3

Numerical Methods for Particle Tracking

Cristian Marchioli

3.1 Introduction

When dealing with tracking of particles with constant mass, we are interested in solving for the following equation:

$$\frac{\partial \mathbf{v}_p(t)}{\partial t} = \frac{\mathbf{F}[\mathbf{v}_p(t)]}{m_i} \ , \quad \mathbf{v}_p(t_0) = \mathbf{v}_{p,0} \ . \tag{3.1}$$

In general terms, this equation can be regarded as a first order ODE subject to an initial condition:

$$\frac{\partial}{\partial t}[\Phi(t)] = f[\Phi(t)] \ , \quad \Phi(t_0) = \Phi_0 \ ; \tag{3.2}$$

where $\Phi(t)$ is a matrix or array representing a set of variables (*e.g.* the particle velocity components). Thus, particle tracking is an initial value (or Cauchy) problem: first, the solution Φ_1 of the ODE at time $t_1 = t_0 + \Delta t$ has to be found. Φ_1 then becomes the new initial condition for computing Φ_2 at time $t_2 = t_1 + \Delta t$ and so on.

To this object, ODEs are solved numerically by converting derivatives into discrete algebraic expressions. This discretization procedure leads to an algebraic equation, which is manipulated to generate an algorithm for the approximate solution of the ODE. The algorithm gives the approximate solution at the $(n+1) - th$ time step in terms of the known solution at the $n - th$ and earlier time steps.

In this chapter, we review the most common time-marching numerical methods for ODEs.

3.2 Explicit and Implicit Methods

Equation 3.2 can be solved analitically by integration:

$$\int_{\Phi_n}^{\Phi_{n+1}} \mathrm{d}\Phi = \int_{t_n}^{t_{n+1}} f(t, \Phi(t))\ \mathrm{d}t\ . \tag{3.3}$$

However, approximation is required to evaluate the integral on the r.h.s. The *explicit method* (or *Euler forward*) replaces the integral with the initial value of the integrand operator f:

$$\frac{\Phi_{n+1} - \Phi_n}{\Delta t} = f(t_n, \Phi_n)\ . \tag{3.4}$$

The *implicit method* (or *Euler backward*) replaces the integral with the final value of the integrand operator f:

$$\frac{\Phi_{n+1} - \Phi_n}{\Delta t} = f(t_{n+1}, \Phi_{n+1})\ . \tag{3.5}$$

Another common method is the *midpoint rule* (or *modified Euler*), which uses the midpoint of the integration time interval:

$$\frac{\Phi_{n+1} - \Phi_n}{\Delta t} = f(t_{n+1/2}, \Phi_{n+1/2}). \tag{3.6}$$

The schematic in figure 3.1 shows the different procedures used by Euler schemes to approximate the integral on the r.h.s of equation 3.3, in comparison with the case of linear interpolation.

Note that Euler methods are first-order: the order of accuracy of the scheme by which the integration of the equation of particle motion is accomplished and the temporal resolution determine the magnitude of the error incurred at each time step. This error is accumulated over time and the cumulative time-stepping error depends also on the duration of tracking. In the case of Euler methods, the solution at the new time-step is computed with an error proportional to Δt^2, where Δt is the time step size. If N time steps are required to compute the solution at some finite final time $t = t_0 + N \cdot \Delta t$, then the final error is proportional to Δt.

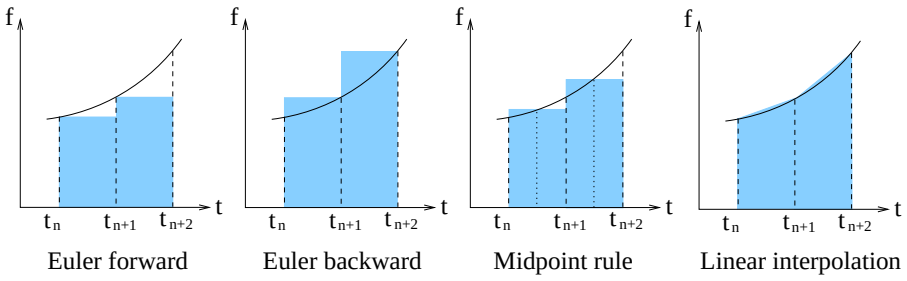

Fig. 3.1. Different procedure for the approximation of the time integral of $f(t, \Phi(t))$ over a finite time interval.

All methods thus produce accurate solutions when the time step size is small. However, many problems in fluid mechanics can only be solved by systems of differential equations that involve a wide range of different time scales. Such problems are said to be *stiff* in a certain interval of integration if the numerical solution in that interval has its step size limited more severely by the stability of the numerical technique than by the accuracy of the technique.

The issue of stability is briefly addressed in Appendix A of this book. The reader is referred to the books by Fletcher and by Ferziger and Peric for further details.

Here, it is important to underline that order of a scheme and stability are two different concepts. For example, the implicit schemes described above are unconditionally stable *i.e.* they yield bounded solutions for every time step if $\partial f(t,\Phi)/\partial\Phi < 0$ but their accuracy can be very limited.

Likewise, order of a scheme and accuracy of a scheme (*i.e.* the truncation error associated with approximating derivatives) are not the same thing. The order of a scheme is generally (tough not always) a reliable guide to the accuracy of that scheme. However, each scheme has its pathological applications which can cause it to break down. The order should be simply regarded as a quantitative measure of the rate at which the error decreases as the time step size decreases. In addition, this is true only when the time step size becomes smaller than a given threshold value, which depends on both the problem to be solved and the scheme used and can not be determined in advance. For time step size larger than the threshold value, the error yielded by two different schemes of the same order may differ by as much as an order of magnitude.

There is of course another requirement for accurate stable integration of the equations of motion. No particle can be allowed to move more than one grid cell in distance during each time step. In other words, the size of the tracking time step must be such that the typical displacement of a particle in any direction is smaller than the grid spacing in the same direction.

3.3 Predictor-Corrector Methods

Explicit Euler methods are generally more easy to program than implicit Euler methods. They also require less computer memory and CPU time per integration step, but they are more unstable. The idea behind *Predictor-Corrector Methods* (PCMs) is to combine the properties of explicit and implicit Euler methods to obtain a method with improved convergence characteristics. The most common PCM predicts the solution at the new time step using the explicit Euler method:

$$\Phi^*_{n+1} = \Phi_n + f(t_n, \Phi_n)\Delta t \ . \tag{3.7}$$

The predicted solution Φ^*_{n+1} is then corrected using the implicit trapezoidal rule (*i.e.* linear interpolation between the initial and the final points):

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{2}\left[f(t_n, \Phi_n) + f(t_{n+1}, \Phi^*_{n+1}\right] \ . \tag{3.8}$$

It can be shown that the highest order of accuracy of such PCM is second-order. For higher orders, a suitable combination of Adams methods can be used (see Section 3.6).

3.4 Crank-Nicolson Methods

Crank-Nicolson Methods (CNMs) are implicit methods which apply the second-order trapezoidal rule to PDEs and ODEs:

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{2}\left[f(t_n, \Phi_n) + f(t_{n+1}, \Phi_{n+1}\right] \ . \tag{3.9}$$

CNMs are commonly used when time accuracy is important. CNMs, Euler methods and PCMs are called two-level methods, since they involve the value of the unknown integral operator at only two time steps. The highest order of accuracy of two-level methods is second order. Higher-order approximations can be obtained by using methods which exploit the information at additional points. In the following, we discuss multi-step methods, which use previously generated solutions and Runge-Kutta methods, which use data at times between t_n and t_{n+1}.

3.5 Runge-Kutta Methods

As mentioned, multi-step methods achieve high order accuracy by efficiently using previously generated solutions. *Runge-Kutta Methods* (RKMs) achieve the same goal in a single step, but at the expense of many evaluations of the derivative per step. Being single-step schemes, RKM are self-starting and thus overcome the difficulties of starting multipoint methods using the specified initial condition. Also, they are more accurate and more stable than multipoint methods of the same order: thus, RKM work well with non-stationary process like particle dispersion studied in a Lagrangian framework.

The general n-step RK scheme applied to equation 3.3 can be written as:

$$\Phi_{n+1} = \Phi_n + \Delta t \sum_{r=1}^{R} c_r f^r \ , \tag{3.10}$$

where:

$$f^r = f(t_n + a_r \Delta t, \Phi_n + \Delta t \sum_{s=1}^{R} b_{rs} f^s) \ , \qquad a_r = \sum_{s=1}^{R} b_{rs} \ . \tag{3.11}$$

Note that b_{rs} are elements of a lower triangular matrix.

RKMs are classified as explicit, implicit and semi-implicit. Here, only the first two will be considered. Implicit schemes guarantee high accuracy and good stability but they are computationally expensive for non-linear initial value problems since they require the iterative solution of a set of non linear algebraic equations for f at each time step. This is much more expensive compared to the explicit schemes, which are easy to program and use less computer memory and computation time per step. Explicit methods, however, suffer from numerical instability when the time step is relatively large. The choice is a trade-off between stability and computational cost, if the schemes have the same order of accuracy. In general, implicit schemes are suitable for stiff differential equations whereas explicit RK schemes are more commonly used when the time step is small.

3.5.1 Second-order Runge-Kutta

The second-order RKM consists of two steps: the first step uses a first-order explicit Euler method to compute $\Phi^*_{n+1/2}$ at half time step, the second step uses the midpoint rule for the full time step to compute Φ_{n+1} at step $t_{n+1} = t_n + \Delta t$:

$$\Phi^*_{n+1/2} = \Phi_n + \frac{\Delta t}{2} f(t_n, \Phi_n) \ , \tag{3.12}$$

$$\Phi_{n+1} = \Phi_n + \Delta t f(t_{n+1/2}, \Phi^*_{n+1/2}) \ . \tag{3.13}$$

3.5.2 Third-order Runge-Kutta

The third-order RKM is derived using a higher order numerical integration scheme and consists of three steps. The first step uses a first-order explicit Euler method to compute $\Phi^*_{n+1/2}$; the second step uses the midpoint rule for the full time step to compute Φ^*_{n+1}. The final step uses the Simpson's rule to correct Φ^*_{n+1} and compute Φ_{n+1}.

$$\Phi^*_{n+1/2} = \Phi_n + \frac{\Delta t}{2} f(t_n, \Phi_n) \ , \tag{3.14}$$

$$\Phi^*_{n+1} = \Phi_n + \Delta t f(t_{n+1/2}, \Phi^*_{n+1/2}) \ , \tag{3.15}$$

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{6} \left[f(t_n, \Phi_n) + 4 f(t_{n+1}, \Phi^*_{n+1}) + \right. \tag{3.16}$$

$$\left. f(t_{n+1}, \Phi^{**}_{n+1}) \right] \ . \tag{3.17}$$

where $\Phi^{**}_{n+1} = \Phi_n + 2(\Phi^*_{n+1} - \Phi^*_{n+1/2})$. A second version is obtained by splitting the integration time step in three parts, as shown in figure 3.2:

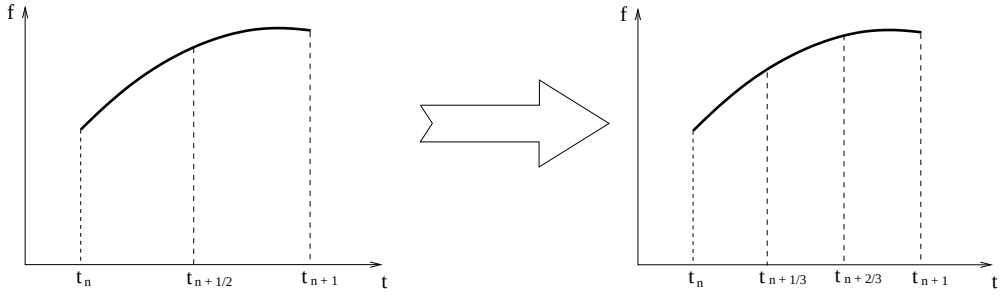

Fig. 3.2. Time step splitting: a) 2nd-order RKM, b) 3rd-order RKM.

$$\Phi^*_{n+1/3} = \Phi_n + \frac{\Delta t}{3} f(t_n, \Phi_n) \ , \tag{3.18}$$

$$\Phi^*_{n+2/3} = \Phi_n + \frac{2\Delta t}{3} f(t_{n+1/3}, \Phi^*_{n+1/3}) \ , \tag{3.19}$$

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{4} \left[f(t_n, \Phi_n) + 3 f(t_{n+2/3}, \Phi^*_{n+2/3}) \right] \ . \tag{3.20}$$

3.5.3 Fourth-order Runge-Kutta

The fourth-order RKM is the most popular among higher-order RKMs and consists of four steps. The first two steps use a first-order explicit Euler predictor and a first-order implicit Euler corrector to compute $\Phi^*_{n+1/2}$ and $\Phi^{**}_{n+1/2}$ at half time step. The third step uses the midpoint rule for the full time step to predict Φ^*_{n+1} which is corrected in the final step by means of the Simpson's 1/3 rule.

$$\Phi^*_{n+1/2} = \Phi_n + \frac{\Delta t}{2} f(t_n, \Phi_n) \ , \tag{3.21}$$

$$\Phi^{**}_{n+1/2} = \Phi_n + \frac{\Delta t}{2} f(t_{n+1/2}, \Phi_{n+1/2}) \ , \tag{3.22}$$

$$\Phi^*_{n+1} = \Phi_n + \Delta t f(t_{n+1/2}, \Phi^{**}_{n+1/2}) \ , \tag{3.23}$$

$$\begin{aligned} \Phi_{n+1} = \Phi_n + \frac{\Delta t}{6} \left[f(t_n, \Phi_n) \right. &+ 2 f(t_{n+1/2}, \Phi^*_{n+1/2}) + \\ &\left. 2 f(t_{n+1/2}, \Phi^{**}_{n+1/2}) + f(t_{n+1}, \Phi^*_{n+1}) \right] \ . \end{aligned} \tag{3.24}$$

A second (less popular) version is based on the Simpson's 3/8 rule and reads:

$$\Phi^*_{n+1/3} = \Phi_n + \frac{\Delta t}{3} f(t_n, \Phi_n) \ , \tag{3.25}$$

$$\Phi^*_{n+2/3} = \Phi_n + \frac{2\Delta t}{3} f(t_{n+1/3}, \Phi^*_{n+1/3}) \ , \tag{3.26}$$

$$\Phi^{**}_{n+2/3} = \Phi^*_{n+2/3} + \frac{\Delta t}{3} \left[f(t_{n+1/3}, \Phi^*_{n+1/3}) - f(t_n, \Phi_n) \right] \ , \tag{3.27}$$

$$\Phi^*_{n+1} = \Phi_n + \Delta t \left[f(t_n, \Phi_n) - f(t_{n+1/3}, \Phi^*_{n+1/3}) + f(t_{n+2/3}, \Phi^{**}_{n+2/3}) \right] \ , \tag{3.28}$$

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{8} \left[f(t_n, \Phi_n) + 3f(t_{n+1/3}, \Phi^*_{n+1/3}) + 3f(t_{n+2/3}, \Phi^{**}_{n+2/3}) + f(t_{n+1}, \Phi^*_{n+1}) \right] \ . \tag{3.29}$$

3.6 Adams Methods

Adams methods are multipoint methods derived by fitting a polynomial to the derivatives at a number of points in time. Explicit Adams methods use data at time t_n in the interpolation polynomial and are known as *Adams-Bashforth Methods* (ABMs). Implicit Adams methods use data at time t_{n+1} and are known as *Adams-Moulton Methods* (AMMs). The first-order explicit/implicit Adams Method is explicit/implicit Euler.

Second-order Adams-Bashforth Method:

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{2} \left[3f(t_n, \Phi_n) - f(t_{n-1}, \Phi_{n-1}) \right] \ . \tag{3.30}$$

Second-order Adams-Moulton Method:

$$\Phi_{n+1} = \Phi_n + \frac{\Delta t}{2} \left[f(t_n, \Phi_n) + f(t_{n+1}, \Phi_{n+1}) \right] \ . \tag{3.31}$$

The second-order AMM is the same as the second-order CNM: thus, it does not use previously computed solution values. The third order formula is more typical because it does involve a previously computed value. For completeness, third-order and fourth-order ABM and AMM are reported (for sake of brevity, we put $f(t, \Phi) = f$).

Third-order and forth-order ABM:

$$(3^{rd}\text{-order}) \ \Phi_{n+1} = \Phi_n + \frac{\Delta t}{12} \left(23f_n - 16f_{n-1} + 5f_{n-2} \right) \ , \tag{3.32}$$

$$(4^{th}\text{-order}) \ \Phi_{n+1} = \Phi_n + \frac{\Delta t}{24} \left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3} \right) \ . \tag{3.33}$$

Third-order and forth-order AMM:

$$(3^{rd}\text{-order})\ \Phi_{n+1} = \Phi_n + \frac{\Delta t}{12}\left(5f_{n+1} + 8f_n - f_{n-1}\right)\ , \quad (3.34)$$

$$(4^{th}\text{-order})\ \Phi_{n+1} = \Phi_n + \frac{\Delta t}{24}\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right)\ . \quad (3.35)$$

Adams methods of order higher than four exist but they are not often used for the solution of ODEs. Interestingly, Adams methods can be combined to obtain PCMs of order higher than second: a common procedure is to use a 3rd/4th order ABM as a predictor and a AMM of the same order as a corrector.

The Adams-Moulton formula is more accurate than the Adams-Bashforth formula of the same order, so that it can use a larger step size; the Adams-Moulton formula is also more stable. A modern code based on Adams methods is relatively easy to program and requires only one evaluation of the derivative per time step. However, it may produce non-physical solutions due to the use of data from several time steps. Another drawback is that an Adams code is more complex than a Runge-Kutta code because it must cope with the difficulties of starting the integration and adapting the time step size. With enough "memorized" values, however, we can use whatever order formula we wish in the step from t_0. Modern Adams codes attempt to select the most efficient formula at each step as well as to choose an optimal step size to achieve a user-specified accuracy.

Some general rules-of-thumb about how to choose between Runge-Kutta methods and Adams methods are given below:

1. If output at many points is needed, Adams methods are generally preferred.
2. If function evaluations are expensive, Adams methods are preferred.
3. If function evaluations are inexpensive and moderate accuracy is required, Runge-Kutta methods are generally best.
4. If storage is at a premium, Runge-Kutta methods are preferred.
5. If accuracy over a wide range of tolerances is needed, the variable order Adams methods will outperform the fixed order Runge-Kutta methods.

3.7 Integration Time Step Size Considerations

The choice of the time step size is crucial in particle tracking: it must be chosen correctly to perform the numerical experiments and to compute the Lagrangian statistics efficiently and accurately.

A theorem, developed by H. Nyquist, states that a signal may be uniquely reconstructed, without error, from samples taken at equal time intervals. The sampling rate (the number of samples taken per unit time, *i.e.* the rate at which the signal is sampled for subsequent use) must be equal to, or greater than, twice the highest frequency component in the signal. If we apply Nyquist's theorem to particle tracking, the tracking time step, Δt_{Tr}, is

the "highest frequency component" and the particle response time τ_p is the "sampling rate" so that it must be:

$$\Delta t_{Tr} \leq \frac{\tau_p}{2} \ , \tag{3.36}$$

in order to obtain an accurate estimate of the particle trajectory, which is the "signal" to be reconstructed.

The magnitude of the time step Δt_{Tr} is bounded not only by the resolution required to compute accurate particle trajectories but also by the available computer disk space. A smaller Δt_{Tr} requires higher storage frequency and larger disk space. On the other hand, accurate particle trajectories need Δt_{Tr} to be smaller than particle characteristic time τ_p. As a consequence, the three-dimensional fluid velocity field needs to be stored at intervals equal to Δt_{Tr}, which is larger than the time step, Δt_{NS}, used in integrating the Navier-Stokes equations for the fluid. A common procedure is to choose $\Delta t_{Tr} = \Delta t_{NS}$ (van Harleem *et al.* 1998).

In their DNS of particle dispersion in a decaying isotropic turbulence, Elghobashi & Truesdell (1992) used $\Delta t_{Tr} = (1/2 \div 1/3)\ \tau_p$ on a 96^3 points grid: the disk space required to store the three fluid velocity components was 10 Mb per time step. Roughly 1 Gb of disk space was required for the complete trajectory of each of the 22^3 tracked particles. A further reduction of the time step size ($\Delta t_{Tr} = 1/4\ \tau_p$) resulted in a negligible difference in the dispersion statistics.

Further comments can be done considering the dependency of the time step size on the character of the turbulence, most likely the integral time scale, the particle's inertia and the particle's settling velocity. Intuitively, we know that, as the Stokes number increases, particles tend not to respond to the acceleration of the surrounding fluid and follow a trajectory quite different than that of the fluid particles. Also, particles with small settling velocity show no preferred direction whereas particles with large settling velocity tend to drift in the direction of the external body force acting on them. In both situations, it is not straightforward to guess which case requires the smallest time step to keep the overall error low (Wang & Stock, 1992).

Results reported in the literature show that:

- the error in the particle location relative to an exact trajectory grows exponentially with time, no matter how small the time step (Wang & Stock, 1992). The smaller the time step size, the longer it takes for the error to become significant.
- if the long-time particle diffusivity is to be calculated, the error in the particle location should be low after several Lagrangian integration times. To this aim, a smaller time step is required with increasing particle Stokes number, St, and settling velocity, $\mathbf{v}_s$. The decrease in the time step size with increasing St (*i.e.* increasing particle mass) is mostly due to the increase in the Lagrangian integration time. The decrease in the time step

size with increasing $\mathbf{v}_s$ is due to the increase in the distance traveled by the particle.

- if the behavior of particle trajectories at a given time after release is of interest, then the time step size limit should be determined for a fixed total Lagrangian integration time. To this aim, larger time steps can be used for particles with larger inertia (*i.e.* larger St), because the trajectories are less random.

The time required to accomplish a simulation for heavy particle dispersion depends on the total Lagrangian integration time, the time step size, the number of particle trajectories computed [1]. In the dispersion simulation by Wang & Stock (1992), 2000 particle trajectories were calculated using 80 Fourier modes: the computation time on a IBM 3090 computer ranged from 2500 to 8000 seconds.

3.8 Application to the Generic Particle Equation of Motion

We next consider the application of some of the numerical schemes described in the previous paragraphs to the equation of motion for a spherical particle subject to drag force only, in the hypotesis of Stokes regime:

$$\frac{\mathrm{d}v_p}{\mathrm{d}t} = \frac{u - v_p}{\tau_p} \tag{3.37}$$

We discretize equation 3.37 and write down the expressions for its solution $v_{p,n+1}$ at the new time step t_{n+1}.

1. Euler explicit:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_n - v_{p,n}\right)$$

2. Euler implicit:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_{n+1} - v_{p,n+1}\right) \rightarrow v_{p,n+1} = \frac{v_{p,n} + \frac{\Delta t}{\tau_p} u_{n+1}}{1 + \frac{\Delta t}{\tau_p}}$$

3. Predictor-corrector:

$$v^*_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_n - v_{p,n}\right)$$

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left[\left(u_n - v_{p,n}\right) + \left(u_{n+1} - v^*_{p,n+1}\right)\right]$$

[1] and the number of Fourier modes to simulate the fluid, if a pseudo-spectral DNS code is used.

4. 2nd-order RK:

$$v^*_{p,n+1/2} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left(u_n - v_{p,n}\right)$$

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_{n+1/2} - v^*_{p,n+1/2}\right)$$

5. 3rd-order RK:

$$v^*_{p,n+1/2} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left(u_n - v_{p,n}\right)$$

$$v^*_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_{n+1/2} - v^*_{p,n+1/2}\right)$$

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{6\tau_p}\left[(u_n - v_{p,n}) + 4(u_{n+1/2} - v^*_{p,n+1/2}) + (u_{n+1} - v^*_{p,n+1})\right]$$

6. 4th-order RK:

$$v^*_{p,n+1/2} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left(u_n - v_{p,n}\right)$$

$$v^{**}_{p,n+1/2} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left(u_{n+1/2} - v^*_{p,n+1/2}\right)$$

$$v^*_{p,n+1} = v_{p,n} + \frac{\Delta t}{\tau_p}\left(u_{n+1/2} - v^{**}_{p,n+1/2}\right)$$

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{6\tau_p}\left[(u_n - v_{p,n}) + 2(u_{n+1/2} - v^*_{p,n+1/2}) + \right.$$
$$\left. + 2(u_{n+1/2} - v^{**}_{p,n+1/2}) + (u_{n+1} - v^*_{p,n+1})\right]$$

7. 2nd-order AM (and 2nd-order CN):

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left[(u_n - v_{p,n}) + (u_{n+1} - v_{p,n+1})\right]$$
$$\rightarrow v_{p,n+1} = \frac{v_{p,n} + \frac{\Delta t}{2\tau_p}(u_{n+1} + u_n - v_{p,n})}{1 + \frac{\Delta t}{2\tau_p}}$$

8. 2nd-order AB:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{2\tau_p}\left[3(u_n - v_{p,n}) - (u_{n-1} - v_{p,n-1})\right]$$

9. 3rd-order AM:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{12\tau_p}\left[5(u_{n+1} - v_{p,n+1}) + 8(u_n - v_{p,n}) - (u_{n-1} - v_{p,n-1})\right]$$
$$\rightarrow v_{p,n+1} = \frac{v_{p,n} + \frac{\Delta t}{12\tau_p}\left[5u_{n+1} + 8(u_n - v_{p,n}) - (u_{n-1} - v_{p,n-1})\right]}{1 + \frac{5\Delta t}{12\tau_p}}$$

10. 3rd-order AB:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{12\tau_p}\left[23(u_n - v_{p,n}) - 16(u_{n-1} - v_{p,n-1}) + 5(u_{n-2} - v_{p,n-2})\right]$$

11. 4th-order AM:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{24\tau_p}\left[9(u_{n+1} - v_{p,n+1}) + 19(u_n - v_{p,n})\right.$$
$$\left.-5(u_{n-1} - v_{p,n-1}) + (u_{n-2} - v_{p,n-2})\right]$$

12. 4th-order AB:

$$v_{p,n+1} = v_{p,n} + \frac{\Delta t}{24\tau_p}\left[55(u_n - v_{p,n}) - 59(u_{n-1} - v_{p,n-1})\right.$$
$$\left.+37(u_{n-2} - v_{p,n-2}) - 9(u_{n-3} - v_{p,n-3})\right]$$

Note that the fluid velocities interpolated at particle position at time $t_{n+1/2}$ and/or t_{n+1}, namely $u_{n+1/2}$ and u_{n+1}, may be required to calculate $v_{p,n+1}$. These velocities are unknown *a priori* and have to be estimated: thus, some approximation is necessary. The easiest (and less accurate) choice is to freeze the fluid velocity field during the time interval between t_n and t_{n+1}, so that $u_n = u_{n+1/2} = u_{n+1}$. This approximation is justified as follows: the time step size Δt_{NS} used in the solution of the Navier-Stokes equation for the fluid is limited by the Courant numerical stability constraint and is typically much smaller than the time step size Δt_{Tr} used in the integration of the particle equation of motion (Kontomaris, Hanratty & McLaughlin 1992). In fact, Δt_{NS} is much smaller than the Kolmogorov fluid time-scale fluctuations based on the volume-averaged viscous dissipation. Another option is to interpolate $u_{n+1/2}$ at particle position at time $t_{n+1/2}$ and u_{n+1} at particle position at time t_{n+1}. To this aim, the particle velocity $v^*_{p,n+1/2}$ and $v^*_{p,n+1}$ predicted by the numerical method can be used.

In order to demonstrate the performance of the different numerical schemes, in figure 3.3, we compare the behavior of the solution of equation 3.37 when Euler methods, PCMs amd Adams methods are used.

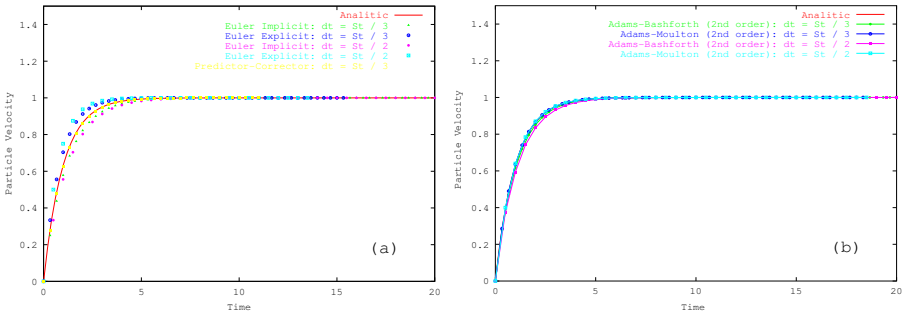

Fig. 3.3. Behavior of the solution by explicit/implicit Euler methods and by second-order PCM (a) and by second-order ABM/AMM (b) using two different time step sizes: $\Delta t = St/2$ and $\Delta t = St/3$.

From figure 3.3a, it is apparent that second-order PCMs produce a more accurate solution than first-order Euler methods. As expected, the implicit Euler method tends to slighlty underpredict whereas the explicit Euler method tends to overpredict the correct value of the solution. Also, the accuracy of Euler methods significantly increases as the time step size decreases. From figure 3.3b, it is apparent that AMMs are more accurate than ABMs of the same order and that the accuracy of both methods increases as the time step size decreases.

Figure 3.4 shows what happens when a numerical scheme is used with time steps which violate the stability condition. We used a fourth-order explicit ABM (figure 3.4a): when the time step size is equal to half the particle Stokes number ($\Delta t = St/2$), oscillations are generated which grow unboundedly with time. In a few steps, the numbers become too large to be handled by the computer. When the computation is carried on with a time step size $\Delta t = St/3$, oscillations are still generated but they grow much slower. Eventually, oscillations disappear with a further reduction of the time step size ($\Delta t = St/4$ and $\Delta t = St/5$ profiles). With an implicit AMM of the same order (figure 3.4b) no problems occurred even for the largest time step used.

Figure 3.5 shows what happens when the time step size is fixed and the order of the numerical scheme is changed. When an explicit ABM is used (figure 3.5a), increasing the order of the method triggers small amplitude oscillations which slowly grow with time. This problem does not occur with the implicit AMM counterpart (figure 3.5b).

Figure 3.6 shows the behavior of the solution when a RKM is used: both the order of the scheme and the time step size are increased. No oscillation in the solution occurs (recall that RMKs are more stable than Adams methods of the same order) but the accuracy significantly decreases as the time step size increases from $\Delta t = St/3$ (figure 3.6a) to $\Delta t = 1.5\ St$ (figure 3.6d). Of course, higher-order RKMs yield more accurate solution for a given time step size.

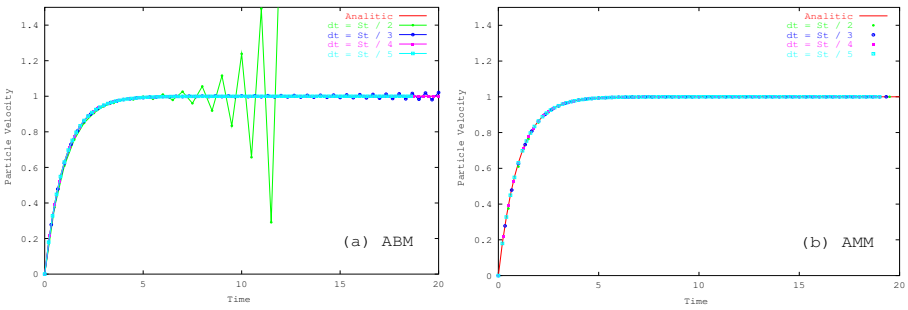

Fig. 3.4. Behavior of the solution by fourth-order ABM (a) and AMM (b) as the time step size Δt is reduced from $\Delta t = St/2$ to $\Delta t = St/5$.

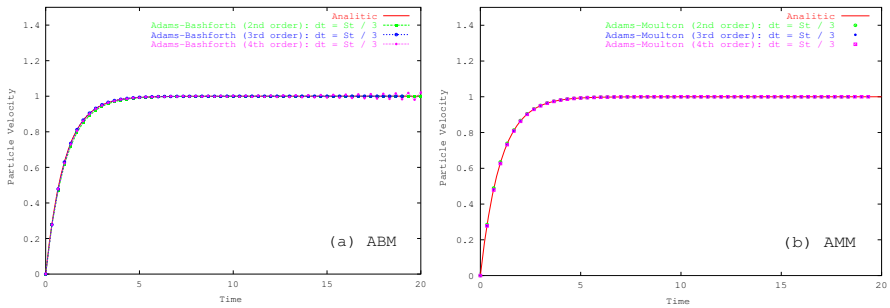

Fig. 3.5. Behavior of the solution by second-, third- and fourth-order ABM (a) and AMM (b) using a time step size $\Delta t = St/3$.

In figure 3.7 the temporal discretization error is shown for various time integration schemes. The discretization error is obtained by subtracting the numerical solution to the exact analytic solution of equation 3.37.

The two Euler methods show the expected first-order behavior: the error and the time step size both reduce by the same amount. The implicit Euler scheme is more accurate than the explicit Euler counterpart but the difference in the solution provided by the two schemes decreases as the time step size decreases.

The second-order schemes show also the expected error reduction rate. The second-order ABM (started by the second-order RKM) yields a larger initial error than both the PCM and the AMM (also started by the second-order RKM) of the same order but the error reduction rate remains the same. Recall that the second-order AMM is the same as the second-order CNM.

A slight difference in the accuracy of the solution occurs when Adams methods of the third-order are used: the ABM provides a slightly larger initial error than AMM but the error reduction rate is again the same as the time step size decreases. We do not show here the estimates of the temporal

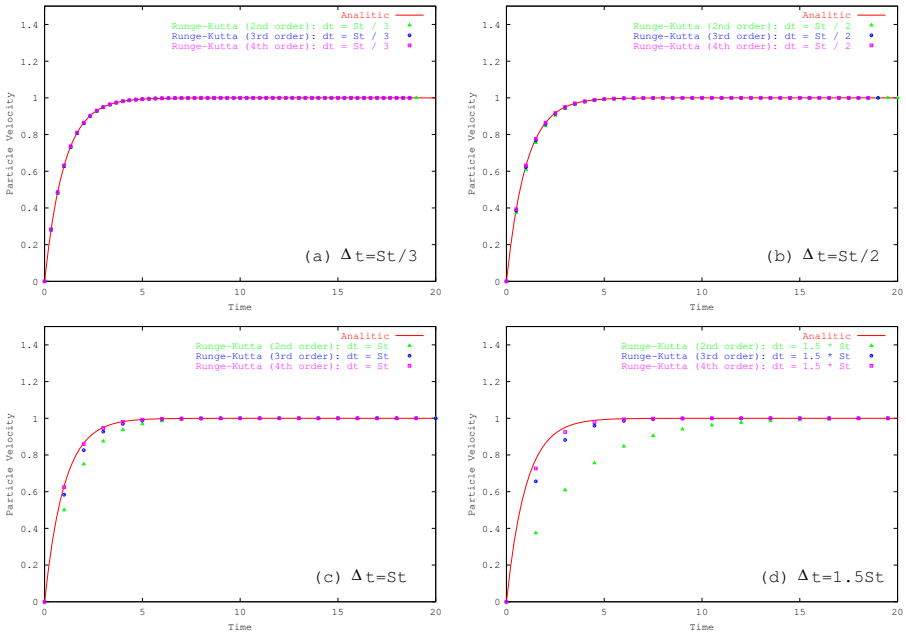

Fig. 3.6. Behavior of the solution by second, third and fourth-order RKM as the time step size Δt is increased from $\Delta t = St/3$ to $\Delta t = 1.5St$.

discretization error for the forth-order Adams methods, due to the oscillatory solution provided by the ABM (see figure 3.4).

The forth-order RKM is by far the most accurate scheme: the error is about four orders of magnitude smaller than that of Euler methods and it is reduced by two orders of magnitude over the time step size reduction considered.

To investigate further the accuracy of temporal discretization, in figure 3.8 we evaluate the convergence of particle velocity as the time step size is reduced. We compare the exact solution of equation 3.37 at time $t = 4.0$ with the numerical solution provided by the schemes already considered in figure 3.7. We performed calculations up to that time using 8, 12, 16 and 20 time steps corresponding to time step sizes of $\Delta t = St/2$, $\Delta t = St/3$, $\Delta t = St/4$, $\Delta t = St/5$ respectively. The convergence rate is consistent with the results shown in figure 3.7. The implicit Euler, the second-order PCM, RKM and ABM underpredict the exact value. The explicit Euler and the second-order AMM (*i.e.* CNM) overpredict the exact value. All schemes show monotonic convergence towards the exact time step independent solution. As expected, the most accurate reference solution is obtained using the forth-order RKM.

To conclude this chapter, we point out that the interpolation error, which mainly depends on the spatial resolution of the small-scale motions of the tubrulent flow (see Chapter 4), is always the major source of numerical errors in the extraction of Lagrangian data. The time-stepping error is generally

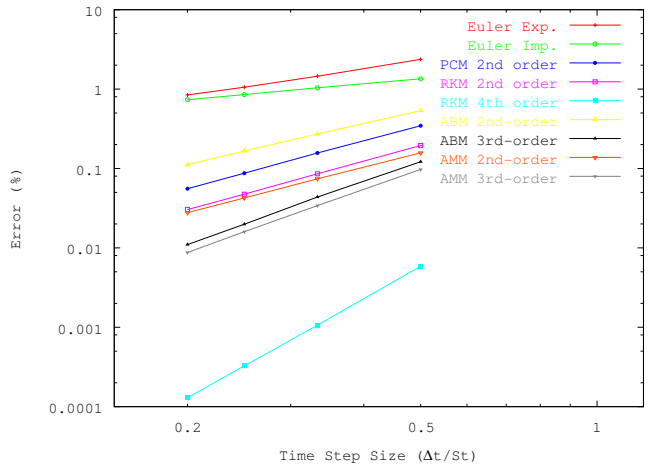

Fig. 3.7. Temporal discretization error for various time integration schemes.

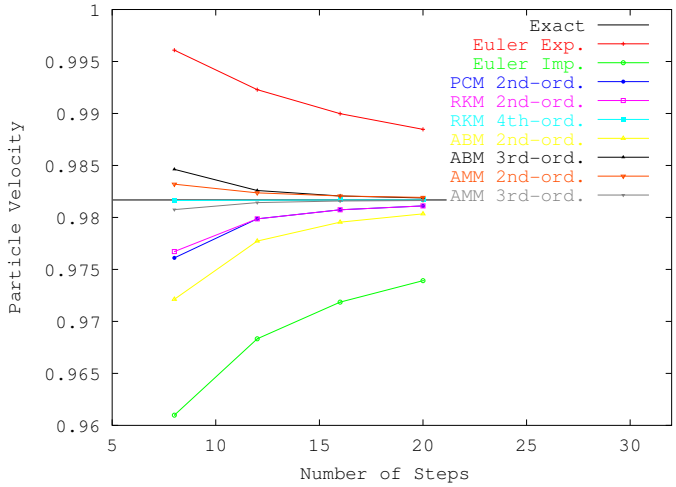

Fig. 3.8. Convergence of v_p at time $t = 4.0$ as the time step size is reduced for various time integration schemes.

less significant because it is restricted to small values by enforcement of the Courant number stability limit (Yeung and Pope, 1988; Yeung and Pope, 1989).

Appendix A

Consider the particle equation of motion in the following form:

$$\frac{\partial v_i^n}{\partial t} = \frac{F(x_i^n)}{m_i} , \qquad \frac{\partial x_i^n}{\partial t} = v_i^n , \tag{3.54}$$

where v_i^n is particle velocity at time t^n at its position x_i^n. The term $F(x_i^n)$ represents the external forces acting on the particle of mass m_i. Combining the above equations into one we get:

$$\frac{\partial^2 x_i^n}{\partial t^2} = \frac{F(x_i^n)}{m_i} , \tag{3.55}$$

where x_i^n is the numerical solution at time t^n. Now, let X_i^n be the exact solution of equation 3.56 at time t^n, that is the solution without round-off error [2]. The numerical error at time t^n can be defined as:

$$\epsilon_i^n = x_i^n - X_i^n . \tag{3.56}$$

Using equation 3.57 to replace x in equation 3.56, an equation for the time evolution of the error ϵ^n is obtained:

$$\frac{\partial^2 \epsilon_i^n}{\partial t^2} = \frac{F(X_i^n + \epsilon_i^n)}{m_i} - \frac{\partial^2 X_i^n}{\partial t^2} . \tag{3.57}$$

Observe that:

$$\frac{\partial^2 X_i^n}{\partial t^2} = \frac{F(X_i^n)}{m_i} , \tag{3.58}$$

and that:

$$F(X_i^n + \epsilon_i^n) - F(X_i^n) = \left.\frac{\partial F}{\partial X}\right|_{X_i^n} \cdot \epsilon_i^n , \tag{3.59}$$

in the limit $\epsilon_i^n \to 0$. Thus, equation 3.58 can be rewritten as:

$$\frac{\partial^2 \epsilon_i^n}{\partial t^2} = \frac{1}{m_i} \left.\frac{\partial F}{\partial X}\right|_{X_i^n} \cdot \epsilon_i^n . \tag{3.60}$$

Assume to approximate the l.h.s. of equation 3.61 by means of a simple central-difference three time-level scheme (also known as leapfrog method [3]):

[2] The round-off error is the error introduced because the computer only stores numbers up to a certain precision.

[3] The leapfrog method is the application of the midpoint rule to an integration interval of size $2\Delta t$

$$\frac{\epsilon_i^{n+1} - 2\epsilon_i^n + \epsilon_i^{n+1}}{\Delta t^2} = \frac{1}{m_i} \left.\frac{\partial F}{\partial X}\right|_{X_i^n} \cdot \epsilon_i^n \ . \tag{3.61}$$

For bounded oscillatory solutions of the form:

$$\epsilon_i^n = (\lambda)^n = (e^{i\omega\Delta t})^n \ , \tag{3.62}$$

equation 3.61 can be recast as:

$$\lambda^2 - \lambda \cdot \left(2 + \frac{\Delta t^2}{m_i} \left.\frac{\partial F}{\partial X}\right|_{X_i^n} \right) + 1 = 0 \tag{3.63}$$

Assuming $\frac{\Delta t^2}{m_i} \cdot \left.\frac{\partial F}{\partial X}\right|_{X_i^n} = \beta$, this equation has two solutions:

$$\lambda_{1,2} = 1 + \frac{\beta}{2} \cdot \left(1 \pm \sqrt{1 + \frac{4}{\beta}} \right) \ , \tag{3.64}$$

which correspond to the error amplification factor. The general solution is:

$$\epsilon_i^n = C_1 \cdot \lambda_1 + C_2 \cdot \lambda_2 \ . \tag{3.65}$$

The scheme is said to be *conditionally stable* provided $|\lambda_{1,2}| \leq 1$. Figure 3.9 shows the behavior of λ_1 and λ_2 as a function of $i\sqrt{\beta}$. When $i\sqrt{\beta} < 2$, λ_1 and λ_2 have an imaginary part, but for $i\sqrt{\beta} \geq 2$ both solutions are real. Complex values must be considered because higher order systems may exhibit complex eigenvalues. Values with zero or negative real part must be considered because they lead to bounded solutions.

For $i\sqrt{\beta} < 2$, it can be shown that $|\lambda_{1,2}| = 1$: not only is the leapfrog scheme stable but also it suffers no amplitude dissipation. When $i\sqrt{\beta} > 2$, $|\lambda_2| > 1$: to guarantee stability, we must calculate the largest value of $\mid m_i^{-1} \partial F / \partial X \mid$ and then set Δt such that:

$$\Delta t < \frac{2}{\sqrt{\left| \frac{1}{m_i} \frac{\partial F}{\partial X} \right|}} \ . \tag{3.66}$$

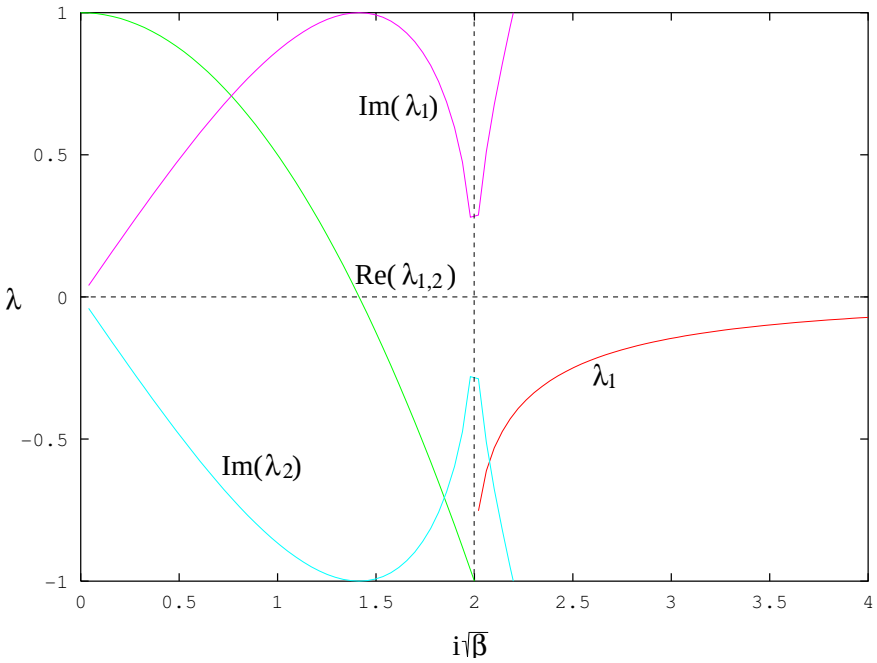

Fig. 3.9. The roots of equation 3.9 as a function of $i\sqrt{\beta}$. For $i\sqrt{\beta} < 2$ both roots have an imaginary component, but both have magnitude $|\lambda_{1,2}| < 1$. For $i\sqrt{\beta} < 2$ both roots are real and $|\lambda_2| < 1$.

3.9 Bibliography

1. Elghobashi, S. & Truesdell, G.C. (1992). Direct simulation of particle dispersion in a decaying isotropic turbulence. *J. Fluid Mech.*, **242**, 655-700.
2. Ferziger, J.H. & Peric, M. (1997). *Computational Methods for Fluid Dynamics*. Springer-Verlag.
3. Fletcher, C.A.J. (2000). *Computational Techniques for Fluid Dynamics*. Springer-Verlag.
4. van Haarlem, B., Boersma, B.J. & Nieuwstadt, T.M. (1998). Direct numerical simulation of particle deposition onto a free-slip and no-slip surface. *Phys. Fluids*, **10**, 2608-2620.
5. Kontomaris, K., Hanratty, T.J. & McLaughlin, J.B. (1992). An algorithm for tracking fluid particles in a spectral simulation of turbulent channel flow. *J. Comput. Phys.*, **103**, 231-242.
6. Wang, L.P. & Stock, D.E. (1992). Numerical simulation of heavy particle dispersion time step and nonlinear drag considerations. *J. Fluids Eng.*, **114**, 100-106.

7. Yeung, P.K. & Pope, S.B. (1988). An algorithm for tracking fluid particles in numerical simulation of homogeneous turbulence. *J. Comput. Phys.*, **79**, 373-416.
8. Yeung, P.K. & Pope, S.B. (1989). Lagrangian statistics from direct numerical simulations of isotropic turbulence. *J. Fluid Mech.*, **207**, 531-586.