau \right) \Bigg|_{t_i}^{t_i+h} - \int_{t_i}^{t_i+h} t \cdot \boldsymbol{a}(t, \boldsymbol{r}(t)) dt \\ &= \int_{t_i}^{t_i+h} (t_i + h - t) \boldsymbol{a}(t, \boldsymbol{r}(t)) dt \\ &= h^2 \int_0^1 (1-s) \boldsymbol{a}(t_i + sh, \boldsymbol{r}(t_i + sh)) ds \quad . \end{aligned}$$

Table 4.8. Coefficients of Stoermer and Cowell methods in backwards difference notation. See e.g. Schubart & Stumpff (1966) for higher-order coefficients.

j	0	1	2	3	4	5	6	7	8
δ_j	1	0	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{19}{240}$	$\frac{3}{40}$	$\frac{863}{12096}$	$\frac{275}{4032}$	$\frac{33953}{518400}$
δ_j^*	1	-1	$\frac{1}{12}$	0	$\frac{-1}{240}$	$\frac{-1}{240}$	$\frac{-221}{60480}$	$\frac{-19}{6048}$	$\frac{-9829}{3628800}$

By adding $\boldsymbol{r}(t_i + h)$ and $\boldsymbol{r}(t_i - h)$ one can eliminate the velocity $\dot{\boldsymbol{r}}_i$, which results in

$$\begin{aligned}&\boldsymbol{r}(t_i+h) - 2\boldsymbol{r}(t_i) + \boldsymbol{r}(t_i-h)\\ &\quad = h^2 \int_0^1 (1-s)[\boldsymbol{a}(t_i+sh, \boldsymbol{r}(t_i+sh)) + \boldsymbol{a}(t_i-sh, \boldsymbol{r}(t_i-sh))]ds \quad .\end{aligned} \tag{4.87}$$

As in the derivation of the Adams–Bashforth formulas one may now use a polynomial through m points

$$(t_{i-m+1}, \boldsymbol{a}_{i-m+1}), \ldots, (t_i, \boldsymbol{a}_i)$$

to approximate $\boldsymbol{a}$ and to evaluate the integral. This yields the Stoermer formula

$$\boldsymbol{r}_{i+1} = 2\boldsymbol{r}_i - \boldsymbol{r}_{i-1} + h^2 \sum_{j=0}^{m-1} \delta_j \nabla^j \boldsymbol{a}_i \tag{4.88}$$

wi

4.2.7 Gauss–Jackson or Second Sum Methods

The Gauss–Jackson or second sum methods (Jackson 1924, Merson 1974) are slightly modified versions of the Stoermer–Cowell methods for second-order differential equations and probably the most recommendable fixed-stepsize multistep methods for orbit computations. The explicit Stoermer formula is replaced by the equation

$$\boldsymbol{r}_{i+1} = h^2 \sum_{j=0}^{m+1} \delta_j \nabla^{j-2} \boldsymbol{a}_i \tag{4.93}$$

and the modified Cowell formula is given by

$$\boldsymbol{r}_{i+1} = h^2 \sum_{j=0}^{m+1} \delta_j^* \nabla^{j-2} \boldsymbol{a}_{i+1} \quad . \tag{4.94}$$

Velocities at each step may be obtained from similar equations that follow from the Adams–Bashforth–Moulton formulas:

$$\boldsymbol{v}_{i+1} = h \sum_{j=0}^{m} \gamma_j \nabla^{j-1} \boldsymbol{a}_i \tag{4.95}$$

and

$$\boldsymbol{v}_{i+1} = h \sum_{j=0}^{m} \gamma_j^* \nabla^{j-1} \boldsymbol{a}_{i+1} \quad . \tag{4.96}$$

The coefficients γ_j, γ_j^*, δ_j and δ_j^* are listed in Tables 4.4, 4.6 and 4.8.

The expressions for $\boldsymbol{r}_{i+1}$ and $\boldsymbol{v}_{i+1}$ involve the use of first and second sums (∇^{-1}, ∇^{-2}), which are generalizations of the backwards differences introduced earlier. They are implicitly defined by the recursions

$$\begin{array}{lcl} \boldsymbol{a}_i & = & \nabla^{-1}\boldsymbol{a}_i - \nabla^{-1}\boldsymbol{a}_{i-1} \\ \nabla^{-1}\boldsymbol{a}_i & = & \nabla^{-2}\boldsymbol{a}_i - \nabla^{-2}\boldsymbol{a}_{i-1} \end{array} \tag{4.97}$$

in close analogy with definition (4.53). By applying the backward difference operator ∇ twice to the explicit second sum formula (4.93) for $\boldsymbol{r}_{i+1}$, one obtains

$$\boldsymbol{r}_{i+1} - 2\boldsymbol{r}_i + \boldsymbol{r}_{i-1} = h^2 \sum_{j=0}^{m-1} \delta_j \nabla^j \boldsymbol{a}_i \tag{4.98}$$

which is just equation (4.88) of the Stoermer method. In a similar manner one may prove the validity of the implict second sum formulas as well as the first sum formulas for the prediction of $\boldsymbol{v}$.

Despite the apparent equivalence of the Bashforth–Moulton and Stoermer-Cowell formulas and the first and second sum formulas, the latter are generally preferred in practical computations. According to Henrici (1962) and Herrick (1971,

1972) the sum formulas are less influenced by round-off errors that result from the finite computing accuracy. This is especially important in long-term integrations where round-off errors are the main source of error, since the local truncation error can always be limited by choosing a high-order method and/or a small stepsize.

In order to use the summed version of the Stoermer–Cowell formulas one has to determine initial values of the first and second sums in addition to the initial set of backward differences. For starting the calculation it is assumed that one knows the position and velocity $(\boldsymbol{r}_j, \boldsymbol{v}_j)$ of the satellite for a given set of m equidistant times $t_j = t_0 + jh$ $(j = -m+1, \ldots, 0)$. These data can always be obtained from the initial values $(t_0, \boldsymbol{r}_0, \boldsymbol{v}_0)$ by a backwards integration with a high-order Runge–Kutta method or an extrapolation method. From position and velocity one is able to calculate the accelerations $\boldsymbol{a}_j$ and the backward differences $\nabla \boldsymbol{a}_0 \ldots \nabla^{m-1} \boldsymbol{a}_0$. The desired values of the first and second sums can now be determined by solving the implicit Adams–Moulton and Cowell formulas for $\nabla^{-1} \boldsymbol{a}_0$ and $\nabla^{-2} \boldsymbol{a}_0$:

$$\nabla^{-1} \boldsymbol{a}_0 = \frac{\boldsymbol{v}_0}{h} - \sum_{j=1}^{m} \gamma_j^* \nabla^{j-1} \boldsymbol{a}_0 \qquad \nabla^{-2} \boldsymbol{a}_0 = \frac{\boldsymbol{r}_0}{h^2} - \sum_{j=1}^{m+1} \delta_j^* \nabla^{j-2} \boldsymbol{a}_0 \quad . \tag{4.99}$$

As an alternative to using a single-step method for obtaining the initial set of accelerations and backward differences one may use a special starting calculation. It involves an iterative refinement of a crude approximation of the satellite's coordinates and the corresponding difference table that may be based e.g. on the assumption of an unperturbed Keplerian orbit. For a detailed description of this method the reader is referred to Herrick (1971, 1972).

4.2.8 Comparison of Multistep Methods

The relative performance of some of the multistep methods described so far is compared in Fig. 4.6. The test set covers a 6th-order Adams–Bashforth method (AB6), two Adams–Bashforth–Moulton methods of order 8 and 12 (ABM8, ABM12) and the variable order, variable stepsize code DE (Shampine & Gordon 1975), all of which may be used for integrating general first-order differential equations. In addition two high-order Stoermer and Stoermer–Cowell methods (S14, SC14) for the integration of second-order differential equations are included.

When considering fixed-order multistep methods the user must be careful to select a method of appropriate order for a given accuracy requirement. While low-order methods may be inefficient for high accuracies, the higher-order methods are subject to instability at low accuracies (i.e. at large stepsizes). As an example, the ABM12 method can only be used to integrate problem D1 with accuracies of better than 9 digits, whereas the ABM8 method becomes inefficient at just the same accuracy. These problems may be avoided, however, by using a variable order and stepsize method like DE, since the automatic order selection avoids an unstable behavior and, simultaneously, guarantees a high efficiency.

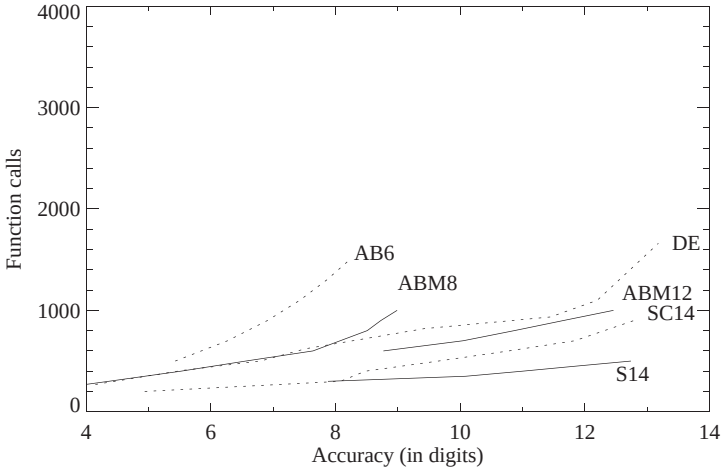


Fig. 4.6. Performance diagram of several multistep methods for test case D1 (e=0.1) of Hull et al. (1972). The number of function calls is plotted versus the relative accuracy in digits.

While the use of a corrector is essential for all but the lowest-order Adams methods, the same is not true for the Stoermer(–Cowell) methods, which are considerably more stable. Predictor methods of the Stoermer type have therefore been preferred by several authors (see e.g. Schubart & Stumpff (1966), Herrick (1971, 1972)) for applications in celestial mechanics, especially for long-term integrations of the solar system. Due to their high stability, Stoermer(–Cowell) methods may be used up to very high orders which makes them the most efficient methods of the test set.

4.3 Extrapolation Methods

The extrapolation method is a powerful single-step method that extends the idea of Richardson extrapolation (i.e. extrapolation to zero stepsize) to the numerical solution of ordinary differential equations. It is often called Bulirsch–Stoer or Gragg–Bulirsch–Stoer method in honor of the pioneering work of Gragg (1965) and Bulirsch & Stoer (1966). A general review of extrapolation methods may be found in Deuflhard (1985).

4.3.1 The Mid-Point Rule

In order to find the solution of a first-order differential equation at some time t_0+H from given initial values $(t_0, \boldsymbol{y_0})$, the interval $[t_0, t_0+H]$ is first subdivided into n (micro-)steps of size $h = H/n$. A simple Euler step is then used to find an approximation $\boldsymbol{u}_1$ at t_0+h, while further values $\boldsymbol{u}_i$ are obtained from the so-called *mid-point rule*:

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0) \\ \boldsymbol{u}_{i+1} &= \boldsymbol{u}_{i-1} + 2h\boldsymbol{f}(t_0+ih, \boldsymbol{u}_i) \qquad (i = 1, \ldots, n-1) \quad . \end{aligned} \tag{4.100}$$