Computational methods for particle tracking in isotropic turbulence

van Hinsberg, M.A.T.

Award date:
2011

Computational methods for particle tracking in isotropic turbulence

M.A.T. van Hinsberg

R-1777-A

10 Januari, 2011

Supervisors : Dr. Ir. J.H.M. ten Thije Boonkkamp
Dr. Ir. B.J.H. van de Wiel
Prof. Dr. H.J.H. Clercx

Abstract

The hydrodynamic force exerted by a fluid on small isolated rigid spherical particles are usually well described by the Maxey-Riley (MR) equations. In order to be able to simulate a large number of particles the methods for solving the MR equations need to be fast and accurate. The most time-consuming contribution in the MR equations is the Basset history force which is a well-known problem for many-particle simulations in turbulence. In this study a novel numerical approach is proposed for the computation of the Basset history force based on the use of exponential functions to approximate the tail of the Basset force kernel. Typically, this approach not only decreases the cpu time and memory requirements for the Basset force computation by more than one order of magnitude, but also increases the accuracy by an order of magnitude. The method has a temporal accuracy of $\mathcal{O}\left(\Delta t^2\right)$ which is a substantial improvement compared to methods available in the literature. Furthermore, the method is partially implicit in order to increase stability of the computation.

Another important aspect in numerical simulations of particle laden turbulent flows is the interpolation of the flow field. For the interpolation many different approaches are used. Where some studies use low order linear interpolation others use high order spline methods. This study focuses on estimating the error made by the interpolation method and compares it with the error made in the discretisation of the flow field. In this way one can balance the errors in order to achieve an optimal result. As a spin-off, a practical method is proposed that enables direct estimation of the interpolation error from the energy spectrum. Furthermore, it is shown how the energy spectrum is changed due to the interpolation. Because high order interpolation methods are computationally expensive, a numerical method is proposed for fast computation.

A third part of the present study considers the Faxén corrections which are first order corrections in the MR equations for increasing particle radii. In most studies these corrections have been implemented by using the Laplacian of the flow field. Recently it has been shown that using volume and surface averages for the calculation of the Faxén corrections give more reliable results for larger particle radii. In this study a method is proposed that does the averaging exactly in spectral space. Besides that our method is more accurate it is also faster.

Contents

Chapter 1

Introduction

The increase in computational power has made it possible to tackle more complex problems. One of these complex problems involves the study of particle transport in turbulent flows. As a result of the wide availability of computational power a dramatic increase in Lagrangian computations with dispersed particles is recently observed [1]. The study of particle transport in turbulent flow is of practical importance. For example in the atmosphere various types of particles can be encountered. A topic that receives a lot of attention nowadays is air pollution in industrialized areas by high concentrations of aerosols that affect human health and weather systems. Also in oceans several types of particles are transported. One could think of algae and plankton. An interesting aspect is the fact that some algae can adjust their density as a function of the received light intensity. As such they become active particles [2, 3].

Maxey and Riley [4] introduced the equations of motion for small ($d_p \ll \eta$, with d_p the particle diameter and η the Kolmogorov length scale) isolated rigid spherical particles in a non-uniform velocity field $\mathbf{u}(\mathbf{x}, t)$. An important assumption is that the particle Reynolds number $Re_p = d_p|\mathbf{u} - \mathbf{u}_p|/\nu \ll 1$, with $\mathbf{u}_p$ the velocity of the particle and ν the kinematic viscosity of the fluid. As we consider small particle diameters and small volume fractions of particles we ignore the effect of two-way and four-way coupling. When considering two-way coupling also fluid-particle interactions are incorporated. Four-way coupling includes particle-particle interactions as well. Time integration of the hydrodynamic force in order to compute particle trajectories is an expensive, time- and memory consuming job. In this study three different aspects are investigated. The three different aspects are: computation of the Basset history force, interpolation of the velocity field and the computation of the Faxén corrections. The different aspects of this study are presented in article style, which makes some repetition unavoidable. Chapter 1 deals with the Basset history force and this part of the study is completed. It is accepted for publication in "Journal of Computational Physics" [5]. Chapter 2 deals with the interpolation. This part of the study is not yet completed. The basis for this article is ready but more simulations need to be done in order to finish this part. The text is not yet up to the desired level of a journal paper due to the limited time available. The goal is to submit this part as well, when completed. Finally, a method for the implementation of the Faxén corrections is proposed. Although this part of the study is far from completed, it provides an attractive starting point for further research. We have therefore decided to include these results in Appendix A and not in a separate chapter. Next we will give a short introduction to the three different aspects.

Let us first reconsider the Basset history force, the term most often neglected is the Basset history force because of its numerical complexity. Many recent studies underline the importance of the Basset force compared to the other forces contributions in the Maxey-Riley equations for particle transport in turbulent flows, see Refs. [6, 7, 8, 9]. Moreover, it can affect the motion of a sedimenting particle [10] or bed-load sediment transport in open channels, where the Basset force becomes extremely important for sand particles [11, 12]. It also might alter the particle velocity in an oscillating flow field [13] or modify the trapping of particles in vortices [14]. Fast and accurate computation of the Basset force is far from trivial. Although several attempts have been made

[15, 16, 17], the computation of the Basset force is still far more time consuming and less accurate than the computation of the other forces in the MR equations. Therefore in Chapter 2 we present a new method that saves time, memory costs and is more accurate.

Second, the interpolation step can be very time consuming and memory demanding as well. Some studies use low order linear interpolation, where others use high order spline interpolations [18, 19]. These high order spline interpolations have the disadvantage of being more computational expensive. Therefore, it is important that they give a significant more accurate result. In order to choose the most appropriate interpolation method for a particular case one needs to compare errors: the interpolation error is compared with the discretisation error of the flow field. In this way one can prevent unnecessary computations. This important aspect is however not often taken into account. In order to estimate the interpolation error different methods are proposed. Further we introduce a general framework for different interpolation methods. Making use of this framework an algorithm is proposed for fast computation of the interpolation. This can easily save an order of magnitude in computing time compared with other algorithms.

Finally we propose a method for the implementation of the Faxén corrections. The Faxén corrections are first order corrections in the MR equations for increasing particle radii. Recently it has been shown that using volume and surface averages for the calculation of the Faxén corrections allows for larger particle radii [20, 21]. In Appendix A a method is proposed that does the averaging exactly in spectral space. Besides that the method is more accurate it is also faster.

Chapter 2

An efficient, second order method for the approximation of the Basset history force

2.1 Introduction

The turbulent dispersion of small inertial particles plays an important role in environmental flows, and in this work we focus on small particles with densities of the same order as that of the surrounding fluid. Examples of such particles that may be present in well-mixed or in density stratified estuaries are plankton, algae, aggregates (all with densities similar to the fluid density) or resuspended sand from the sea bottom (particle densities in this case several times that of the fluid). Particle collisions and the formation of aggregates of marine particles or sediment depend on the details of the small-scale trajectories of the particles in locally homogeneous and isotropic turbulence. At these scales the details of the hydrodynamic force acting on (light) inertial particles are relevant.

Maxey and Riley [4] introduced the equation of motion for small ($d_p \ll \eta$, with d_p the particle diameter and η the Kolmogorov length scale) isolated rigid spherical particles in a non-uniform velocity field $\mathbf{u}(\mathbf{x}, t)$. An important assumption is that the particle Reynolds number $Re_p = d_p|\mathbf{u} - \mathbf{u}_p|/\nu \ll 1$, with $\mathbf{u}_p$ the velocity of the particle and ν the kinematic viscosity of the fluid. As we consider small particle diameter and small volume fraction of particles we ignore the effect of two-way and four-way coupling. The relative importance of the terms in the hydrodynamic force depends on the ratio of particle-to-fluid density and the particle diameter. The computation of all the different forces in the Maxey-Riley equation is an expensive time- and memory consuming job. Therefore, assumptions are often made regarding the forces that can be neglected in the study of particle dispersion. The number of studies underpinning these assumptions, however, is rather limited due, for example, to the lack of efficient algorithms to take into account the effects of the Basset history force with sufficient numerical accuracy. This term was first discovered by Boussinesq in 1885. An elaborate overview of the work on the different terms in the Maxey-Riley equation and their numerical implementation can be found in the paper by Loth [22] and a historical account of the equation of motion was given in a review article by Michaelides [23].

The term most often neglected is the Basset history force because of its numerical complexity. Many recent studies underline the importance of the Basset force compared to the other forces contributions in the Maxey-Riley equation for particle transport in turbulent flows, see Refs. [6, 7, 8, 9]. Moreover, it can affect the motion of a sedimenting particle [10] or bed-load sediment transport in open channels, where the Basset force becomes extremely important for sand particles [11, 12]. It also might alter the particle velocity in an oscillating flow field [13] or modify the trapping of particles in vortices [14].

Fast and accurate computation of the Basset force is far from trivial. Although several attempts have been made [15, 16, 17], the computation of the Basset force is still far more time consuming and less accurate than the computation of the other forces in the MR equation. Therefore we present a new method that saves time, memory costs and is more accurate.

The MR equation and the subtlities with regard to the computation of the Basset history force are introduced in Section 2.2. Next, in Section 2.3 and 2.4, the new method is introduced, where Section 2.3 focuses on the approximation of the tail of the Basset history force and Section 2.4 on the numerical integration of the Basset history force. Thereafter, validation of the method using analytical solutions is discussed in Section 2.5. A simulation of isotropic turbulence, with light inertial particles embedded in the flow, has been performed. In Section 2.6 we compare the results from this simulation with the new implementation of the full MR equation with the old version used by van Aartrijk and Clercx [8]. Finally, concluding remarks are given in Section 2.7.

2.2 Particle tracking

Particle trajectories in a Lagrangian frame of reference satisfy

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{u}_p, \tag{2.1}$$

with $\mathbf{x}_p$ the particle position and $\mathbf{u}_p$ its velocity. According to Maxey and Riley [4] the equation of motion for an isolated rigid spherical particle in a nonuniform velocity field $\mathbf{u}$ is given by

$$\begin{aligned} m_p\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{d}t} &= 6\pi a\mu\left(\mathbf{u}-\mathbf{u}_p+\frac{1}{6}a^2\nabla^2\mathbf{u}\right)+m_f\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t}-(m_p-m_f)g\mathbf{e}_z \\ &\quad +\frac{1}{2}m_f\left(\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t}-\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{d}t}+\frac{1}{10}a^2\frac{\mathrm{d}}{\mathrm{d}t}\left(\nabla^2\mathbf{u}\right)\right)+6a^2\rho\sqrt{\pi\nu}\int_{-\infty}^{t}K_{\mathrm{B}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau \\ &= \mathbf{F}_{\mathrm{St}}+\mathbf{F}_{\mathrm{P}}+\mathbf{F}_{\mathrm{G}}+\mathbf{F}_{\mathrm{AM}}+\mathbf{F}_{\mathrm{B}}. \end{aligned} \tag{2.2}$$

The equation of motion includes time derivatives of the form d/dt taken along the particle path and derivatives of the form D/Dt taken along the path of a fluid element. The particle mass is given by m_p, a is the radius of the particle, $\mu = \rho\nu$ is the dynamic viscosity, ρ and ν are the density of the fluid and its kinematic viscosity, m_f is the mass of the fluid element with a volume equal to that of the particle and $\mathbf{e}_z$ is the unit vector in the opposite direction of the gravitational force. The forces in the right-hand side of this equation denote the Stokes drag, local pressure gradient in the undisturbed fluid, gravitational force, added mass force and the Basset history force, respectively. The Faxén correction proportional to $\nabla^2\mathbf{u}$ has been included in the Stokes drag, added mass and Basset force [24]. According to Homann *et al.* [21] these corrections reproduce dominant finite-size effects on velocity and acceleration fluctuations for neutrally buoyant particles with diameter up to four times the Kolmogorov scale η. For the added mass term the form described by Auton *et al.* [25] is used. Moreover, the history force convolution function $\mathbf{g}(t)$ and its kernel are

$$\mathbf{g}(t) = \frac{\mathrm{d}\mathbf{f}(t)}{\mathrm{d}t}\ , \qquad \mathbf{f}(t) = \mathbf{u}-\mathbf{u}_p+\frac{1}{6}a^2\nabla^2\mathbf{u}\ , \qquad K_{\mathrm{B}}(t) = \frac{1}{\sqrt{t}}. \tag{2.3}$$

Equation (2.2) is valid when $a \ll \eta$, but, as mentioned above, the Faxén correction can weaken this condition. Furthermore, the particle Reynolds number must be small ($Re_p \ll 1$), as are the velocity gradients around the particle. Finally, the initial velocity of the particle and fluid must be equal, if this is not the case a second term appears in the Basset history force [17]. The coupled system (2.1) and (2.2) is in principle suitable for integration by any standard method, e.g. the fourth order Runge-Kutta method.

The Basset history force $\mathbf{F}_{\mathrm{B}}$ presents additional challenges. First, the evaluation of the Basset force can become extremely time consuming and memory demanding. This is due to the fact that

every time step an integral must be evaluated over the complete history of the particle. Several attempts have been made to solve this problem. Michaelides [17] uses a Laplace transform to find a novel way for computing the Basset force. This procedure can be used for linear problems, but is not suitable for space dependent velocity fields for which the coupled system (2.1) and (2.2) is nonlinear. Another solution is provided by Dorgan and Loth [16] and Bombardelli *et al.* [15]. In these papers the integral is evaluated over a finite window from $t - t_{\text{win}}$ until t. This can be represented by a change in the kernel of the Basset force. The window kernel is thus defined as

$$K_{\text{win}}(t) = \begin{cases} K_{\text{B}}(t) & \text{for} \quad t \leq t_{\text{win}}, \\ 0 & \text{for} \quad t > t_{\text{win}}. \end{cases} \tag{2.4}$$

The kernel of the Basset force is decreasing very slowly for $t \to \infty$, thus t_{win} must be chosen rather large. For Bombardelli *et al.* [15] this problem turned out to be less important because they used a different kernel, which decreases faster for $t \to \infty$. Although the application of the window kernel saves CPU time, the computation of the Basset force is still far more expensive than the evaluation of the other forces in the MR equation. It turns out to be approximately 100 to 1000 times more time consuming depending on the application.

A second issue concerns the kernel of the Basset force, which is singular for $t \to 0$. A standard approach to deal with the singularity of the Basset kernel is to employ specific quadrature rules such as the second order Euler-Maclaurin formula [26]. Another approach is presented by Tatom [27] who uses a fractional derivative method. This approach was tested by Bombardelli *et al.* [15]. From their results it can be easily shown that the integration method with specific quadrature rules has only temporal accuracy $\mathcal{O}(\sqrt{\Delta t})$ and that the fractional derivative approach has a temporal accuracy $\mathcal{O}(\Delta t)$. In computations of turbulent flows with particles, other discretization methods involved are at least second order. Therefore, it is not sufficient to have a first order integration method for the Basset force.

Our goal is to derive a robust and efficient method for the computation of the Basset force that overcomes all the problems mentioned above and to find an approach that is suitable for different forms of the kernel. Furthermore, our method must be stable and at least second order accurate in time. A third requirement is that it should be less time consuming and memory demanding than previous methods.

2.3 Approximation of the tail of the Basset force

To get a better understanding of the Basset force we will first show that the contribution of this force is finite at any given time. To do this, some restrictions on $\mathbf{f}(t)$ and $\mathbf{g}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}(t)$ should be made. First, $\mathbf{f}(t)$ must be a continuous function and its derivative must exist almost everywhere. Further, $\mathbf{f}(t)$ and $\mathbf{g}(t)$ must be in the L^∞space with norm B_1 and B_2, respectively. The restrictions on $\mathbf{f}(t)$ and $\mathbf{g}(t)$ are thus:

$$\mathbf{f} \in C^0, \qquad \|\mathbf{f}\|_\infty = B_1, \qquad \|\mathbf{g}\|_\infty = B_2, \tag{2.5}$$

where $\|\cdot\|_\infty$ is defined as:

$$\|\mathbf{f}\|_\infty = \inf\{C \geq 0 : |\mathbf{f}(t)| \leq C \text{ almost everywhere}\}, \tag{2.6}$$

and $|\cdot|$ is the usual length of the vector. We assume that for particles in (turbulent) flows with $\mathbf{f}(t) = \mathbf{u} - \mathbf{u}_p + \frac{1}{6}a^2\nabla^2\mathbf{u}$ these conditions are satisfied as both the flow field and its Laplacian satisfy these conditions. With the conditions in (2.5) it is possible to find an upper bound for $\mathbf{F}_B$. The integral is split into two parts, in order to control both the singularity in the Basset kernel

and the tail of the integral. This yields

$$\begin{aligned}
\left|\frac{\mathbf{F}_{\mathrm{B}}}{c_{\mathrm{B}}}\right| &= \left|\int_{-\infty}^{t} K_{\mathrm{B}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau\right| \\
&= \left|\int_{-\infty}^{t-\frac{B_1}{B_2}} \frac{\mathbf{g}(\tau)}{\sqrt{t-\tau}}\mathrm{d}\tau + \int_{t-\frac{B_1}{B_2}}^{t} \frac{\mathbf{g}(\tau)}{\sqrt{t-\tau}}\mathrm{d}\tau\right| \\
&\leq \left|\left[\frac{\mathbf{f}(\tau)}{\sqrt{t-\tau}}\right]_{-\infty}^{t-\frac{B_1}{B_2}} - \int_{-\infty}^{t-\frac{B_1}{B_2}} \frac{\mathbf{f}(\tau)}{2(t-\tau)^{3/2}}\mathrm{d}\tau\right| + \int_{t-\frac{B_1}{B_2}}^{t} \frac{|\mathbf{g}(\tau)|}{\sqrt{t-\tau}}\mathrm{d}\tau \\
&\leq \sqrt{B_1 B_2} + \frac{B_1}{2}\int_{-\infty}^{t-\frac{B_1}{B_2}} \frac{1}{(t-\tau)^{3/2}}\mathrm{d}\tau + B_2\int_{t-\frac{B_1}{B_2}}^{t} \frac{1}{\sqrt{t-\tau}}\mathrm{d}\tau \\
&= 4\sqrt{B_1 B_2}. \qquad (2.7)
\end{aligned}$$

Here $c_{\mathrm{B}} = 6a^2\rho\sqrt{\pi\nu}$ is introduced for convenience. We now consider the window kernel for calculation of the Basset force $\mathbf{F}_{\text{B-win}}$. In the limit of $t_{\mathrm{win}} \to \infty$ the difference between $\mathbf{F}_{\mathrm{B}}$ and $\mathbf{F}_{\text{B-win}}$ must vanish. Using integration by parts, one can derive

$$\begin{aligned}
\left|\frac{\mathbf{F}_{\mathrm{B}} - \mathbf{F}_{\text{B-win}}}{c_{\mathrm{B}}}\right| &= \left|\int_{-\infty}^{t} K_{\mathrm{B}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau - \int_{-\infty}^{t} K_{\mathrm{win}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau\right| \\
&= \left|\int_{-\infty}^{t-t_{\mathrm{win}}} \frac{\mathbf{g}(\tau)}{\sqrt{t-\tau}}\mathrm{d}\tau\right| \leq \frac{2B_1}{\sqrt{t_{\mathrm{win}}}}. \qquad (2.8)
\end{aligned}$$

The error made by using the window kernel instead of the Basset kernel is indeed becoming negligibly small for $t_{\mathrm{win}} \to \infty$. Unfortunately, this convergence is very slow, implying that t_{win} must be very large, and a better approach for the computation of the Basset force must be found. This is done by introducing a new kernel with a modified tail, in short the modified Basset kernel $K_{\mathrm{mod}}(t)$, as follows

$$\begin{aligned}
K_{\mathrm{mod}}(t) &= \begin{cases} K_{\mathrm{B}}(t) & \text{for} \quad t \leq t_{\mathrm{win}} \\ K_{\mathrm{tail}}(t) & \text{for} \quad t > t_{\mathrm{win}} \end{cases} \\
&\lim_{t\to\infty} K_{\mathrm{tail}}(t) = 0. \qquad (2.9)
\end{aligned}$$

This new kernel also implies a modified history force denoted by $\mathbf{F}_{\text{B-mod}}$. For now $K_{\mathrm{tail}}(t)$ is not yet defined but must be chosen such as to approximate the Basset kernel as close as possible. Using integration by parts in the last step, the upper bound for the error induced by the modified Basset force $\mathbf{F}_{\text{B-mod}}$ becomes:

$$\begin{aligned}
\left|\frac{\mathbf{F}_{\mathrm{B}} - \mathbf{F}_{\text{B-mod}}}{c_{\mathrm{B}}}\right| &= \left|\int_{-\infty}^{t} K_{\mathrm{B}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau - \int_{-\infty}^{t} K_{\mathrm{mod}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau\right| \\
&= \left|\int_{-\infty}^{t-t_{\mathrm{win}}} (K_{\mathrm{B}} - K_{\mathrm{tail}})(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau\right| \\
&\leq B_1\left\{\left|K_{\mathrm{B}}(t_{\mathrm{win}}) - K_{\mathrm{tail}}(t_{\mathrm{win}})\right|\right. \\
&\quad \left. + \int_{t_{\mathrm{win}}}^{\infty}\left|\frac{\mathrm{d}(K_{\mathrm{B}} - K_{\mathrm{tail}})(t)}{\mathrm{dt}}\right|\mathrm{d}t\right\}. \qquad (2.10)
\end{aligned}$$

As the upper bound in relation (2.10) depends on t_{win}, it turns out to be beneficial to rescale the time and kernel as follows:

$$\widetilde{K}_{\mathrm{tail}}(\widetilde{t}) = \frac{K_{\mathrm{tail}}(t)}{K_{\mathrm{B}}(t_{\mathrm{win}})}, \qquad \widetilde{t} = \frac{t}{t_{\mathrm{win}}}. \qquad (2.11)$$

Applying the same scaling to $K_{\mathrm{B}}(t) = 1/\sqrt{t}$ we find

$$\widetilde{K}_{\mathrm{B}}(\widetilde{t}) = \frac{K_{\mathrm{B}}(t)}{K_{\mathrm{B}}(t_{\mathrm{win}})} = K_{\mathrm{B}}(\widetilde{t}). \tag{2.12}$$

Note that this cannot be done for a general kernel. Eq. (2.10) can now be reformulated as

$$\left| \frac{\mathbf{F}_{\mathrm{B}} - \mathbf{F}_{\text{B-mod}}}{c_{\mathrm{B}}} \right| \leq \frac{B_1}{\sqrt{t_{\mathrm{win}}}} \left\{ \left| 1 - \widetilde{K}_{\mathrm{tail}}(1) \right| + \int_1^{\infty} \left| \frac{\mathrm{d}(K_{\mathrm{B}} - \widetilde{K}_{\mathrm{tail}})(\widetilde{t})}{\mathrm{d}\widetilde{t}} \right| \mathrm{d}\widetilde{t} \right\}. \tag{2.13}$$

When comparing (2.8) and (2.13) one can see that a good approximation $\widetilde{K}_{\mathrm{tail}}(\widetilde{t})$ of the tail reduces the error in (2.13) significantly in comparison with (2.8).

In order to find a good approximation $\widetilde{K}_{\mathrm{tail}}(\widetilde{t})$ we start with (2.10). The right hand side of (2.10) can be minimized and thereby minimizing the error in $\mathbf{F}_{\text{B-mod}}$. When determining $K_{\mathrm{tail}}(t)$ it is important that computation time is kept low. In order to achieve this, exponential functions are used because they can be implemented in a recursive way as explained later on. At first we start with one exponential function as follows,

$$K_{\mathrm{tail}}(t) = a \exp\left(-bt\right). \tag{2.14}$$

Here a and b are two positive constants. As a first guess we require that $K_{\mathrm{tail}}(t_{\mathrm{win}}) = K_{\mathrm{B}}(t_{\mathrm{win}})$ and $\frac{\mathrm{d}}{\mathrm{d}t} K_{\mathrm{tail}}(t_{\mathrm{win}}) = \frac{\mathrm{d}}{\mathrm{d}t} K_{\mathrm{B}}(t_{\mathrm{win}})$ in order to determine a and b. In this way $K_{\mathrm{mod}}(t)$, defined in (2.9), is continuously differentiable. Doing this results in

$$K_{\mathrm{tail}}(t) = \sqrt{\frac{e}{t_{\mathrm{win}}}} \exp\left(-\frac{t}{2t_{\mathrm{win}}}\right). \tag{2.15}$$

Fig. 2.1 shows several kernels, where the modified Basset kernel is given by (2.15). The error

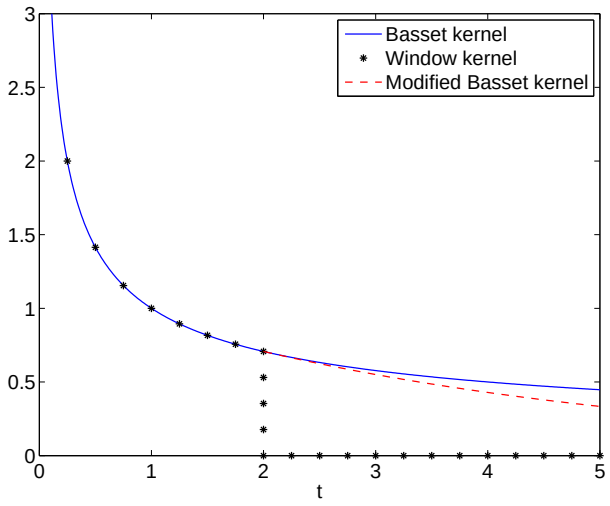

Figure 2.1: Basset kernel (solid line), window kernel (dots) and the modified Basset kernel (dashed line) for $t_{\mathrm{win}} = 2$.

by applying the modified Basset kernel is obviously smaller compared to the error for the window method. In order to minimize the error even more, multiple exponential functions can be used. Relation (2.15) provides an ansatz for the choice of a and b. Thus we write $K_{\mathrm{tail}}(t)$ as

$$K_{\mathrm{tail}}(t) = \sum_{i=1}^{m} a_i K_i(t), \qquad K_i(t) = \sqrt{\frac{e}{t_i}} \exp\left(-\frac{t}{2t_i}\right), \tag{2.16}$$

Table 2.1: Coefficients a_i and $\widetilde{t}_i$ in $\widetilde{K}_{\text{tail}}(\widetilde{t})$ with $m = 10$

$\widetilde{t}_i$	a_i
0.1	0.23477481312586
0.3	0.28549576238194
1	0.28479416718255
3	0.26149775537574
10	0.32056200511938
40	0.35354490689146
190	0.39635904496921
1000	0.42253908596514
6500	0.48317384225265
50000	0.63661146557001

with a_i and t_i positive constants. The functions $K_i(t)$ satisfy the following properties: $K_i(t_i) = K_{\text{B}}(t_i)$ and $\frac{\text{d}}{\text{d}t}K_i(t_i) = \frac{\text{d}}{\text{d}t}K_{\text{B}}(t_i)$. Combining (2.11) and (2.16), we obtain the following dimensionless representation for the tail:

$$\widetilde{K}_{\text{tail}}(\widetilde{t}) = \sum_{i=1}^{m} a_i \widetilde{K}_i(\widetilde{t}) \quad , \quad \widetilde{K}_i(\widetilde{t}) = \sqrt{\frac{e}{\widetilde{t}_i}} \exp\left(-\frac{\widetilde{t}}{2\widetilde{t}_i}\right) \quad , \quad \widetilde{t}_i = \frac{t_i}{t_{\text{win}}}. \tag{2.17}$$

The coefficients a_i and $\widetilde{t}_i$ should be chosen in such a way that the upper bound in (2.13) is minimized. However, Newton iteration will not work for this problem, and instead we consider the expression

$$\left(1 - \widetilde{K}_{\text{tail}}(1)\right)^2 + \int_1^{\infty} \widetilde{t}\left(\frac{\text{d}(K_{\text{B}} - \widetilde{K}_{\text{tail}})}{\text{d}\widetilde{t}}\right)^2 \text{d}\widetilde{t}, \ , \tag{2.18}$$

which provides a good indication for the optimal values of a_i and $\widetilde{t}_i$. In (2.18) an extra multiplication with $\widetilde{t}$ is introduced to correct for the change in norm. After minimizing the expression in (2.18), we can verify whether the error in (2.13) is of the same order. Since $\widetilde{K}_i(\widetilde{t}_i) = K_{\text{B}}(\widetilde{t}_i)$ and $\frac{\text{d}}{\text{d}\widetilde{t}}\widetilde{K}_i(\widetilde{t}_i) = \frac{\text{d}}{\text{d}t}K_{\text{B}}(\widetilde{t}_i)$, the function $\widetilde{K}_i(\widetilde{t})$ approximate $K_{\text{B}}(\widetilde{t})$ very well around $\widetilde{t}_i$. The kernel K_{B} must be approximated over a large range of $\widetilde{t}$-values and as a consequence $\widetilde{t}_i$ must also have a large range. Furthermore, K_{B} is changing slowly for large $\widetilde{t}$ so the small $\widetilde{t}_i$ must be close to each other whereas the large $\widetilde{t}_i$ can be far apart. The approach for finding a_i and $\widetilde{t}_i$ is thus the following. First, make a reasonable choice for $\widetilde{t}_i$, and second, calculate a_i by minimizing (2.18). Finally, determine the term between brackets from (2.13). Another slightly different set of $\widetilde{t}_i$-values can be chosen to see if a better approximation can be made. In Table 2.1 the result is shown for $m = 10$. Here one can see that some values of $\widetilde{t}_i$ are smaller than 1. This is surprising because the kernel K_{B} is not being approximated below $\widetilde{t} = 1$. When tuning the $\widetilde{t}_i$-values we found, however, that this improves the approximation.

From Fig. 2.2 it can be seen that $\widetilde{K}_{\text{tail}}$ approximates K_{B} relatively well over a wide range of $\widetilde{t}$. From Fig. 2.3 one can see that the error decays for large $\widetilde{t}$ (note the huge range of $\widetilde{t}$ in both figures).

Using (2.17) in combination with Table 2.1 for $\widetilde{K}_{\text{tail}}(\widetilde{t})$ the part between brackets in (2.13) can be calculated

$$\left|1 - \widetilde{K}_{\text{tail}}(1)\right| + \int_1^{\infty} \left|\frac{\text{d}(K_{\text{B}} - \widetilde{K}_{\text{tail}})(\widetilde{t})}{\text{d}\widetilde{t}}\right| \text{d}\widetilde{t} \approx 9.5 \cdot 10^{-3}. \tag{2.19}$$

Comparing this result with the window method (2.8) a factor of more than 200 is gained in accuracy. When keeping the same accuracy but changing the window, t_{win} can be decreased by a factor of $200^2 = 40000$.

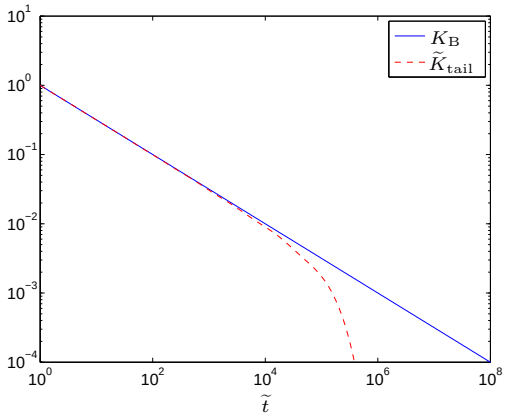

Figure 2.2: The kernels $K_{\mathrm{B}}(\widetilde{t})$ and $\widetilde{K}_{\mathrm{tail}}(\widetilde{t})$.

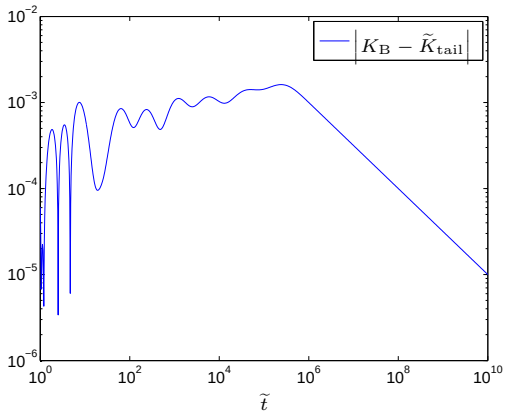

Figure 2.3: The error $\left|K_{\mathrm{B}} - \widetilde{K}_{\mathrm{tail}}\right|$.

2.4 Numerical approximation

In this section the numerical integration is discussed. First, the integration of the window and tail kernels are elaborated. Second, the overall numerical scheme for solving Eq. (2.1) and (2.2) is explained.

The integration of the Basset force with the modified kernel (2.9) and (2.16) is split into two parts, the window kernel and the tail of the kernel as follows,

$$\begin{aligned}
\mathbf{F}_{\text{B-mod}}(t) &= c_{\mathrm{B}} \int_{-\infty}^{t} K_{\mathrm{mod}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau \\
&= c_{\mathrm{B}} \int_{t-t_{\mathrm{win}}}^{t} K_{\mathrm{B}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau + c_{\mathrm{B}} \int_{-\infty}^{t-t_{\mathrm{win}}} K_{\mathrm{tail}}(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau \\
&= \mathbf{F}_{\text{B-win}}(t) + \mathbf{F}_{\text{B-tail}}(t).
\end{aligned} \tag{2.20}$$

In the following, methods are described for the calculation of $\mathbf{F}_{\text{B-win}}$ and $\mathbf{F}_{\text{B-tail}}$.

First, we consider the Basset force due to the window kernel $\mathbf{F}_{\text{B-win}}$. The kernel of the Basset force is singular for $t \to 0$ which impedes use of the ordinary trapezoidal rule. In order to deal with the singularity we introduce an alternative, trapezoidal-based method, referred to as the TB-method. The idea is as follows. The trapezoidal rule is based on linear interpolation of the integrand on each subinterval. In our approach $\mathbf{g}(t)$ is approximated by its linear interpolant $\mathbf{P}_1(t)$, and subsequently the integration of $K_{\mathrm{B}}(t-\tau)\mathbf{P}_1(\tau)$ is done exactly. For the numerical implementation we start with the discretization of the interval $[t-t_{\mathrm{win}}, t]$, given by $\tau_n = t - n\Delta t, \quad n = 0, 1, 2, \cdots, N$ with $\Delta t = t_{\mathrm{win}}/N$. Now the integral can be split as

$$\mathbf{F}_{\text{B-win}}(t) = c_{\mathrm{B}} \sum_{n=1}^{N} \int_{\tau_n}^{\tau_{n-1}} \frac{\mathbf{g}(\tau)}{\sqrt{t-\tau}} \mathrm{d}\tau \ . \tag{2.21}$$

The next step is to approximate $\mathbf{g}(\tau)$ by its linear interpolant on each subinterval, which yields

$$\mathbf{F}_{\text{B-win}}(t) \approx c_{\mathrm{B}} \sum_{n=1}^{N} \int_{\tau_n}^{\tau_{n-1}} \frac{\mathbf{g}_n + (\mathbf{g}_{n-1} - \mathbf{g}_n)(\tau - \tau_n)/\Delta t}{\sqrt{t-\tau}} \mathrm{d}\tau, \tag{2.22}$$

where $\mathbf{g}_n \equiv \mathbf{g}(\tau_n)$. After the change of variable $\tau' = t - \tau$ this integral can be evaluated and the

following result[1] is obtained:

$$\mathbf{F}_{\text{B-win}}(t) \approx \frac{4}{3}c_{\text{B}}\mathbf{g}_0\sqrt{\Delta t} + c_{\text{B}}\mathbf{g}_N\frac{\sqrt{\Delta t}\left(N-\frac{4}{3}\right)}{(N-1)\sqrt{N-1}+(N-\frac{3}{2})\sqrt{N}}$$
$$+c_{\text{B}}\sqrt{\Delta t}\sum_{n=1}^{N-1}\mathbf{g}_n\left(\frac{n+\frac{4}{3}}{(n+1)\sqrt{n+1}+(n+\frac{3}{2})\sqrt{n}}+\frac{n-\frac{4}{3}}{(n-1)\sqrt{n-1}+(n-\frac{3}{2})\sqrt{n}}\right). \quad (2.23)$$

From the result above one can see that three inner products must be calculated each time step, one inner product for each spatial dimension. One vector contains all the values $\mathbf{g}_n$ which must be shifted by one index each time step. The other vector containing the coefficients in (2.23) is calculated once at the start of the computation. In this way the computational time is kept minimal. The part with $\mathbf{g}_0$ will be treated in a different way as explained later on in order to improve stability.

Next, the numerical integration of the tail of the Basset force is discussed. The idea is to find a recursive formulation in order to minimize computation efforts. Using expression (2.16) for K_{tail}, $\mathbf{F}_{\text{B-tail}}$ becomes:

$$\mathbf{F}_{\text{B-tail}}(t) = \sum_{i=1}^{m} a_i c_{\text{B}} \int_{-\infty}^{t-t_{\text{win}}} K_i\left(t-\tau\right)\mathbf{g}(\tau)\mathrm{d}\tau = \sum_{i=1}^{m} a_i \mathbf{F}_i(t)\ , \quad (2.24)$$

Here, $\mathbf{F}_i$ represents the contribution of the i-th exponential function. Now $\mathbf{F}_i$ is split into two parts, as follows.

$$\mathbf{F}_i(t) = c_{\text{B}}\int_{t-t_{\text{win}}-\Delta t}^{t-t_{\text{win}}} K_i(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau + c_{\text{B}}\int_{-\infty}^{t-t_{\text{win}}-\Delta t} K_i(t-\tau)\mathbf{g}(\tau)\mathrm{d}\tau$$
$$= \mathbf{F}_{i\text{-di}}(t) + \mathbf{F}_{i\text{-re}}(t)\ , \quad (2.25)$$

where we have to compute $\mathbf{F}_{i\text{-di}}$ directly and where $\mathbf{F}_{i\text{-re}}$ can be computed recursively. For $\mathbf{F}_{\text{i-di}}$ the same procedure is followed as with the window kernel. Using this procedure the following result[2] can be obtained:

$$\mathbf{F}_{i\text{-di}}(t) \approx c_{\text{B}}\sqrt{\frac{e}{t_i}}\int_{t_{\text{win}}}^{t_{\text{win}}+\Delta t}\exp\left(-\frac{\tau'}{2t_i}\right)\left(\mathbf{g}_N+\frac{t_{\text{win}}-\tau'}{\Delta t}(\mathbf{g}_N-\mathbf{g}_{N+1})\right)\mathrm{d}\tau' = 2c_{\text{B}}\sqrt{et_i}$$
$$\exp\left(-\frac{t_{\text{win}}}{2t_i}\right)\left\{\mathbf{g}_N\left[1-\varphi\left(-\frac{\Delta t}{2t_i}\right)\right]+\mathbf{g}_{N+1}\exp\left(-\frac{\Delta t}{2t_i}\right)\left[\varphi\left(\frac{\Delta t}{2t_i}\right)-1\right]\right\}, \quad (2.26)$$

where $\varphi(z) = (e^z-1)/z = 1+\frac{1}{2}z+\frac{1}{6}z^2+\mathcal{O}\left(z^3\right)$. Finally, $\mathbf{F}_{\text{i-re}}$ can be easily calculated using the value of $\mathbf{F}_i$ at the previous time step:

$$\begin{aligned}\mathbf{F}_{i\text{-re}}(t) &= c_{\text{B}}\int_{-\infty}^{t-t_{\text{win}}-\Delta t}\sqrt{\frac{e}{t_i}}\exp\left(-\frac{t-\tau}{2t_i}\right)\mathbf{g}(\tau)\mathrm{d}\tau \\ &= \exp\left(-\frac{\Delta t}{2t_i}\right)c_{\text{B}}\int_{-\infty}^{t-t_{\text{win}}-\Delta t}\sqrt{\frac{e}{t_i}}\exp\left(-\frac{t-\Delta t-\tau}{2t_i}\right)\mathbf{g}(\tau)\mathrm{d}\tau \\ &= \exp\left(-\frac{\Delta t}{2t_i}\right)\mathbf{F}_{\text{i}}(t-\Delta t)\ . \end{aligned} \quad (2.27)$$

In this last part the overall numerical scheme is discussed. To solve equation (2.1) and (2.2) numerically the second-order Adams-Bashforth (AB2) method is implemented. For a differential equation $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{h}(t,\mathbf{y})$ the scheme reads $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}\left(3\mathbf{h}^n - \mathbf{h}^{n-1}\right)$, where $\mathbf{h}^n = \mathbf{h}(t^n,\mathbf{y}^n)$.

[1] This formulation is preferred to avoid loss of significant digits in the computation of $\mathbf{F}_{\text{B-win}}$.

[2] Note that in equation (2.26) Taylor series must be used for $\varphi\left(-\frac{\Delta t}{2t_i}\right)$ when $\Delta t \ll t_i$.

Equation (2.1) can be directly integrated with this scheme but for equation (2.2) some modifications are needed. In order to have a stable scheme, the $\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{dt}}$ term in the added mass force is treated in an implicit way instead of explicit. Moreover, it turned out that the AB2-method has poor stability properties for the calculation of the Basset force using the window method. Extremely small time steps must be taken in order to have a stable solution. An alternative method circumventing stability problems is to bring a part of the Basset force (the contribution $\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{dt}}$ evaluated at t) to the left hand side. Eq. (2.2) is then reformulated as

$$\left(m_p + \frac{1}{2}m_f + \frac{4}{3}c_\mathrm{B}\sqrt{\Delta t}\right)\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{dt}} = \mathbf{F}_\mathrm{St} + \mathbf{F}_\mathrm{P} + \mathbf{F}_\mathrm{G} + \mathbf{F}'_\mathrm{AM} + \mathbf{F}'_\mathrm{B}\ , \tag{2.28}$$

with $\mathbf{F}'_\mathrm{AM} = \frac{1}{2}m_f\left(\frac{\mathrm{D}\mathbf{u}}{\mathrm{Dt}} + \frac{1}{10}a^2\frac{\mathrm{d}}{\mathrm{dt}}(\nabla^2\mathbf{u})\right)$ and $\mathbf{F}'_\mathrm{B} = \mathbf{F}_\mathrm{B} - \frac{4}{3}c_\mathrm{B}\sqrt{\Delta t}\frac{\mathrm{d}\mathbf{u}_p}{\mathrm{dt}}$. In this way the Basset force becomes partially implicit instead of completely explicit. Finally, as only the time derivative along the particle path $\frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}}$ is available, the time derivative along the path of a fluid element $\frac{\mathrm{D}\mathbf{u}}{\mathrm{Dt}}$ is computed according to

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{Dt}} = \frac{\partial\mathbf{u}}{\partial t} + u_j\frac{\partial\mathbf{u}}{\partial x_j} = \frac{\partial\mathbf{u}}{\partial t} + u_{p,j}\frac{\partial\mathbf{u}}{\partial x_j} + (u_j - u_{p,j})\frac{\partial\mathbf{u}}{\partial x_j} = \frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}} + (u_j - u_{p,j})\frac{\partial\mathbf{u}}{\partial x_j}\ . \tag{2.29}$$

2.5 Validation of the Basset force integration

In this section four test cases are presented in order to validate the methods for the integration of the Basset force. The first example tests the trapezoidal-based (TB) method and compares the results with the semi-derivative (SD) approach by Bombardelli *et al.* [15]. Example 2 and 3 test the the overall numerical scheme. Here both stability and convergence are tested for the explicit and the partially implicit TB-method. Finally, example 4 shows the efficiency of the Basset force using the tail kernel.

Example 1: Basset integral for a given convolution function
In order to demonstrate the advantages of the TB-method, the convergence of this method is compared with the SD-approach of Bombardelli *et al.* [15]. To that end the arbitrary test function $g(\tau) = \cos\tau$ is used. The exact Basset integral is given by

$$\begin{aligned} F_\mathrm{B}(t) &= c_\mathrm{B}\int_0^t \frac{\cos\tau}{\sqrt{t-\tau}}\mathrm{d}\tau = 2c_\mathrm{B}\int_0^{\sqrt{t}}\cos(t-\sigma^2)\mathrm{d}\sigma \\ &= c_\mathrm{B}\sqrt{2\pi}\left(C(\sqrt{2t/\pi})\cos t + S(\sqrt{2t/\pi})\sin t\right), \end{aligned} \tag{2.30}$$

with $\sigma = \sqrt{t-\tau}$ and $C(t)$ and $S(t)$ the Fresnel cosine and sine functions [28], respectively.

The Basset integral F_B was evaluated at $t = 50\pi$ with different numbers of points N uniformly distributed in the interval $[0, 50\pi]$. The results for both the SD-approach and the TB-method are presented in Table 2.2. Here, it can be seen that the error of the TB-method is substantially smaller than that of the SD-approach. When increasing the number of points N it can be seen that the TB-method is second-order accurate in time (in agreement with analysis that can be done by using Taylor series), whereas the SD-approach is first-order accurate in time. More methods have been compared by Bombardelli *et al.* [15] but these methods have even lower order of convergence than the SD-approach.

Example 2: Space-dependent steady velocity field
In order to test the overall numerical scheme for the computation of particle trajectories we have implemented a particular space-dependent steady velocity field. The particle trajectory is a circle and given by $(x(t), y(t)) = (r\cos\omega t, -r\sin\omega t)$, where r and ω denote the radius and the angular velocity, respectively. The velocity field and its derivation is given in appendix B. For the test case, exactly one revolution is simulated, from $t = 0$ until $t = 2\pi$. In order to test the stability of the

Table 2.2: Relative error and order of convergence for the Basset integral, for the SD-approach [15] and the TB-method.

points N	Relative error SD	Order SD	Relative error TB	Order TB
81	$4.03 \cdot 10^{-1}$		$1.34 \cdot 10^{-1}$	
243	$1.37 \cdot 10^{-1}$	1.0	$2.54 \cdot 10^{-2}$	1.5
729	$4.66 \cdot 10^{-2}$	1.0	$3.29 \cdot 10^{-3}$	1.9
2,187	$1.56 \cdot 10^{-2}$	1.0	$3.93 \cdot 10^{-4}$	1.9
6,561	$5.22 \cdot 10^{-3}$	1.0	$4.54 \cdot 10^{-5}$	2.0
19,683	$1.74 \cdot 10^{-3}$	1.0	$5.15 \cdot 10^{-6}$	2.0
59,049	$5.80 \cdot 10^{-4}$	1.0	$5.80 \cdot 10^{-7}$	2.0
177,147	$1.93 \cdot 10^{-4}$	1.0	$6.49 \cdot 10^{-8}$	2.0
531,441	$6.45 \cdot 10^{-5}$	1.0	$7.24 \cdot 10^{-9}$	2.0
1,594,323	$2.15 \cdot 10^{-5}$	1.0	$8.06 \cdot 10^{-10}$	2.0

overall scheme two different approaches have been tested. One with the completely explicit time integration procedure for the Basset force and the other with the partially implicit procedure, see Section 2.4. For both the implicit and explicit method the Basset force is computed with the TB-method and show second-order convergence in Δt. The relative error is computed with $\mathbf{x}_p(2\pi)$. The results are presented in Table 2.3 and clearly indicate that the explicit scheme is very unstable when the number of time steps is smaller than 256. Even when taking 256 time steps the explicit procedure may be unstable and the data in Table 2.3 are put in parenthesis to indicate this uncertainty. The partially implicit scheme remains stable even with the number of time steps as small as 16.

Table 2.3: Relative error and order of convergence for the overall numerical scheme, tested for the trajectory of a small particle in a space dependent steady velocity field.

number of time steps	Relative error explicit	Order explicit	Relative error implicit	Order implicit
16	unstable		$3.63 \cdot 10^{-1}$	
32	unstable		$8.32 \cdot 10^{-2}$	2.1
64	unstable		$2.09 \cdot 10^{-2}$	2.0
128	unstable		$5.28 \cdot 10^{-3}$	2.0
256	$\left(4.80 \cdot 10^{-2}\right)$		$1.33 \cdot 10^{-3}$	2.0
512	$3.05 \cdot 10^{-4}$		$3.33 \cdot 10^{-4}$	2.0
1024	$7.68 \cdot 10^{-5}$	2.0	$8.34 \cdot 10^{-5}$	2.0
2048	$1.93 \cdot 10^{-5}$	2.0	$2.09 \cdot 10^{-5}$	2.0
4096	$4.84 \cdot 10^{-6}$	2.0	$5.21 \cdot 10^{-6}$	2.0

Example 3: Time-dependent velocity field
The trajectory of a spherical particle in an arbitrary time-dependent velocity field can rather straightforwardly be computed as long as the velocity field is smooth enough. The derivation of the particle trajectory uses Laplace transforms and the analytical procedure is given in appendix C. The overall numerical scheme is tested by computing the trajectory of a particle in the following one-dimensional, time-dependent velocity field

$$u(t) = \frac{(m_p - m_f)g}{6\pi a\mu} \cos 2t \ . \tag{2.31}$$

The total force on the particle is zero at $t = 0$, i.e, $\mathbf{F}_{\mathrm{St}}$ and $\mathbf{F}_{\mathrm{G}}$ are in balance. In order to compute the Basset force the implicit TB-method is used. The integration is carried out from $t = 0$ until

$t = 2\pi$. The relative error is computed for $u_p(2\pi)$ and is presented in Table 2.4, where once again second-order time accuracy is confirmed. From these test cases, using both a time-dependent and a space-dependent velocity field for the computation of particle trajectories, we can conclude that the (partially implicit) TB-method is stable and second-order accurate in time, and conjecture that this remains the case for particles in arbitrary time- and space-varying flow fields.

Table 2.4: Relative error and order of convergence for the overall numerical scheme, for the velocity field (2.31).

time steps	Relative error	Order
16	$9.96 \cdot 10^{-2}$	
32	$2.38 \cdot 10^{-2}$	2.1
64	$5.57 \cdot 10^{-3}$	2.1
128	$1.31 \cdot 10^{-3}$	2.1
256	$3.13 \cdot 10^{-4}$	2.1
512	$7.56 \cdot 10^{-5}$	2.0
1024	$1.84 \cdot 10^{-5}$	2.0
2048	$4.53 \cdot 10^{-6}$	2.0
4096	$1.12 \cdot 10^{-6}$	2.0

Example 4: Computational efficiency due to modified kernel integration
In this example the computational savings when using the modified tail kernel, given in (2.9) and (2.16), is investigated based on analysis of the number of flops per time step, per particle and per space dimension. For the window kernel this is $N + 1$ flops because only one vector dot product is calculated. For each exponential function three extra flops are needed. To see how efficient the tail kernel works the upper bound (2.13) for the error is plotted as a function of the computation time, Fig 2.4. Different numbers (indicated by m) of exponential functions are taken into account. The results are plotted in Fig 2.4. Here it can be seen that the best choice for m depends on the particular situation. Because the computation time is directly connected with N as mentioned above, N must be chosen optimally. From Fig 2.4 one can see that N must be chosen small, otherwise increasing m would be a better option. Because $t_{\text{win}} = N\Delta t$ this immediately gives an optimal value for t_{win}. Furthermore, the results show a significant saving in computation time. This can easily be a factor of 100 or more. When looking to the memory requirements the results are even better. For the window method as many memory locations as flops are needed whereas each exponential function only takes one memory location instead of 3 flops. So using the tail kernel not only saves time but also memory.

Typically, the use of the tail kernel reduces the computational costs of the Basset force by more than an order of magnitude, whereas the memory requirement is even reduced more. Furthermore, the error is reduced by more than an order of magnitude. The question remains, of course, whether the computational savings directly result in faster simulations. This depends on the remaining part of the simulation. Although the other force contributions in (2.28) can be calculated much faster than the Basset force this does not have to hold for the interpolation of the velocities in a turbulence simulation. The velocity of the flow field is only computed at the grid points and an interpolation must be carried out to compute the velocity at the particle position. This may be very time consuming and it can become the new bottleneck. The reduction of CPU-time might then not be as big as expected but it remains significant. Additionally, the decrease in memory requirement may become essential when increasing the number of particles in turbulence simulations.

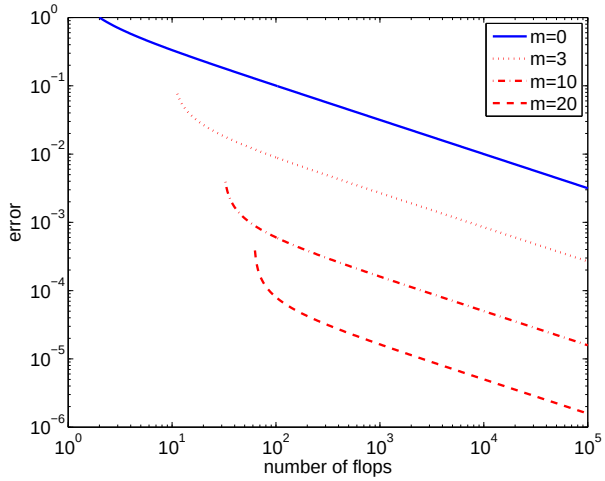

Figure 2.4: Upper bound for the error in the approximation of the Basset force as a function of the number of flops for different number of exponential functions (indicated by m).

2.6 Light particles in isotropic turbulence

In this section a brief statistical analysis of velocities of particles, released in an isotropic turbulent flow, is provided. The isotropic turbulence simulation is performed by means of direct numerical simulations. The numerical code consist of two parts. First the Navier-Stokes equations with the Boussinesq approximation are solved on a triple periodic domain using a pseudo-spectral code [2, 3] (Eulerian approach). Second, the particle trajectories are obtained by the Lagrangian approach as explained in the previous sections. The simulation is performed on a 128^3 grid. The number of (light) particles is 20,000 and the particle-to-fluid density ratio $\rho_p/\rho_f = 4$ (thus all terms in the MR equation are relevant, see Refs. [8, 9]). The integral-scale Reynolds number is $\mathrm{Re} = UL/\nu = 1333$, with U the typical root-mean-square velocity and L the integral length scale. The Stokes number St is typically in the range $0.1 \leq \mathrm{St} \leq 1.0$ [9] and particles are tracked for a period of approximately two eddy turnover times.

Two simulations have been carried out under exactly the same flow conditions and particle tracking is either based on the classical approach (window method) or on the novel integration method (exponential method) for the Basset kernel. In the first simulation only the window kernel (2.4) has been used, where the number of time steps in the window is $n = 500$. The other one uses the modified window kernel, given in (2.9) and (2.16). In this case only five time steps are taken into account in the window, so $n = 5$. For the tail of the Basset kernel the number of exponential functions $m = 10$.

In order to study a particle trajectory we start with considering the energy spectrum of the particle. To obtain the energy spectrum, we first need to calculate the autocorrelation $R(\tau)$ of the velocity, which is defined by

$$R(\tau) = \frac{\langle u_p(t)u_p(t+\tau)\rangle}{\langle u_p(t)^2\rangle} \ . \tag{2.32}$$

Here, $\langle\cdot\rangle$ denotes the average in over the different particles. The particles are embedded in a homogeneous isotropic turbulent flow and no gravitation is applied. Therefore, we are allowed to average over the components of the velocity vector of all particles. No time averaging has been applied for the present velocity data as this run covers only one or two eddy turnover times. The results for the autocorrelation of the velocity are shown in Fig.2.5 and we see that the results for both the window method and the exponential method are comparable. The energy spectrum obtained from the particle velocities can be calculated by taking the cosine transform of the autocorrelation function, and is shown in Fig.2.6. Although the results are similar we are

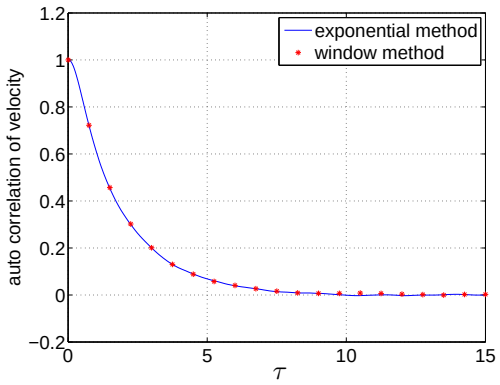

Figure 2.5: Autocorrelation of the particle velocity $\mathbf{u}_p$. The solid line represents the result from the exponential method and the dots those from the window method.

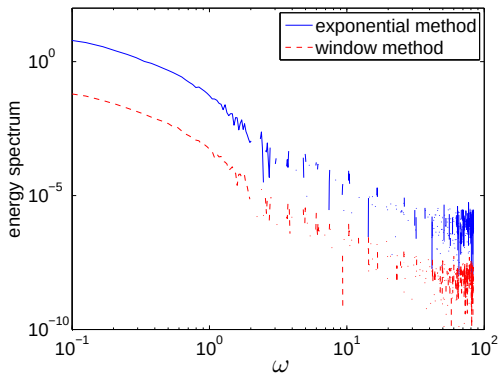

Figure 2.6: Energy spectrum of the particle velocity. The graph of the window method (dashed line) is shifted downward with respect to the spectrum from the exponential method (solid line) by a factor of 100 for clarity.

interested in possible differences between the two spectra. If these differences have an overall trend this would mean that statistical properties can be influenced by the different methods of evaluating the Basset kernel. However, to observe any error in the evaluation of the Basset force kernel the differences should be larger than the statistical noise.

A starting point for an analytical evaluation of possible differences between the window method and the exponential method consists of the response of a single particle in a uniform oscillating flow field. We are therefore interested in the periodic solution u_p of a spherical particle responding to an oscillating velocity field $u = \cos\omega t$ (or $u = \mathcal{R}[\exp(i\omega t)]$, with i the imaginary unit and $\mathcal{R}$ denoting the real part of this expression). The particle velocity can then be expressed as $u_p = \mathcal{R}[V\exp(i\omega t)]$ with V a complex amplitude, which is dependent on the method chosen to evaluate the Basset force kernel. For the window method and the exponential method we introduce V_{win} and V_{exp}, respectively. For V_{exp} the exact solution V_{ex} is used since the error of the exponential method is assumed to be negligibly small, see also Fig. 2.3. In general, $|V_{\text{win}}| \neq |V_{\text{ex}}|$ which means that some frequencies are suppressed with the window method while others may be amplified. This should become visible in the energy spectrum of particle velocities.

In order to find V_{win} Eq. (2.2) should be solved for $u = \mathcal{R}[\exp(i\omega t)]$ and $u_p = \mathcal{R}[V\exp(i\omega t)]$, resulting in the following integro-differential equation:

$$i\omega m_p V_{\text{win}} = 6\pi a\mu\,(1 - V_{\text{win}}) + \frac{i\omega}{2} m_f\,(3 - V_{\text{win}}) + i\omega c_{\text{B}}(1 - V_{\text{win}}) \int_{t-t_{\text{win}}}^{t} \frac{e^{-i\omega(t-\tau)}}{\sqrt{t-\tau}}\mathrm{d}\tau\ . \quad (2.33)$$

Here, we used the fact that the velocity field is uniform, one dimensional and that no gravity is applied. Applying the change in variables $\sigma = \sqrt{(t-\tau)\omega}$, allows us to find an expression for V_{win} i.e.,

$$V_{\text{win}} \;=\; 1 + \frac{(m_f - m_p)i\omega}{6\pi a\mu + \left(\frac{1}{2}m_f + m_p\right) i\omega + c_{\text{B}}\sqrt{2\omega\pi}\,Q(\sqrt{2t_{\text{win}}\omega\pi})}, \quad (2.34)$$

where $Q(t) = S(t) + iC(t)$, with $C(t)$ and $S(t)$ the Fresnel cosine and sine functions, respectively [28]. V_{ex} can now be found by taking $V_{\text{ex}} = \lim_{t_{\text{win}}\to\infty} V_{\text{win}}$ which results in

$$V_{\text{ex}} \;=\; 1 + \frac{(m_f - m_p)i\omega}{6\pi a\mu + \left(\frac{1}{2}m_f + m_p\right) i\omega + c_{\text{B}}\sqrt{\frac{\omega\pi}{2}}(1+i)}. \quad (2.35)$$

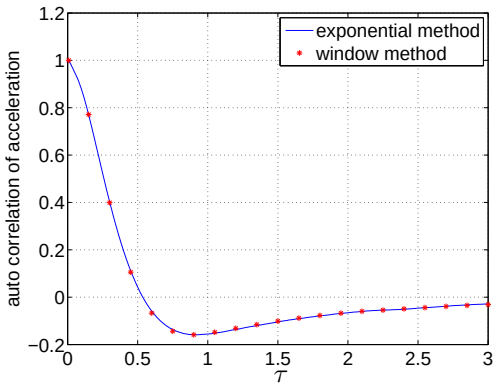

Figure 2.7: Autocorrelation of the particle acceleration $\mathbf{a}_p = d\mathbf{u}_p/dt$. The solid line represents the result from the exponential method and the dots those from the window method.

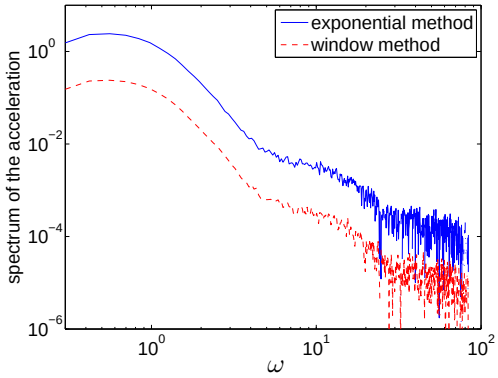

Figure 2.8: Spectrum of the particle acceleration. The graph of the window method (dashed line) is shifted downward with respect to the spectrum from the exponential method (solid line) by a factor of 10 for clarity.

Inspection of the energy spectrum displayed in Fig. 2.6 reveals that the noise becomes more important for increasing ω. The effects from the different methods for the computation of the Basset force kernel turned out to be most important for $\omega \geq 1$. Unfortunately, the noise in the energy spectrum is already larger than predicted for the differences between the window and exponential method. One way of decreasing the error would be averaging over time, but with the limited number of eddy turnover times in the present simulation this is not feasible. However, an alternative approach exists in comparing the autocorrelation of the particle acceleration.

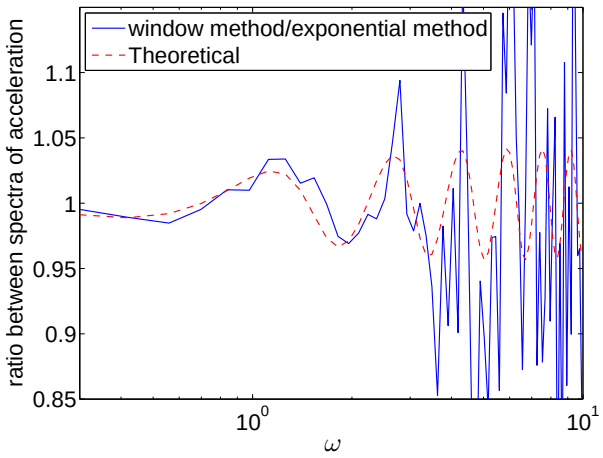

Figure 2.9: The theoretical ratio $\left(\frac{|V_{\mathrm{win}}|}{|V_{\mathrm{ex}}|}\right)^2$ (dashed line) compared with a similar ratio of the particle acceleration spectra (solid line).

The autocorrelation of the particle acceleration is plotted in Fig. 2.7. Here, the typical time scale is much shorter than that of the particle velocity, therefore it is possible to also average over time. The spectrum is calculated by taking the cosine transform of the autocorrelation acceleration function and is displayed in Fig. 2.8. Because the particle acceleration is used instead of the particle velocity, higher frequencies (shorter time scales) become more important. In this way deteriorating influence of the noise on the spectrum is shifted to higher frequencies. Nevertheless, the computed spectrum should still be affected by the method that is chosen for the evaluation

of the Basset force kernel. In order to observe the differences the best approach is to plot the ratio of the two spectra as function of frequency. When no essential difference exists between the window and exponential method their ratio would be equal to one with some noise added to it. However, the window method suppresses some frequency components while others are amplified, so the deviation from one is a measure for the error in the window method. In Fig. 2.9 the ratio of both spectra is shown in combination with the theoretical ratio defined by $\left(\frac{|V_{\mathrm{win}}|}{|V_{\mathrm{ex}}|}\right)^2$. From Fig. 2.9 it can be seen that the theoretical ratio predicts the ratio obtained from the simulation, including the local maxima and minima quite well, provided the frequency is not too high. For higher frequencies the noise becomes larger but the theoretical and computational ratios still seem to have the same trend. The novel exponential method to evaluate the Basset force kernel might be considered as an excellent and efficient method for tracking of many particles in turbulent flows.

2.7 Conclusions

We have introduced a novel method for the evaluation of the Basset force kernel and analysed several aspects of its implementation. The tail of the Basset force kernel is approximated by exponential functions. The contribution of these exponential functions can be calculated in a recursive way which makes it very efficient. Typically the use of the tail kernel reduces the computational costs of the Basset force by more than an order of magnitude, whereas the memory requirement is reduced even more. Furthermore, the error in the tail of the Basset force is also reduced by more than an order of magnitude in comparison with the traditional window method.

A trapezoidal-based method is developed in order to deal with the singularity of the Basset force. This method has a temporal accuracy of $\mathcal{O}\left(\Delta t^2\right)$ where other methods only have a temporal accuracy of $\mathcal{O}\left(\Delta t\right)$ or lower. This method is made partially implicit in order to make it more stable.

The method has been implemented in a tracking algorithm for (light) inertial particles in turbulent flows. The isotropic turbulence simulation shows that the error made by the window method can influence statistics on the particle trajectories. This has been illustrated with the velocity and acceleration spectra. Therefore, the novel exponential method is preferred over the classical window method. Because the new implementation is much faster than the classical one, more particles can be taken into account in simulations, which opens possibilities for further research.

Chapter 3

Interpolation schemes in DNS simulations of turbulence: error estimates and implementation

3.1 Introduction

Maxey and Riley [4] introduced the equations of motion for small ($d_p \ll \eta$, with d_p the particle diameter and η the Kolmogorov length scale) isolated rigid spherical particles in a non-uniform velocity field $\mathbf{u}(\mathbf{x}, t)$. An important assumption is that the particle Reynolds number $Re_p = d_p|\mathbf{u} - \mathbf{u}_p|/\nu \ll 1$, with $\mathbf{u}_p$ the velocity of the particle and ν the kinematic viscosity of the fluid. As we consider small particle diameters and small volume fractions of particles we ignore the effects of two-way and four-way coupling. An elaborate overview of the different terms in the Maxey-Riley equations and their numerical implementation can be found in the paper by Loth [22] and a historical account of the equations of motion was given in a review article by Michaelides [23]. Time integration of these equations to compute particle trajectories is an expensive, time- and memory consuming job. First, the computation of the Basset history force can be computationally very expensive. However, a significant reduction can be obtained by fitting the diffusive kernel of the Basset history force with exponential functions, as recently shown by Hinsberg *et al.* [5]. Second, the interpolation step can be very time consuming and memory demanding as well. Where some studies use low order linear interpolation others use high order spline interpolations [18, 19]. These high order spline interpolations have the disadvantage of being more computational expensive. Therefore, it is important that they give a significantly more accurate result.

The turbulent flow is approximated by means of direct numerical simulations. Our numerical code consist of two parts. First, the Navier-Stokes equations with the Boussinesq approximation are solved on a triple periodic domain using a pseudo-spectral code [2, 3] (Eulerian approach). We consider an incompressible flow, which results in a divergence free velocity field. Second, the particle trajectories are obtained by a Lagrangian approach using the Maxey-Riley equations. For this last step the fluid velocity and its first derivatives must be known at the centers of the particles. Because we use a pseudo-spectral code these can be calculated everywhere. Unfortunately this would be far too expensive and in practice only small amounts of particles can be tracked $\mathcal{O}(10)$. Therefore, the velocity is usually represented on a finite rectangular grid and an interpolation should be carried out. In order to choose the most appropriate interpolation method for a particular case one needs to compare errors: the interpolation error is compared with the discretisation error of the flow field. In this way one can prevent unnecessary computations.

In this study we focus on comparing errors in order to make a well-founded choice for the interpolation method. First, Sections 3.2 and 3.3 give practical methods on how to estimate the interpolation error. Section 3.2 focuses on calculating the interpolation error off-line. In this

way only statistics are needed of the turbulent flow without having to compute interpolations. In Section 3.3 an even more practical method is proposed. This method only needs the energy spectrum to predict the interpolation error. Further, different interpolation methods are discussed in Section 3.4 and a general framework is proposed for them. Section 3.5 discusses some algorithms on how to calculate the interpolation efficiently. Thereafter, the results are shown in Section 3.6. Finally, concluding remarks are given in Section 3.7.

3.2 Interpolation error

In order to avoid unnecessary computations it is important that the interpolation error has the same order of magnitude as the discretisation error of the flow field. This section focuses on the estimation of the interpolation error. Furthermore, it is important to estimate the magnitude of the error in order to say something about the accuracy of the generated data. In this section we focuss on how to calculate the interpolation error off-line. In this way only statistics are needed of the turbulent flow without having to compute interpolations, which results in an efficient method.

In general the center of a particle will follow a path $\mathbf{x}_p(t)$ and the flow velocity field is given by $\mathbf{u}(\mathbf{x},t)$. Say that we need to find the velocity at the center of the particle, i.e. $\mathbf{u}(\mathbf{x}_p(t),t)$, but instead we find the approximation $\widetilde{\mathbf{u}}(\mathbf{x}_p(t),t)$, which is due to the interpolation. Let Φ be the interpolation operator that maps $\mathbf{u}$ onto $\widetilde{\mathbf{u}}$, so $\widetilde{\mathbf{u}} = \Phi[\mathbf{u}]$. The relative interpolation error ϵ can be computed by the L^2-norm like

$$\epsilon = \lim_{T\to\infty} \sqrt{\frac{\int_0^T |\mathbf{u}-\widetilde{\mathbf{u}}|^2(\mathbf{x}_p(t),t)\mathrm{d}t}{\int_0^T |\mathbf{u}|^2(\mathbf{x}_p(t),t)\mathrm{d}t}}, \tag{3.1}$$

where $|\cdot|$ denotes the usual 2-norm. In principle this relation could be used to calculate the interpolation error because $\mathbf{u}(\mathbf{x}_p(t),t)$ can be calculated from its Fourier components. Unfortunately this would be far too expensive and in practice only small amounts of particles can be tracked $\mathcal{O}(10)$. Instead Equation (3.1) will be used to validate the following steps.

We assume that the system is ergodic which means that the ensemble average is equal to the time average. Therefore ϵ does not depend on the choice of the particle and an average over particles can be taken. Furthermore, we assume that the particle has no preferential location in order to replace the particle average by a space average. Now we can average over space and time, where $\langle\cdot\rangle_T$ denotes the average over time, i.e,

$$\langle f(t)\rangle_T = \frac{1}{T}\int_0^T f(t)\mathrm{d}t. \tag{3.2}$$

In practice the time average should be taken over several large eddy turnover times. The space average is taken over the whole domain V which is $(0,1)^3$ in dimensionless units. In this way one finds

$$\epsilon^2 = \frac{\left\langle \iiint_V |\mathbf{u}-\widetilde{\mathbf{u}}|^2(\mathbf{x},t)\mathrm{d}\mathbf{x}\right\rangle_T}{\left\langle \iiint_V |\mathbf{u}|^2(\mathbf{x},t)\mathrm{d}\mathbf{x}\right\rangle_T}. \tag{3.3}$$

We introduce the following inner products and corresponding norms:

$$\begin{aligned} \langle \mathbf{f},\mathbf{g}\rangle_1 &= \int_0^1 \mathbf{f}\cdot\mathbf{g}^*(x)\mathrm{d}x, \qquad \|\mathbf{f}\|_1^2 = \langle \mathbf{f},\mathbf{f}\,\rangle_1 = \int_0^1 |\mathbf{f}(x)|^2\mathrm{d}x\ , \\ \langle \mathbf{f},\mathbf{g}\rangle_3 &= \int_0^1\int_0^1\int_0^1 \mathbf{f}\cdot\mathbf{g}^*(\mathbf{x})\mathrm{d}x\mathrm{d}y\mathrm{d}z, \qquad \|\mathbf{f}\|_3^2 = \langle \mathbf{f},\mathbf{f}\,\rangle_3 = \int_0^1\int_0^1\int_0^1 |\mathbf{f}(\mathbf{x})|^2\mathrm{d}x\mathrm{d}y\mathrm{d}z. \end{aligned} \tag{3.4}$$

Here $\mathbf{f}\cdot\mathbf{g}$ denotes the usual inner product and $\mathbf{g}^*$ denotes the complex conjugate of $\mathbf{g}$. These inner products and norms are also used for scaler fields like f and g where they change in a reduce to

the ordinary product $fg*$. The velocity field is expanded in a three-dimensional Fourier series, so

$$\begin{aligned} \mathbf{u}(\mathbf{x},t) &= \sum_{\mathbf{k}\in\mathbb{Z}^3} \mathbf{c}_{\mathbf{k}}(t) U_{\mathbf{k}}(\mathbf{x}), \\ U_{\mathbf{k}}(\mathbf{x}) &= e^{2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}} = U_{kx}(x)U_{ky}(y)U_{kz}(z), \\ U_k(\xi) &= e^{2\pi \mathrm{i}k\xi}, \qquad (k = kx, ky, kz, \quad \xi = x, y, z). \end{aligned} \tag{3.5}$$

Here $\mathbf{k}$ is the wavenumber. The complex valued functions, $U_{\mathbf{k}}$, constitute to an orthonormal basis with respect to the inner product $\langle\cdot,\cdot\rangle_3$. We introduce the interpolant of $U_{\mathbf{k}}$: $\widetilde{U}_{\mathbf{k}} = \Phi[U_{\mathbf{k}}]$. When the interpolation operator Φ is linear, we find

$$\epsilon^2 = \frac{\left\langle \left\| \sum_{\mathbf{k}} \mathbf{c}_{\mathbf{k}} (U_{\mathbf{k}} - \widetilde{U}_{\mathbf{k}}) \right\|_3^2 \right\rangle_T}{\sum_{\mathbf{k}} \langle |\mathbf{c}_{\mathbf{k}}|^2 \rangle_T}. \tag{3.6}$$

Note that also high order interpolation methods can have a linear operator Φ. In Appendix D we prove that all $U_{\mathbf{k}} - \widetilde{U}_{\mathbf{k}}$ are orthogonal for different $\mathbf{k}$ with respect to the inner product $\langle\cdot,\cdot\rangle_3$. In this case we get:

$$\epsilon^2 = \frac{\sum_{\mathbf{k}} \epsilon_{\mathbf{k}}^2 \langle |\mathbf{c}_{\mathbf{k}}|^2 \rangle_T}{\sum_{\mathbf{k}} \langle |\mathbf{c}_{\mathbf{k}}|^2 \rangle_T}, \qquad \epsilon_{\mathbf{k}} = \|U_{\mathbf{k}} - \widetilde{U}_{\mathbf{k}}\|_3. \tag{3.7}$$

Now $\epsilon_{\mathbf{k}}$ can be calculated without having to do a simulation. Next, we require Φ to satisfy the property

$$\tilde{U}_{\mathbf{k}} = \Phi[U_{\mathbf{k}}] = \Phi[U_{kx}U_{ky}U_{kz}] = \varphi[U_{kx}]\varphi[U_{ky}]\varphi[U_{kz}] = \tilde{U}_{kx}\tilde{U}_{ky}\tilde{U}_{kz}, \tag{3.8}$$

which is the case for almost all interpolation methods. $\varphi[\cdot]$ is the one dimensional variant of the operator $\Phi[\cdot]$. Property (3.8) is used to prove the continuity of the interpolation field. Furthermore, it is used to build the three-dimensional interpolation out of one-dimensional interpolations, which saves computing time. Using that $\|U_k\|_1 = 1$, $\epsilon_{\mathbf{k}}^2$ can be written as

$$\epsilon_{\mathbf{k}}^2 = 1 + s_1(kx)s_1(ky)s_1(kz) - 2s_2(kx)s_2(ky)s_2(kz), \tag{3.9}$$

with

$$\begin{aligned} s_1(k) &= \|\widetilde{U}_k\|_1^2, \\ s_2(k) &= \langle \widetilde{U}_k, U_k \rangle_1. \end{aligned} \tag{3.10}$$

Now combining (3.7) with (3.9) and (3.10) gives us a method for the calculation of the interpolation error. This method is based on the assumptions that the system is ergodic and that there is no preferential position for the particles. Because we consider isotropic turbulence the system should be ergodic. The second assumption is not always true. Some particles will cluster depending on the size and the density of the particles. However, fluid particles are not able to cluster due to the fact that we consider incompressible flows. The advantage of this method over using relation (3.1) is that no simulations of turbulence have to be done when $\langle |\mathbf{c}_{\mathbf{k}}|^2 \rangle_T$ is known.

3.3 Approximation of the interpolation error

In this section the error estimate ϵ is further simplified. In (3.7) a summation must be taken over the three-dimensional vector $\mathbf{k}$. In order to evaluate only a one-dimensional sum one can use that the flow is isotropic, i.e., $\langle |\mathbf{c}_{\mathbf{k}}|^2 \rangle_T = \langle |\mathbf{c}_k|^2 \rangle_T$ for $k = |\mathbf{k}|$. In the end this results in a practical method that only needs the energy spectrum to predict the interpolation error. If the flow is

isotropic the three-dimensional energy spectrum is spherically symmetric, $\langle|\mathbf{c_k}|^2\rangle_T = \langle|\mathbf{c}_k|^2\rangle_T$. We can use this to write the interpolation error like

$$\epsilon_{\text{iso}}^2 = \frac{\sum_k \epsilon_k^2 k^2 \langle|\mathbf{c}_k|^2\rangle_T}{\sum_k k^2 \langle|\mathbf{c}_k|^2\rangle_T}, \qquad \epsilon_k^2 = [\epsilon_{\mathbf{k}}^2]_{|\mathbf{k}|=k}. \tag{3.11}$$

where $[\cdot]_{|\mathbf{k}|=k}$ denotes the space average over the surface of a sphere in k-space with radius k. Here the approximation is made that k-space is continuous instead of discrete. This is a good approximation for large k, because many modes have $|\mathbf{k}| = k$. For small k this approximation is less accurate but these modes have a small contribution to the interpolation error because they are very well approximate. These functions are very slowly changing and therefore the interpolation gives small errors. Note that $k^2\langle|\mathbf{c}_k|^2\rangle_T$ is proportional to the energy of the modes with $k = |\mathbf{k}|$. In this way the integrated energy spectrum in combination with ϵ_k is sufficient to calculate the error. In order to be able to compute $[\epsilon_{\mathbf{k}}^2]_{|\mathbf{k}|=k}$ easily, the following derivation is made. Starting from the second relation in (3.7) and introducing $e_k = U_k - \widetilde{U}_k$ one finds

$$\epsilon_{\mathbf{k}}^2 = \|U_{kx}U_{ky}U_{kz} - (U_{kx} - e_{kx})(U_{ky} - e_{ky})(U_{kz} - e_{kz})\|_3^2. \tag{3.12}$$

We assume that the error is relatively small compared to the actual Fourier component. Under this assumption we have that $\|e_k\|_1 \ll \|U_k\|_1$ and only the lowest powers of e_k need to be taken into account. Using that $\|U_k\|_1 = 1$ one finds

$$\begin{aligned}\epsilon_{\mathbf{k}}^2 &\approx \|e_{kx}U_{ky}U_{kz} + U_{kx}e_{ky}U_{kz} + U_{kx}U_{ky}e_{kz}\|_3^2 \\ &\leq \|e_{kx}\|_1^2 + \|e_{ky}\|_1^2 + \|e_{kz}\|_1^2.\end{aligned} \tag{3.13}$$

The $\leq$ sign is due to the triangle inequality of a norm $\|f+g\| \leq \|f\| + \|g\|$. We restrict ourselves to interpolations based on polynomial functions and we define the order n of the method as follows. n is the highest degree of a polynomial function for which the interpolation is still exact. When the order of the interpolation method is known the following approximation can be made

$$\|e_k\|_1^2 \quad \approx \quad ck^{2(n+1)}, \tag{3.14}$$

where c is some constant. The reason for this formula is the following. Given that a method has order n the amplitude of e_k is proportional to the $(n+1)$–th derivative of U_k. From this one gets that e_k is proportional to k^{n+1}. This is also shown by Figure 3.4 in Section 3.6. Using this one finds that

$$\epsilon_{\mathbf{k}}^2 \quad \approx \quad ck_x^{2(n+1)} + ck_y^{2(n+1)} + ck_z^{2(n+1)}. \tag{3.15}$$

Next, the average needs to be taken over a spherical surface, $[\epsilon_{\mathbf{k}}^2]_{|\mathbf{k}|=k}$. Because of symmetry reasons we only need to calculate the contribution of $ck_z^{2(n+1)}$. The contribution of the other terms are equal to the contribution of this term. The calculation for the surface average is done in spherical coordinates $\mathbf{k} = k(\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi)$ as follows,

$$\begin{aligned}\frac{1}{3}[\epsilon_{\mathbf{k}}^2]_{|\mathbf{k}|=k} \approx c[k_z^{2(n+1)}]_{k=|\mathbf{k}|} &= \frac{c}{4\pi k^2}\int_0^{\pi}\int_0^{2\pi}(k\cos(\phi))^{2(n+1)}k^2\sin(\phi)\mathrm{d}\theta\mathrm{d}\phi \\ &= \frac{ck^{2(n+1)}}{4\pi}\int_0^{\pi}\int_0^{2\pi}\cos^{2(n+1)}(\phi)\sin(\phi)\mathrm{d}\theta\mathrm{d}\phi = \frac{ck^{2(n+1)}}{2n+3}.\end{aligned} \tag{3.16}$$

Thus we obtain:

$$\epsilon_k^2 = [\epsilon_{\mathbf{k}}^2]_{|\mathbf{k}|=k} \approx \frac{3}{2n+3}ck^{2(n+1)} \approx \frac{3}{2n+3}\|e_k\|_1^2. \tag{3.17}$$

Combining equation (3.11) with (3.17) results in a practical method that only needs the energy spectrum to predict the interpolation error. Note that some approximations are needed and therefore the result will be less accurate than in the previous section.

3.4 Interpolation methods

In this section different interpolation methods are discussed. In order to describe them, a general framework is presented. Further we show in Appendix E a special optimality property for some of the interpolation methods.

We restrict ourselves to interpolation methods that obey property (3.8) and have a linear operator Φ. We consider the one-dimensional case first. Three-dimensional interpolation methods can be constructed from the one-dimensional versions using that Φ is linear and relation (3.8). Let u be a one-dimensional velocity field and $\tilde{u}$ its interpolant: $\tilde{u} = \varphi[u]$.

In general the particle position is between two adjacent grid points and an interpolation needs to be carried out. In this section we assume that $u(x)$ is exactly described by a finite Fourier series. In practice these Fourier series are computed by our pseudo-spectral code. A Fast Fourier Transform (FFT) can be executed in order to find $u(x)$ exactly on a grid with constant grid size Δx. Using the information from adjacent grid points an interpolated velocity field can be constructed. The interpolated velocity field $\tilde{u}$ is piecewise polynomial. For each interval (x_j, x_{j+1}) we have

$$\tilde{u}(x) = \sum_{j=0}^{N-1} a_j x^j = \mathbf{a}^T \bar{\mathbf{x}}, \qquad x \in (x_j, x_{j+1}), \qquad \bar{\mathbf{x}} = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^N \end{pmatrix}. \tag{3.18}$$

Here vector $\mathbf{a}$ depends on the interval investigated and $\mathbf{a}^T$ denotes the transpose of $\mathbf{a}$. Further, N is the length of vector $\mathbf{a}$ which is one more than the degree of the polynomial. In this way we get the restriction $n \leq N-1$. So in general the particle position x_p is between two neighboring grid points (x_j, x_{j+1}). Without loss of generality we can translate and rescale x so that x_p lies in the unit interval $[0, 1]$. For a Hermite spline interpolation the values $\tilde{u}(x)$ and of its derivatives up to the order of $m \equiv N/2 - 1$ must coincide with those of the original function at $x = 0$ and $x = 1$, i.e.,

$$\frac{\mathrm{d}^l \tilde{u}}{\mathrm{d}x^l}(0) = \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(0), \qquad \frac{\mathrm{d}^l \tilde{u}}{\mathrm{d}x^l}(1) = \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(1), \qquad l = 0, .., m. \tag{3.19}$$

These derivatives are known exactly because the Fourier coefficients of u are known. Unfortunately this has the drawback that more FFTs need to be executed. Another option is to approximate the derivatives by using finite deference methods on neighboring grid points like is done by Lalescu *et al.* [19]. In this way no extra FFTs need to be computed, unfortunately this method is less accurate.

Next, we will present the general framework, and it is illustrate with an example. The example is Hermite spline interpolation with $N = 4$ and $m = 1$. So the interpolation uses the value and the derivative in two neighboring grid points to construct the interpolation polynomial. We will refer to the example with the subindex $_{\mathrm{ex}}$ attached to the variables. We have chosen for this method because it has a special property. For this interpolation method the second derivative becomes a piecewise linear function. Comparison with the actual second derivative shows that this piecewise linear function is optimal with respect to the L^2-norm, which is shown in Appendix E. Because the second derivative is related to viscous effects it is important that this is approximated well.

First, the discrete values of u and possible derivatives need to be computed on the grid. The vector $\mathbf{b}$ contains these discrete values of the velocity field. In general (and for the example) we have

$$\mathbf{b} = \mathbf{f}[u], \qquad \mathbf{f}_{\mathrm{ex}}[u] = \begin{pmatrix} u(0) \\ u(1) \\ \frac{\mathrm{d}u}{\mathrm{d}x}(0) \\ \frac{\mathrm{d}u}{\mathrm{d}x}(1) \end{pmatrix}. \tag{3.20}$$

Here $\mathbf{f}$ depends on the interpolation method and is an operator that maps a function onto a vector.

Second, the coefficients a_j of the polynomial basis need to be computed. Because φ is a linear operator, we can write without loss of generality,

$$\mathbf{a}^T = \mathbf{b}^T\mathbf{M}, \qquad \mathbf{M}_{\text{ex}} = \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \tag{3.21}$$

Here $\mathbf{M}$ is the matrix that depends on the interpolation method.

Finally, substituting relation (3.21) into (3.18) gives

$$\varphi[u](x_p) = \tilde{u}(x_p) = \mathbf{a}^T\bar{\mathbf{x}}_p = \mathbf{b}^T\mathbf{M}\bar{\mathbf{x}}_p. \tag{3.22}$$

In order to compute the derivative of $\tilde{u}(x)$ the polynomial basis functions should be differentiated. This can be done by multiplying $\mathbf{a}$ by the differentiation matrix $\mathbf{D}$, so

$$\mathbf{a}^{(1)T} = \mathbf{a}^T\mathbf{D}, \qquad \mathbf{D} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \ddots & & & \vdots \\ 0 & 2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & N & 0 \end{pmatrix} \tag{3.23}$$

where $\mathbf{a}^{(1)}$ contains the coefficients for the derivative. Now we have,

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}(x_p) = \mathbf{a}^{(1)T}\bar{\mathbf{x}}_p = \mathbf{b}^T\mathbf{M}\mathbf{D}\bar{\mathbf{x}}_p. \tag{3.24}$$

In this way we have created a theoretical framework that can describe the interpolation methods. This framework can be used to implement the interpolation methods in a straight forward way. In Section 3.5 it is used to generate fast algorithms for the implementation of the method. Next we are going to extend the method to the three-dimensional case.

The three-dimensional interpolation for a scalar field is carried out applying three times one-dimensional interpolations, see Figure 3.1. The interpolation consists of three steps, in which the three spatial directions are interpolated one after each other. The order in which the spatial directions are interpolated does not matter. Furthermore, splitting the three-dimensional interpolation into one-dimensional versions can be done for all interpolation methods as long as two conditions are met. First, the interpolation method must have a linear interpolation operator Φ and, second, condition (3.8) must be met. Since the interpolation operator is linear, if a property holds for one arbitrary mode than this property carries over to the complete solution. When considering one mode we can use condition (3.8) to see that one dimensional interpolations can be done in the way shown by Fig. 3.1. From this equation one can also see that it does not matter which direction is interpolated first. With the same reasoning one can easily show that the three-dimensional interpolation has the same degree of smoothness as the one-dimensional case.

The extension for the general framework to the three dimensional case goes as follows. The vector $\mathbf{b}$ becomes a three dimensional tensor like

$$\mathbf{B} = (\mathbf{f}_z \circ \mathbf{f}_y \circ \mathbf{f}_x)[u], \tag{3.25}$$

where $\circ$ represents the vector outer product [29], like

$$\mathbf{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}, \qquad A_{i_1 i_2 i_3} = a^{(1)}_{i_1} a^{(2)}_{i_2} a^{(3)}_{i_3}. \tag{3.26}$$

Similar as before, $\tilde{u}(\mathbf{x}_p)$ can be represented by

$$\Phi[u](\mathbf{x}_p) = \tilde{u}(\mathbf{x}_p) = \mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}\bar{\mathbf{z}}_p), \tag{3.27}$$

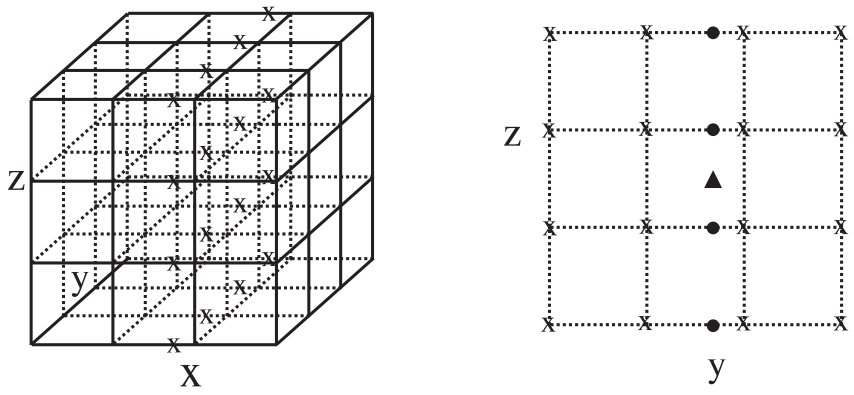

Figure 3.1: Graphical description of the three-dimensional interpolation, using three times one-dimensional interpolations for $N = 4$. First N^2 one-dimensional interpolations are carried out in the x-direction (crosses). Second N interpolations are carried out in y-direction (dots in the right figure) and from these N results finally one interpolated value is derived in z-direction (triangle).[2]

where $\bar{\times}_n$ denotes n-mode vector product [29], like

$$\mathbf{A} = \mathbf{B}\bar{\times}_n\mathbf{f}, \qquad A_{i_1\cdots i_{n-1}i_{n+1}\cdots i_{\mathcal{N}}} = \sum_{i_n} B_{i_1\cdots i_{\mathcal{N}}} f_{i_n}, \tag{3.28}$$

where $\mathcal{N}$ denotes the dimension of tensor $\mathbf{B}$. For the derivative $\frac{\partial\tilde{u}}{\partial x}(\mathbf{x}_p)$ one gets

$$\frac{\partial\tilde{u}}{\partial x}(\mathbf{x}_p) = \mathbf{B}\bar{\times}_3(\mathbf{M}^{(1)}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}\bar{\mathbf{z}}_p), \tag{3.29}$$

where $\mathbf{M}^{(1)} = \mathbf{MD}$. Likewise also the derivatives $\frac{\partial\tilde{u}}{\partial y}(\mathbf{x}_p)$ and $\frac{\partial\tilde{u}}{\partial z}(\mathbf{x}_p)$ can be computed. When the scaler field $u(\mathbf{x})$ becomes a vector field $\mathbf{u}(\mathbf{x})$, tensor $\mathbf{B}$ becomes four dimensional where the last dimension contains the three components of $\mathbf{u}$.

3.5 Implementation

For a fast calculation the algorithm used is essential. In this section we show fast change for the interpolation. When even a little difference is made in the algorithm the efficiency can drop easily by a factor of two or even more.

Relations (3.27) and (3.29) provide in a good starting point for an efficient implementation of the interpolation. The matrices $\mathbf{M}$ and $\mathbf{M}^{(1)}$ only need to be computed once which can be done first. Second vectors $\bar{\mathbf{x}}_p$, $\bar{\mathbf{y}}_p$ and $\bar{\mathbf{z}}_p$ can be computed which only needs to be done once for each position. In Table 3.1 we keep track of all the computed quantities. One flop denotes one multiplication with one addition. Further we show the number of flops for the general case and for $N = 4$. In this case we look at the three-dimensional velocity field $\mathbf{u}(\mathbf{x})$. Because $\mathbf{u}$ is now a vector, tensor $\mathbf{B}$ becomes one dimension larger. Now tensor $\mathbf{B}$ is four dimensional where the last dimension contains the three components of $\mathbf{u}$. For the Maxey and Riley equations [4] also all the derivatives are needed so they are computed as well. Table 3.1 shows a fast algorithm for the overall computation of these components. The main idea is to reduce the dimension of the tensors as soon as possible in order to generate an efficient method.

Table 3.1: Algorithm for interpolation, with computational costs

Computed	Number of flops	Number of flops for $N = 4$
$\bar{\mathbf{x}}_p$, $\bar{\mathbf{y}}_p$ and $\bar{\mathbf{z}}_p$	$3N$	12
$\mathbf{M}\bar{\mathbf{x}}_p$, $\mathbf{M}\bar{\mathbf{y}}_p$ and $\mathbf{M}\bar{\mathbf{z}}_p$	$3N^2$	48
$\mathbf{M}^{(1)}\bar{\mathbf{x}}_p$, $\mathbf{M}^{(1)}\bar{\mathbf{y}}_p$ and $\mathbf{M}^{(1)}\bar{\mathbf{z}}_p$	$3N(N-1)$	36
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)$	$3N^3$	192
$\mathbf{B}\bar{\times}_3(\mathbf{M}^{(1)}\bar{\mathbf{x}}_p)$	$3N^3$	192
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)$	$3N^2$	48
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}^{(1)}\bar{\mathbf{y}}_p)$	$3N^2$	48
$\mathbf{B}\bar{\times}_3(\mathbf{M}^{(1)}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)$	$3N^2$	48
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}\bar{\mathbf{z}}_p)$	$3N$	12
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}^{(1)}\bar{\mathbf{z}}_p)$	$3N$	12
$\mathbf{B}\bar{\times}_3(\mathbf{M}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}^{(1)}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}\bar{\mathbf{z}}_p)$	$3N$	12
$\mathbf{B}\bar{\times}_3(\mathbf{M}^{(1)}\bar{\mathbf{x}}_p)\bar{\times}_2(\mathbf{M}\bar{\mathbf{y}}_p)\bar{\times}_1(\mathbf{M}\bar{\mathbf{z}}_p)$	$3N$	12
Total:	$6N^3 + 15N^2 + 12N$	672

In order to see how efficient this algorithm is one can compare the computational costs with a lower bound. The lower bound we use is the size of $\mathbf{B}$ which is $3N^3$. In order to be able to use all the information in tensor $\mathbf{B}$ also $3N^3$ flops are needed for the computation. For large N one finds that the algorithm of Table 3.1 is only a factor 2 less efficient than this lower bound. Note that there might be higher lower bounds and therefore this factor can be less. When the derivatives are not needed, less needs to be calculated and the algorithm becomes as efficient as the lower bound for the case that N goes to infinity.

Now we examine the Hermite spline interpolation with $N = 4$. We are going to compare our algorithm with the algorithm proposed by Lekien and Marsden [18]. First we need to show that their method is equivalent to ours. This can be easily seen by the fact that their multiplication matrix is equal to $\mathbf{M}_{\text{ex}}^T \otimes \mathbf{M}_{\text{ex}}^T \otimes \mathbf{M}_{\text{ex}}^T$ after some columns are swapped. The swapping of columns is due to the definition of vectors. Further the superscript T denotes the transpose and $\otimes$ is the Kronecker tensor product. The method of Lekien and Marsden is based on the fact that they first calculate the coefficients for the polynomial basis once. After this is done they can calculate the actual velocity or derivatives at many points in the interval evaluated. According to Lekien and Marsden this method is beneficial when the derivatives are needed or the interpolation needs to be done multiple times for one element. Our algorithm already involves the derivatives, therefore this does not matter. Because the coefficients of the polynomial basis still have all the information, the tensor containing these coefficients has again size $3N^3$. This means that even after the calculation of the polynomial basis the method is bounded by $3N^3$ flops. Further their study does not provide in an algorithm that combines the calculation of the different derivatives. This means that these derivatives have to be calculated separately and the lower bound becomes $12N^3$ flops. Note that our algorithm of Table 3.1 is already faster. But even when all interpolations need to be done in one interval and an algorithm similar to ours would be constructed that combines the calculation of the derivatives it would be only a factor of 1.14 faster. Further the computation of the coefficients of the polynomial basis could be done a factor 5 faster by calculating it like $\mathbf{B} \times_3 \mathbf{M} \times_2 \mathbf{M} \times_1 \mathbf{M}$. Here $\times_n$ denotes the n-mode product [29]. For our proposed algorithm this is not needed therefor we do not go into details.

Our algorithm of Table 3.1 is only a factor 3.5 less efficient than our lower bound for $N = 4$. When for example methods of Lalescu *et al.* [19] are implemented in a straight forward way without checking that unnecessary computations are avoided, $18N^4(N + 1)$ flops are needed. For the case that $N = 4$ it is a factor 120 less efficient than our lower bound. This explains the large factors found by Lekien and Marsden [18]. In our case the computation time is reduced by a factor 34 compared with the most straight forward implementation of the same interpolation method.

Apart from the advantages of the Hermite spline interpolations, there is also the disadvantage that exact derivatives are needed. This means that more FFTs must be carried out in order to compute these derivatives. In order to fill tensor $\mathbf{B}$ (3.25) for Hermite spline interpolation with $N = 4$ the following quantities at the grid are needed:

$$\mathbf{u}, \frac{\partial \mathbf{u}}{\partial \mathrm{x}}, \frac{\partial \mathbf{u}}{\partial \mathrm{y}}, \frac{\partial \mathbf{u}}{\partial \mathrm{z}}, \frac{\partial^2 \mathbf{u}}{\partial \mathrm{x} \partial \mathrm{y}}, \frac{\partial^2 \mathbf{u}}{\partial \mathrm{x} \partial \mathrm{z}}, \frac{\partial^2 \mathbf{u}}{\partial \mathrm{y} \partial \mathrm{z}}, \frac{\partial^3 \mathbf{u}}{\partial \mathrm{x} \partial \mathrm{y} \partial \mathrm{z}}. \tag{3.30}$$

This means that 8 full three-dimensional FFT-s must be carried out. In practice a full three dimensional FFT is carried out in the following way:

$$\mathbf{u} = \mathbf{F}[\mathbf{U}] = F_x F_y F_z[\mathbf{U}], \qquad \frac{\partial \mathbf{u}}{\partial \mathrm{x}} = \mathbf{F}_{\partial x}[\mathbf{U}] = F_{\partial x} F_y F_z[\mathbf{U}]. \tag{3.31}$$

Here $\mathbf{F}$ denotes the the full three-dimensional FFT, and F_x is a full one-dimensional FFT in x-direction. Further the index ∂x denotes the partial derivative. In order to compute one full three-dimensional FFT three full one-dimensional FFT-s are done after each other in where the order of these full one-dimensional FFT-s does not matter. In this way 24 full one-dimensional FFT-s must be carried out. Avoiding double computations we start with computing $F_z[\mathbf{U}]$ and $F_{\partial z}[\mathbf{U}]$. After this, the operators F_y an $F_{\partial y}$ act on the results. Finally the operators F_x an $F_{\partial x}$ act on the last results and all the quantities of (3.30) are computed. In this way only 2+4+8=14 full one dimensional FFT-s must be carried out. This saves a factor of $24/14 \approx 1.7$ in computation time.

3.6 Results

In this section results are shown. First the discretisation error is computed. Second the interpolation error is computed for different interpolation methods. These errors are computed in the proposed ways to show if these methods give comparable results. Further the energy spectrum is calculated for the interpolated field.

We start with the calculation of the discretisation error ε. Because we need to compare it with the interpolation error the same norm must be taken for both. This would mean that,

$$\varepsilon^2 = \frac{\langle \|\mathbf{u} - \widehat{\mathbf{u}}\|_3^2 \rangle_T}{\langle \|\widehat{\mathbf{u}}\|_3^2 \rangle_T} \tag{3.32}$$

needs to be computed, where $\widehat{\mathbf{u}}$ is the exact velocity field. Unfortunately this will not work. Because of the chaotic behavior of turbulence even the smallest change can eventually result in a completly different velocity field. Because of this one can only look at statistical data and estimate the error from this. Therefore the following is used

$$\varepsilon^2 = \frac{\left(\langle \|\mathbf{u}\|_3 \rangle_T - \langle \|\widehat{\mathbf{u}}\|_3 \rangle_T\right)^2}{\langle \|\widehat{\mathbf{u}}\|_3^2 \rangle_T}. \tag{3.33}$$

Now $\mathbf{u}$ can be expanded in a Fourier series like is done before and the following result is found

$$\varepsilon^2 = \frac{\sum_k \left(\langle |k\mathbf{c}_k| \rangle_T - \langle |k\widehat{\mathbf{c}}_k| \rangle_T\right)^2}{\sum_k \langle k^2 |\widehat{\mathbf{c}}_k|^2 \rangle_T}. \tag{3.34}$$

Now one can use the energy spectrum like before to compute this error as well. The advantage of using the energy spectrum is that the error becomes mode dependent. This discriminates the contribution of different modes to the total error. In this way one can see which modes have the biggest contribution to the error.

For the computation of the discretisation error two simulations need to be done. In order to simulate approximate the exact field a simulation is done with double spatial resolution, here the discretisation error is that much smaller that it can be neglected. This is due to the fact that the

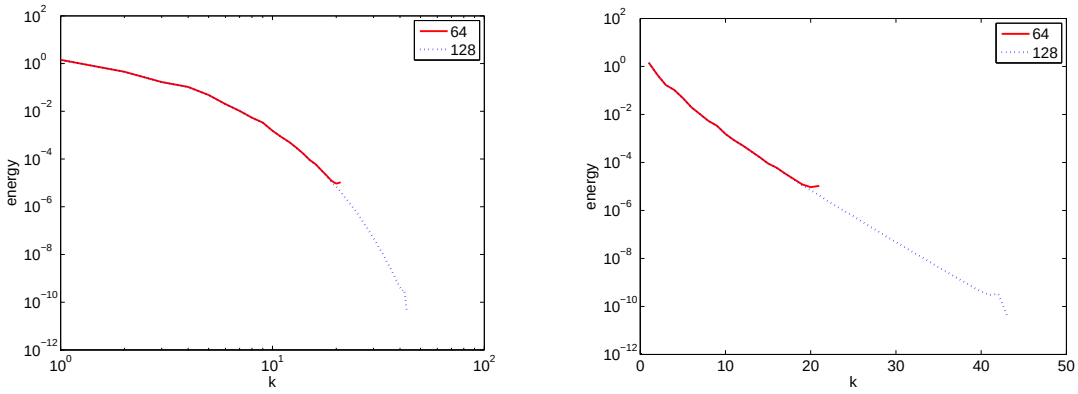

Figure 3.2: Energy spectra plotted in log-log scale and log-lin scale.

smallest scales with the highest wave numbers are in the viscous range, where exponential decay is expected. For the two simulations everything is kept the same apart from the resolution. Further time steps are small enough in order to be able to neglect the time integration error as well. This is verified by running two simulations where one has double resolution in time compared with the other one. The difference could be neglected and we continued our simulations with the highest time resolution. This is done in order to be sure that errors are due to the spatial discretisation error.

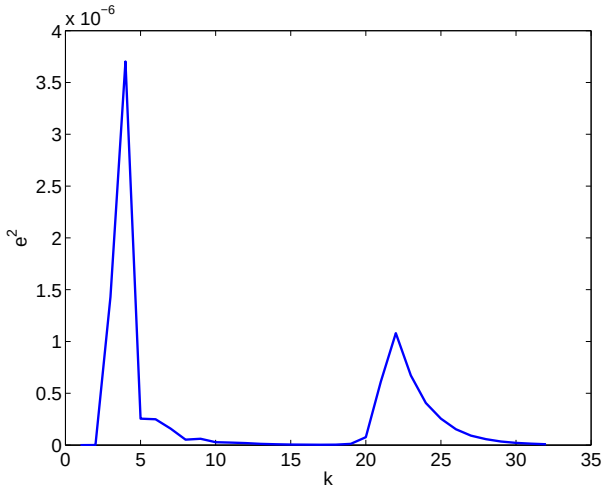

Figure 3.3: discretisation error ε computed by (3.34) by using the data from Figure 3.2.

The two simulation are computed on a 64^3 and 128^3 cubed grid with the dimensionless viscosity 0.032 and dimensionless time step $\Delta t = 0.0016$. The simulations are carried out for around 100 large eddy turnover times in order to get reliable statistical data. Further in order to avoid aliasing in the nonlinear term $k_{\max}\Delta x/2 = \frac{2}{3}$, where $k_{\max}$ is the maximal wave number. The energy spectra are given by Fig. 3.2. Here it can be seen that the biggest relative error is made in the highest wave numbers. This is also expected because they are most influenced by the resolution. Next Fig. 3.3 gives the discretisation error ε computed by (3.34). Here one can see that suddenly two local maxima appear in the error. The left maxima is due to the statistical error. The lowest modes have the biggest contribution to the statistical error because they contain most energy. Further, because the energy spectra changes over many orders of magnitude this statical error is still visible. Running longer simulations would reduce this statistical error, therefor we dismiss

this part of the error. So for the contribution of the discretisation error we consider the modes: $k \in (15, k_{\max})$. In this way we found that the discretisation error ε was $1.88 \cdot 10^{-3}$.

Table 3.2: Interpolation error for different methods

method	ϵ	ϵ_{iso}	n	N	continuity	description
1	$9.01 \cdot 10^{-3}$	$7.00 \cdot 10^{-3}$	1	2	c^0	linear interpolation
2	$1.30 \cdot 10^{-3}$	$1.28 \cdot 10^{-3}$	2	4	c^1	appr. derivative
3		$3.91 \cdot 10^{-4}$	4	6	c^2	appr. 1st and 2nd derivative
4		$1.02 \cdot 10^{-4}$	3	4	c^1	real derivative
5	0	0	∞	∞	c^∞	FFT

For the computation of the interpolation error ϵ the different proposed methods for computing the error can be used. First we calculate the error with relations: (3.7), (3.9) and (3.10). Second it is calculated by (3.11) and (3.17). The results are shown in Table 3.2. For the second method, $\|e_k\|_1^2$ needs to be calculated and it is shown by Fig. 3.4. From Table 3.2 several things can be observed. First we see that our practical method, that only uses the energy spectrum is useful to predict the correct order order of magnitude. Second the order of the interpolation method does not give all the information. One can see that not in all cases a higher order method gives smaller errors. When comparing the discretisation error with the interpolation error one can see that linear interpolation is not good enough for this case. Higher order methods have to be chosen to make the errors comparable.

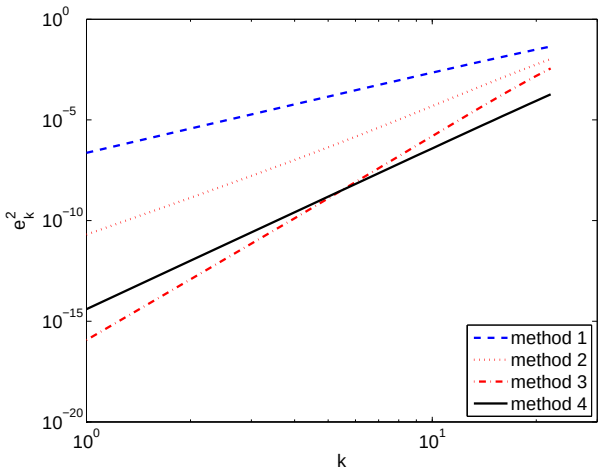

Figure 3.4: Error from interpolation method ϵ_k^2 computed by (3.17)

Using the theory from Appendix D one can even construct the energy spectrum after the interpolation is done. This is shown in Fig. 3.5. Here one can see that two things are changed. First the energy in the highest modes is lower after the interpolation. Second energy in modes higher than k_{max} is introduced due to the interpolation method. Note that the newly introduced energy in the higher modes is relatively small compared with the total energy in the spectrum.

3.7 Conclusions

We have introduced different practical methods for computing the interpolation error. First we have introduced an accurate method that uses the full three dimensional energy spectrum to compute this error. Second we introduced a practical method that only needs the one dimensional energy spectrum to compute the same error. These methods are validated by full turbulent simulations and it is shown that they give comparable results.

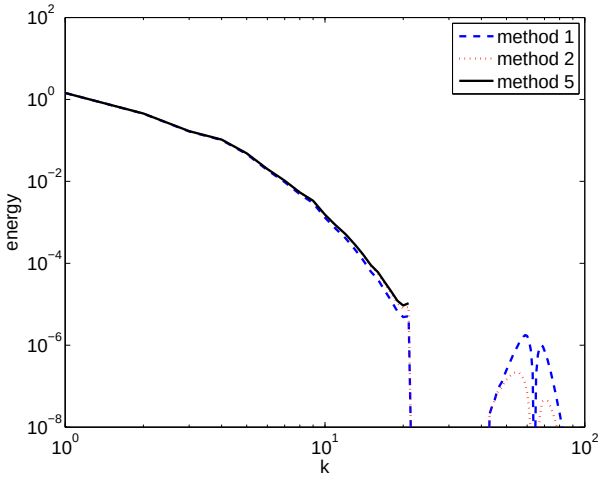

Figure 3.5: energy spectrum after carrying out the interpolation

Further a general framework is proposed for different interpolation methods. Making use of this framework an algorithm is proposed for fast computation of the interpolation. This can easily save an order of magnitude in computing time compared with other algorithms. It is shown that that the number of flops needed for this algorithm is close to a theoretical lower bound.

In order to avoid unnecessary computations the discretisation and interpolation error should be of the same order of magnitude. The methods introduced give a practical way of estimating the errors. Comparison of these errors shows that linear interpolation is not always sufficient. Further it is show how the energy spectrum is changed due to the interpolation.

Chapter 4

Conclusions

In this study we have investigated three different topics concerning particle tracking in turbulence. We have investigated the Basset history force, the interpolation and the Faxén forces. In short we will repeat the main conclusions.

For the Basset history force, we have introduced a novel method for the evaluation of the Basset force kernel and analysed several aspects of its implementation. The tail of the Basset force kernel is approximated by exponential functions. The contribution of these exponential functions can be calculated in a recursive way which makes it very efficient. Typically the use of the tail kernel reduces the computational costs of the Basset force by more than an order of magnitude, whereas the memory requirement is reduced even more. Furthermore, the error in the tail of the Basset force is also reduced by more than an order of magnitude in comparison with the traditional window method. Further a trapezoidal-based method is developed in order to deal with the singularity of the Basset force. This method has a temporal accuracy of $\mathcal{O}\left(\Delta t^2\right)$ where other methods only have a temporal accuracy of $\mathcal{O}\left(\Delta t\right)$ or lower. This method is made partially implicit in order to make it more stable. Thereafter the method has been implemented in a tracking algorithm for (light) inertial particles in turbulent flows. The isotropic turbulence simulation shows that the error made by the window method can influence statistics on the particle trajectories. This has been illustrated with the velocity and acceleration spectra. Therefore, the novel exponential method is preferred over the classical window method. Because the new implementation is much faster than the classical one, more particles can be taken into account in simulations, which opens possibilities for further research.

For the interpolation we have introduced different practical methods for computing the interpolation error. First we have introduced a very accurate method that uses the full three-dimensional energy spectrum to compute this error. Second we have introduced a practical method that only needs the one-dimensional energy spectrum to approximate the same error. These methods are validated by full turbulence simulations and it is shown that they give comparable results. Further a general framework is proposed for different interpolation methods. Making use of this framework an algorithm is proposed for fast implementation of the interpolation. It is shown that the number of flops needed for this algorithm is close to a theoretical lower bound. This algorithm can easily save an order of magnitude in computing time compared with other algorithms. In order to avoid unnecessary computations the discretisation and interpolation errors should be of the same order of magnitude. The methods introduced give a practical way of comparing them. Further it is show how the energy spectrum is changed due to the interpolation. Comparison of the errors for our test case shows that linear interpolation is not sufficient.

Finally in Appendix A we propose a method for the implementation of the Faxén corrections by changing the velocity field. This method is based on exact integration and not on a finite order approximation of ka as is done in most studies. Recently, it has been shown that these exact averages for the calculation of the Faxén corrections give more reliable results for further increasing the particle radii. This new method provides a fast and exact way of computing them.

Acknowledgements

This report marks the completion of a journey in obtaining my degree in both Applied Physics and Industrial & Applied Mathematics. I would not have come this far without the help and encouragements of many people. Therefore I would like to thank the following persons in particular:

Foremost, I would like to express my sincere gratitude to my supervisors: Prof. Dr. Herman Clercx, Dr. Ir. Jan ten Thije Boonkkamp and Dr. Ir. Bas van de Wiel. My first scientific publication would not have gone this smoothly without the help of Herman and Jan. Where Herman was always there for the clear physical explanation, Jan made sure that all the mathematical details were correct. And although it is almost impossible to replace Herman, Bas was always there to ask questions and to give comments if necessary. I will always remember the nice talks about your ongoing work.

Besides my direct supervisors I am grateful to my future promoter Prof. Dr. Federico Toschi. Thanks to you and Herman I got the possibility to go to the international school "Fluctuations and Turbulence in the Microphysics and Dynamics of Clouds" in Porquerolles. This school gave me new ideas for my study. Furthermore, we had interesting conversations about the topics of this thesis. This has definitely helped me to understand some physical aspects. You, together with my supervisors, formed a team in which many different disciplines are represented. Without this team I could not have made such progress in this field.

Besides my advisors, I would like to thank the rest of my graduation committee: Dr. J.D.R. Harting and Dr. S.W. Rienstra for their contribution to the committee.

I would like to thank Marleen van Aartrijk for helping me to understand the numerical code. Whenever I had a problem, she always provided me with an answer to my e-mails. Further I would like to thank Ben van den Broek for contributing to the derivation of the analytical solution presented in Appendix C.

I thank my colleagues at TU/e for the enjoyable time. Especially to my friends from my office for providing a stimulating environment. Only the panpipe music with Christmas did not contribute to this, but hiding the L.P. provided a good solution.

I wish to thank my friends from All Terrain for all the support and helping me to get through difficult times. Sporting has always helped me to clear my mind. In this way one can start with a fresh mind on the next day.

Last but not the least, I am grateful to my family for their support and motivation during my study. To my sister Esther for helping me creating some of the figures. To my parents Fons and Netty for their motivation, (financial) support and providing a loving environment throughout my study.

Appendix A

Faxén corrections

In this appendix we propose a method for the implementation of the Faxén corrections. The Faxén corrections are first order corrections in the MR equations for increasing particle radii. In most studies these corrections have been implemented by using the Laplacian of the flow field. This gives more reliable results for increasing particle radii compared to simulations without these corrections[21]. Recently it has been shown that using volume and surface averages for the calculation of the Faxén corrections allows for even larger particle radii [20]. In this section a method is proposed that does the averaging exactly in spectral space. Besides that the method is more accurate it is also faster.

The Faxén corrections are realized by changing the velocity field. The changed velocity field is computed by multiplying the Fourier components of the original velocity field by predetermined factors. How to compute these factors and why this can be done is explained in this appendix.

We start with a stationary velocity field $u(\mathbf{x})$, for now this velocity field consists of only one single Fourier component with wavenumber k, $u(\mathbf{x})$ given by

$$u(\mathbf{x}) = \cos(kz). \tag{A.1}$$

The particles are treated by one-way coupling and in a Lagrangian way. Due to Faxén corrections the field should be averaged over the surface or over the volume of the particle depending on the term in the Maxey and Riley equations. The particle has radius a and its center is located at $\mathbf{x}_p$. In the limit of small particles i.e. $ka \ll 1$, the surface averaged field $[u]_s$ becomes

$$[u]_s(\mathbf{x}_p, a) = \cos(kz_p)\left(1 - \frac{1}{6}(ka)^2\right) + \mathcal{O}\left((ka)^4\right), \tag{A.2}$$

which is a third order approximation. When $ka \geq 1$ also higher order terms need to be included. Employing the Taylor series of $u(\mathbf{x})$ around $\mathbf{x}_p$, we get

$$u(\mathbf{x}) = \sum_{n=0}^{\infty} \frac{(-1)^n k^{2n}}{(2n)!} \cos(kz_p)(z - z_p)^{2n} + \sum_{n=0}^{\infty} \frac{(-1)^n k^{2n+1}}{(2n+1)!} \sin(kz_p)(z - z_p)^{2n+1}. \tag{A.3}$$

The odd functions are dismissed because they will average to zero due to symmetry reasons. The

integration over the surface of the particle is done in spherical coordinates, as follows,

$$\begin{aligned}
[u]_s(\mathbf{x}_p, a) &= \frac{1}{4\pi a^2}\iint_{|\mathbf{x}-\mathbf{x}_p|=a} u(\mathbf{x})\mathrm{d}S, \\
&= \frac{\cos(kz_p)}{4\pi}\int_0^{\pi}\int_0^{2\pi}\sum_{n=0}^{\infty}\frac{(-1)^n k^{2n}}{(2n)!}(a\cos\phi)^{2n}\sin(\phi)\mathrm{d}\theta\mathrm{d}\phi, \\
&= \frac{\cos(kz_p)}{2}\sum_{n=0}^{\infty}\frac{(-1)^n (ka)^{2n}}{(2n)!}\int_0^{\pi}\cos^{2n}(\phi)\sin(\phi)\mathrm{d}\phi, \\
&= \cos(kz_p)\sum_{n=0}^{\infty}\frac{(-1)^n (ka)^{2n}}{(2n+1)!}. \qquad \text{(A.4)}
\end{aligned}$$

The surface average can also be written as

$$[u]_s(\mathbf{x}_p, a) = \cos(kz_p)\frac{\sin(ka)}{ka}. \qquad \text{(A.5)}$$

In order to find the volume average $[u]_v(\mathbf{x}_p)$ we integrate the surface average like

$$\begin{aligned}
[u]_v(\mathbf{x}_p, a) &= \frac{1}{\frac{4}{3}\pi a^3}\iiint_{|\mathbf{x}-\mathbf{x}_p|\leq a} u(\mathbf{x})\mathrm{d}V, \\
&= \frac{3}{4\pi a^3}\int_0^a 4\pi r^2 [u]_s(\mathbf{x}_p, r)\mathrm{d}r, \\
&= \cos(kz_p)\frac{3}{ka^3}\int_0^a \sin(kr) r\mathrm{d}r. \qquad \text{(A.6)}
\end{aligned}$$

Integration by parts gives

$$[u(\mathbf{x}_p)]_v = \cos(kz_p)\frac{3}{k^3a^3}\big(\sin(ka) - \cos(ka)ka\big). \qquad \text{(A.7)}$$

The general velocity modes in Fourier space will have the following form

$$u(\mathbf{x}) = \cos(k_x x)\cos(k_y y)\cos(k_z z). \qquad \text{(A.8)}$$

Because the averages are taken radially symmetric the general case can be easily deduced from the special case by the relation.

$$k^2 = k_x^2 + k_y^2 + k_z^2. \qquad \text{(A.9)}$$

Fig. A.1 shows the multiplication factors for the Fourier components for the surface and the volume averages. All curves have a multiplication factor of 1 in the limit of low ka which is expected. In this case the particle radius is much smaller than the wave length and taking the local average does not change the wave. For high ka different limits are found for the different curves. Where the actual averages go to zero, the finite order methods go to plus or minus infinity. From a physical point of view it is expected that this limit is zero. When the particle is much larger than the wavelength one expects that the contribution of this wave averages out and becomes negligible. This is found for the exact averages but not for finite order methods. Because these exact averages have the right limit for high ka it is expected that they give more reliable results. This is supported by the results of Calzavarini *et al.* [20].

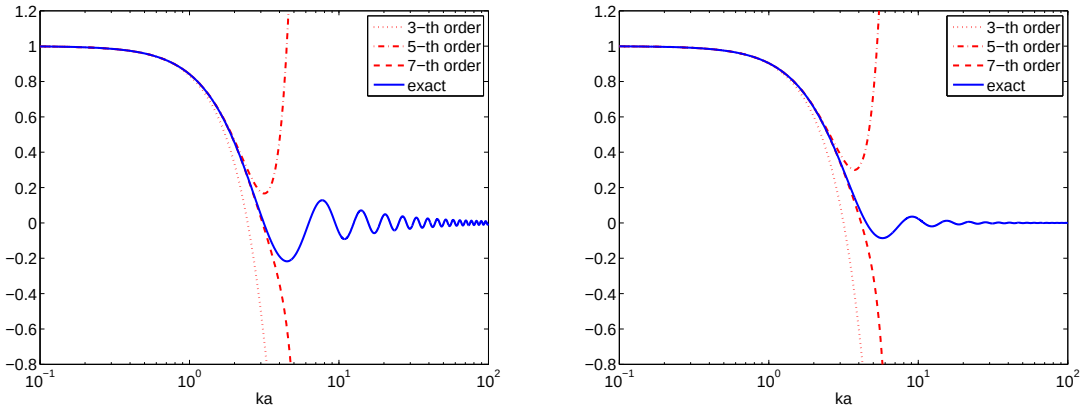

Figure A.1: multiplication factors for the Fourier components for the surface (left) and volume (right) averages.

Appendix B

Flow field for circular particle trajectories

In this appendix the space dependent velocity field is derived that allows a circular particle trajectory as solution of the MR equation. Suppose the particle trajectory and velocity is given by

$$\mathbf{x}_p = r\mathcal{R}\left[\mathbf{e}\ e^{-ikt}\right], \qquad \mathbf{u}_p = -rk\mathcal{R}\left[i\mathbf{e}\ e^{-ikt}\right]\ , \tag{B.1}$$

with $\mathbf{e} = \mathbf{e}_x - i\mathbf{e}_y$. For the flow velocity field we are looking for solutions of (2.2) of the form

$$\mathbf{u} = -\mathcal{R}\left[s(x+iy)\mathbf{e}\right] + 2\alpha z\mathbf{e}_z\ , \tag{B.2}$$

with $s = \alpha + i\beta$ a complex constant. For $\beta = 0$ the velocity field represents a sink flow $\mathbf{u} = (-\alpha x, -\alpha y, 2\alpha z)$ and for $\alpha = 0$ it represents solid body rotation: $\mathbf{u} = (\beta y, -\beta x, 0)$. A spherical particle released in the plane $z = 0$ will remain there due to the symmetry of the flow. Substituting (B.1) and (B.2) (assuming $z = 0$) in equation (2.2), and taking into account that no gravity is applied, the Faxén corrections are 0 for a linear velocity field and $\frac{Du}{Dt} = \mathcal{R}\left[s^2(x+iy)\mathbf{e}\right]$, yields the following quadratic relation for s:

$$\begin{aligned} -\omega^2\left(m_p + \frac{1}{2}m_f\right) = 6\pi a\mu(-s+i\omega) + \frac{3}{2}m_f s^2 \\ +c_{\mathrm{B}}\sqrt{\frac{\omega\pi}{2}}(\omega+is)(1+i)\ . \end{aligned} \tag{B.3}$$

There are two solutions for s, but one of the solutions of s results in an unphysical particle trajectory and is therefore discarded.

Appendix C

Time dependent velocity field

In this appendix the particle trajectory is derived given the uniform, time dependent velocity field (2.31). The particle is released with an initial velocity $\mathbf{u}_p(0)$. For a uniform velocity field Eq. (2.2) can be simplified to

$$-\left(m_p+\frac{1}{2}m_f\right)\frac{\mathrm{d}\mathbf{w}}{\mathrm{dt}}=6\pi a\mu\mathbf{w}+(m_f-m_p)\frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}}-(m_p-m_f)g\mathbf{e}_z + c_\mathrm{B}\int_0^t K_\mathrm{B}(t-\tau)\frac{\mathrm{d}\mathbf{w}(\tau)}{\mathrm{d}\tau}\mathrm{d}\tau, \quad \text{(C.1)}$$

where $\mathbf{w}=\mathbf{u}-\mathbf{u}_p$. The velocity field $\mathbf{u}$ will be expanded in a Fourier series, $\mathbf{u}(t)=\sum_{n=-\infty}^{\infty}\mathbf{u}_n e^{in\omega t}$. The Laplace transform of $\mathbf{w}$ is given by $\mathbf{W}(s)=\int_0^\infty e^{-st}\mathbf{w}(t)\mathrm{d}t$, and the Laplace transform of equation (C.1) reads

$$-\left(m_p+\frac{1}{2}m_f+c_\mathrm{B}\sqrt{\frac{\pi}{s}}\right)(s\mathbf{W}-\mathbf{w}(0))=6\pi a\mu\mathbf{W}-\frac{(m_p-m_f)}{s}g\mathbf{e}_z + (m_f-m_p)\sum_{n=-\infty}^{\infty}\mathbf{u}_n\frac{in\omega}{s-in\omega}\ . \quad \text{(C.2)}$$

Using spitting in partial fractions this yields for $\mathbf{W}(s)$

$$\mathbf{W}(s)=\frac{c}{\sqrt{s}}\left(\frac{c_+}{\sqrt{s}+c_-}-\frac{c_-}{\sqrt{s}+c_+}\right)\left(\mathbf{w}(0)-\frac{m_p-m_f}{6\pi a\mu}g\mathbf{e}_z\right)+\frac{m_p-m_f}{6\pi a\mu}\frac{g}{s}\mathbf{e}_z + \sum_{n=-\infty}^{\infty}\mathbf{c}_n\left\{\frac{1}{(c_++\sqrt{in\omega})(c_-+\sqrt{in\omega})(s-in\omega)}+\frac{\sqrt{in\omega}(c_++c_-)}{(c_+^2-in\omega)(c_-^2-in\omega)\sqrt{s}(\sqrt{s}+\sqrt{in\omega})} + \frac{c}{\sqrt{s}}\left(\frac{c_+}{(c_+^2-in\omega)(\sqrt{s}+c_+)}-\frac{c_-}{(c_-^2-in\omega)(\sqrt{s}+c_-)}\right)\right\}, \quad \text{(C.3)}$$

with c_+, c_-, c and $\mathbf{c}_n$ constants given by

$$c_\pm=\frac{c_\mathrm{B}\sqrt{\pi}\pm\sqrt{c_\mathrm{B}^2\pi-12\pi a\mu(2m_p+m_f)}}{2m_p+m_f},$$
$$c=\frac{2m_p+m_f}{2\sqrt{c_\mathrm{B}^2\pi-12\pi a\mu(2m_p+m_f)}},$$
$$\mathbf{c}_n=\frac{in\omega(m_p-m_f)\mathbf{u}_n}{m_p+\frac{1}{2}m_f}. \quad \text{(C.4)}$$

Transformation back to physical space results in

$$\mathbf{w}(t) = c\left[c_+\psi\left(c_-\sqrt{t}\right) - c_-\psi\left(c_+\sqrt{t}\right)\right]\left(\mathbf{w}(0) - \frac{m_p - m_f}{6\pi a\mu}g\mathbf{e}_z\right) + \frac{m_p - m_f}{6\pi a\mu}g\mathbf{e}_z$$
$$+ \sum_{n=-\infty}^{\infty}\mathbf{c}_n\left\{\frac{1}{(c_+ + \sqrt{in\omega})(c_- + \sqrt{in\omega})}e^{in\omega t} + \frac{\sqrt{in\omega}(c_+ + c_-)}{(c_+^2 - in\omega)(c_-^2 - in\omega)}\psi\left(\sqrt{in\omega t}\right)\right.$$
$$\left. + \frac{cc_+}{c_+^2 - in\omega}\psi\left(c_+\sqrt{t}\right) - \frac{cc_-}{c_-^2 - in\omega}\psi\left(c_-\sqrt{t}\right)\right\}, \quad \text{(C.5)}$$

with $\psi(z) = \exp\left(z^2\right)\operatorname{erfc}(z)$.

Appendix D

Proof: $U_{\mathbf{k}} - \widetilde{U}_{\mathbf{k}}$ are orthogonal

In this Section we prove that all $U_{\mathbf{k}} - \widetilde{U}_{\mathbf{k}}$ are orthogonal for different $\mathbf{k}$ with respect to the inner product $\langle \cdot, \cdot \rangle_3$. In order to show this we start with a one-dimensional linear interpolation. In the end it can be extended to a more general case.

$\varphi[\cdot]$ is the one dimensional variant of the interpolation operator $\Phi[\cdot]$. The idea is to write the operator φ with some basic operations on functions. This can be done in the following way:

$$\widetilde{U}_k = \varphi[U_k] = (U_k D) * C. \tag{D.1}$$

Here D represents the discretisation on the grid. The function D is a train of delta functions with a distance Δx between them. Further C is the convolution function that represents the interpolation method and * represents a convolution. So each interpolation method has a unique function C. We show the functions for linear interpolation in Fig. D.1.

In order to show that all $U_k - \widetilde{U}_k$ are orthogonal we take the Fourier transform denoted by $F[\cdot]$ of $\widetilde{U}_k$.

$$F[\widetilde{U}_k] = F[(U_k D) * C] = (F[U_k] * F[D])F[C]. \tag{D.2}$$

For linear interpolation these functions are shown by Fig. D.1. Because $F[U_k]$ is one delta function, a convolution with this function results in a shift. Further $F[D]$ is again a train of delta functions. Now they are separated by $(\Delta x)^{-1}$. Concluding, $F[U_k] * F[D]$ is still a train of delta functions. Multiplying it with $F[C]$ only changes the amplitude which means that $F[\widetilde{U}_k]$ still consists of a discrete set of Fourier components. Because they are separated by exactly $(\Delta x)^{-1}$, different $\widetilde{U}_k$ do not share any frequency components. This means that $\widetilde{U}_k$ is orthogonal. Because the frequency U_k is also in $\widetilde{U}_k$, the functions $U_k - \widetilde{U}_k$ are also orthogonal.

The extension to the three dimensional case is rather straightforward and is therefore not done here. The basic idea is to create the three dimensional functions out of a multiplication from one dimensional components.

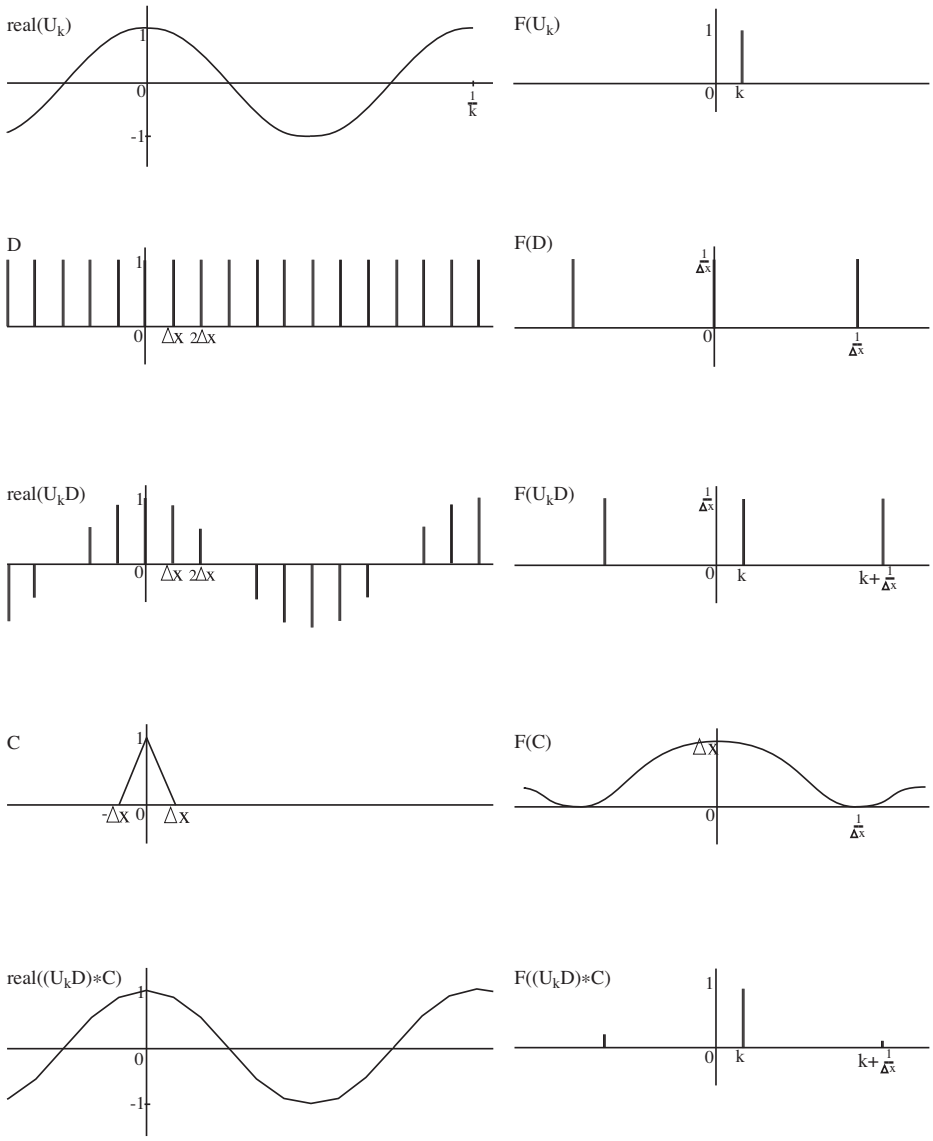

Figure D.1: Linear interpolation. The pins represent delta functions with the hight equal to the prefactor.

Appendix E

Proof of optimal polynomial function

In this section we show some special properties that hold for Hermite spline interpolations. Especially we examine the case where $N = 4$. For this case the second derivative becomes a piecewise linear function. Comparison with the actual second derivative shows that this piecewise linear function is optimal with respect to the L^2 norm.

We start by examining the first derivative. When looking at the interpolation one can see that N conditions are needed to make it $C^{N/2-1}$ and the order of the polynomial function is $N-1$. For the derivative the order of polynomial function decreases with one. Therefore one condition less is needed, the new conditions become:

$$\begin{aligned} \int_0^1 \frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}\mathrm{d}x &= \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\mathrm{d}x \\ \frac{\mathrm{d}^l\tilde{u}}{\mathrm{d}x^l}(0) &= \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(0) \\ \frac{\mathrm{d}^l\tilde{u}}{\mathrm{d}x^l}(1) &= \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(1) \\ l &= 1,..,m \end{aligned} \tag{E.1}$$

The first condition has a special property. To see this one needs to transform the polynomial basis into a orthogonal polynomial basis B with respect to the inner product $\langle\rangle_1$. This is done by Gram Schmidt orthogonalization. Now one gets

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x} = \sum_{j=0}^{N-2} \hat{a}_j^{(1)} B_j(x). \tag{E.2}$$

$$\begin{aligned} B_0 &= 1 \\ B_1 &= \sqrt{3}(2x-1) \\ B_2 &= \sqrt{5}(6x^2-6x+1) \\ B_3 &= \sqrt{7}(20x^3-30x^2+12x-1) \end{aligned} \tag{E.3}$$

Now one can show that $\hat{a}_0^{(1)}$ is chosen optimally with respect to the L^2 norm.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\hat{a}_0^{(1)}} \int_0^1 \left(\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}(x) - \frac{\mathrm{d}u}{\mathrm{d}x}(x)\right)^2 \mathrm{d}x = 0 \\
\int_0^1 2\left(\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}(x) - \frac{\mathrm{d}u}{\mathrm{d}x}(x)\right) \frac{\mathrm{d}}{\mathrm{d}\hat{a}_0^{(1)}} \frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}(x)\mathrm{d}x = 0 \\
\int_0^1 2\left(\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}(x) - \frac{\mathrm{d}u}{\mathrm{d}x}(x)\right) B_0 \mathrm{d}x = 0 \\
\int_0^1 \frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}\mathrm{d}x = \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\mathrm{d}x
\end{aligned} \tag{E.4}$$

Next we are going to look at the second derivative. Now we only consider the case of $N = 4$, the theory could be extended for more N but it is not done here. Because we are looking at the second derivative only two conditions are left, namely:

$$\begin{aligned}
\int_0^1 \frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}\mathrm{d}x = \int_0^1 \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\mathrm{d}x, \\
\int_0^1 \int_0^\alpha \frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}\mathrm{d}x\mathrm{d}\alpha = \int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\mathrm{d}x\mathrm{d}\alpha.
\end{aligned} \tag{E.5}$$

Now one can show that $\hat{a}_0^{(2)}$ is chosen optimally, in the same way as before. Further not only $\hat{a}_0^{(2)}$ is chosen optimal but also $\hat{a}_1^{(2)}$. This means that there is no better approximation in the L^2 norm of this second derivative with a linear function. The proof goes as follows:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\hat{a}_1^{(2)}} \int_0^1 \left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right)^2 \mathrm{d}x = 0 \\
\int_0^1 2\left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right) \frac{\mathrm{d}}{\mathrm{d}\hat{a}_1^{(2)}} \frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x)\mathrm{d}x = 0 \\
\int_0^1 \left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right)(2x-1)\mathrm{d}x = 0 \\
\int_0^1 \left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right) x\mathrm{d}x = 0 \\
\alpha \int_0^\alpha \left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right)\mathrm{d}x\Big|_{\alpha=0}^{\alpha=1} - \int_0^1 \int_0^\alpha \left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x)\right)\mathrm{d}x\mathrm{d}\alpha = 0 \\
\int_0^1 \int_0^\alpha \frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}\mathrm{d}x\mathrm{d}\alpha = \int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\mathrm{d}x\mathrm{d}\alpha.
\end{aligned} \tag{E.6}$$

In the derivation several times the first relation of equation (E.5) is used. Further integration by parts is used in the second last step.

So for the Hermite spline interpolation with $N = 4$ we have shown that the second derivative is still a good approximation of the actual second derivative. Even stronger, there is no better piecewise linear function for the approximation of this second derivative with respect to the L^2 norm.

Bibliography

[1] F. Toschi and E. Bodenschatz. Lagrangian properties of particles in turbulence. *Annu. Rev. Fluid Mech.*, **41**:375–404, 2009.

[2] M. van Aartrijk. *Dispersion of inertial particles in stratified turbulence*. PhD thesis, Eindhoven University of Technology, 2008.

[3] M. van Aartrijk, H.J.H Clercx, and K.B. Winters. Single-particle, particle-pair, and multiparticle dispersion of fluid particles in forced stably stratified turbulence. *Phys. Fluids*, **20**:025104 1–16, 2008.

[4] M.R. Maxey and J.J. Riley. Equation of motion for a small rigid sphere in a nonuniform flow. *Phys. Fluids*, **26**:883–889, 1983.

[5] M.A.T. van Hinsberg, J.H.M. ten thije Boonkkamp, and H.J.H Clercx. An efficient, second order method for the approximation of the Basset history force. *J. Comput. Phys.*, 2011.

[6] R. Mei, R.J. Adrian, and T.J. Hanratty. Particle dispersion in isotropic turbulence under Stokes drag and Basset force with gravitational settling. *J. Fluid Mech.*, **225**:481–495, 1991.

[7] V. Armenio and V. Fiorotto. The importance of the forces acting on particles in turbulent flows. *Phys. Fluids*, **13**:2437–2440, 2001.

[8] M. van Aartrijk and H.J.H Clercx. Vertical inertial particle dispersion in stably stratified turbulence: The influence of the Basset force. *Phys. Fluids*, **22**:013301 1–9, 2010.

[9] M. van Aartrijk and H.J.H Clercx. The dynamics of small inertial particles in weakly stratified turbulence. *J. Hydro-Envir. Res.*, **4**:103–114, 2010.

[10] Y.D. Sobral, T.F. Oliveira, and F.R. Cunha. On the unsteady forces during the motion of a sedimenting particle. *Powder Technol.*, **178**:129–141, 2007.

[11] I. Niño and M. García. Using Lagrangian particle saltation observations for bedload sediment transport modelling. *Hydrolog. Process*, **12**:1197–1218, 1998.

[12] N. Mordant and J.F. Pinton. Velocity measurement of a settling sphere. *Eur. Phys. J. B*, **18**:343–352, 2000.

[13] D.J. Vojir and E.E. Michaelidis. Effect of the history term on the motion of rigid spheres in a viscous fluid. *Int. J. Multiphase Flow*, **20**:547–556, 1994.

[14] P. Tanga and A. Provenzale. Dynamics of advected tracers with varying buoyancy. *Physica D*, **76**:202–215, 1994.

[15] F.A. Bombardelli, A.E. González, and Y.I. Niño. Computation of the Particle Basset Force with a Fractional-Derivative Approach. *J. Hydr. Div.*, **134**:1513–1520, 2008.

[16] A.J. Dorgan and E. Loth. Efficient calculation of the history force at finite Reynolds numbers. *Int. J. Multiphase Flow*, **33**:833–848, 2007.

[17] E.E. Michaelides. A novel way of computing the Basset term in unsteady multiphase flow computations. *Phys. Fluids A*, **4(7)**:1579–1582, 1992.

[18] F. Lekien and J. Marsden. Tricubic interpolation in three dimensions. *Int. J. Numer. Meth. Engng*, **63**:455–471, 2005.

[19] C.C. Lalescu, B.Teaca, and D.Carati. Implementation of high order spline interpolations for tracking test particles in discretized fields. *J. Comput. Phys.*, **229**:5862–5869, 2010.

[20] E. Calzavarini, R. Volk, E. Lévêque, J.-F. Pinton, and F. Toschi. Impact of trailing wake drag on the statistical properties and dynamics of finite-sized particle in turbulence. *ArXiv:1008.2888v1*, 2010.

[21] H. Homann and J. Bec. Finite-size effects in the dynamics of neutrally buoyant particles in turbulent flow. *J. Fluid. Mech*, **651**:81–91, 2010.

[22] E. Loth. Numerical approaches for motion of dispersed particles, droplets and bubbles. *Prog. Energ. Combust.*, **26**:161–223, 2000.

[23] E.E. Michaelides. Hydrodynamic Force and Heat/Mass Transfer From Particles, Bubbles, and DropsThe Freeman Scholar Lecture. *J. Fluids Eng.*, **125(2)**:209–238, 2003.

[24] H. Faxén. Die Bewegung einer starren Kugel längs der Achse eines mit zäher Flüssigkeit gefüllten Rohres. *Ark. Mat. Astron. Fys.*, **17**:1–28, 1923.

[25] T.R. Auton, J.C.R. Hunt, and M. Prud'homme. The force exerted on a body in inviscid unsteady non-uniform rotational flow. *J. Fluid Mech.*, **197**:241–257, 1988.

[26] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes*. Cambridge University Press, New York, 1992.

[27] F.B. Tatom. The Basset force as a semiderivative. *Appl. Sci. Res.*, **45**:283–285, 1988.

[28] A. Jeffrey. *Handbook of mathematical formulas and integrals*, pages 245–248. Elsevier, United Kingdom, 2004.

[29] T.G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *Siam Review*, **51(3)**:455–500, 2009.