Elsevier Editorial System(tm) for Journal Of Computational Physics

Manuscript Draft

Manuscript Number:

Title: Approximating the Basset force by optimizing the method of van Hinsberg et al.

Article Type: Regular Article

Keywords: Basset, history force, window method, particle-laden, numerical

Corresponding Author: Mr. Guillermo Casas, M.D.

Corresponding Author's Institution: Barcelona Tech

First Author: Guillermo Casas

Order of Authors: Guillermo Casas; Alex Ferrer; Eugenio Oñate

Abstract: In this work we put the method proposed by van Hinsberg et al. to the test, highlighting its accuracy and efficiency in a sequence of benchmarks of increasing complexity. Furthermore, we explore the possibility of systematizing the way in which the method's free parameters are determined by generalizing the optimization problem that was considered originally. Finally, we provide a list of worked-out values, ready for implementation in large-scale particle-laden flow simulations.

Suggested Reviewers: M. A. T. van Hinsberg
M.A.T.v.Hinsberg@tue.nl
Main author cited. He introduced of the method MAE.

Ksenia Guseva
ksenia.guseva@uni-oldenburg.de
Cited in the paper. Worked on the importance of history force in particle-laden flows. Recent reports of the computational difficulties related to memory requirements.

Patricio Moreno-Casas
patriciomoreno@miuandes.cl
First author in very recent review on methods for the calculation of the history force. Reviewed both methods mainly used in the paper.

Significance and novelty of this paper

The paper addresses the problem of including the Basset history force in particle-laden flow simulations, which is known to be extremely computationally expensive. However, its importance has been stressed in several publications.

We consider the method of approximation by exponentials (MAE) presented by van Hinsberg et al. to approximate the Basset history force in particle-laden flow simulations at low particle Reynolds numbers. This method introduces a set of free parameters that must be determined by optimizing an error bound. However, the original work considered only half of the parameters in the optimization problem, leaving the remaining parameters to be determined by an heuristic argument. We consider the full set of parameters in an extended optimization problem than turns out to be considerably more challenging. A full worked-out list of parameters is provided in an appendix, ready for implementation.

Furthermore, for the first time the second-order-accurate extension of the method is implemented and throughly tested, showing its remarkable accuracy and efficiency. Previous implementation of high-order schemes (larger than one) had not been implemented together with a window method, such as the MAE.

***Research Highlights**

- We generalize the optimization problem that arises in the method of approximation by exponentials of van Hinsberg et al., whose solution determines the free parameters in this method.
- A worked out list of parameters is provided, ready for implementation.
- We provide a complete time-integration algorithm, extending it to second order by coupling it with state-of-the-art quadrature schemes in the window region.
- We perform extensive testing to show that the method is accurate and can be used in practical particle-laden flow simulations.

Approximating the Basset force by optimizing the method of van Hinsberg et al.

G. Casas[a,*], A. Ferrer[a,b], E. Oñate[a]

[a] *CIMNE Centre Internacional de Mètodes Numèrics en Enginyeria, Campus Nord UPC, Edifici C-1, c/Jordi Girona 1-3, 08034 Barcelona*
[b] *Escola Superior dEnginyeries Industrial, Aeroespacial i Audiovisual de Terrassa, Campus de Terrassa, Edifici TR45. C. Colom, 11 08222 Terrassa, Spain*

Abstract

In this work we put the method proposed by van Hinsberg et al. [12] to the test, highlighting its accuracy and efficiency in a sequence of benchmarks of increasing complexity. Furthermore, we explore the possibility of systematizing the way in which the method's free parameters are determined by generalizing the optimization problem that was considered originally. Finally, we provide a list of worked-out values, ready for implementation in large-scale particle-laden flow simulations.

Keywords: Basset, history force, window method, particle-laden, numerical
2010 MSC: 00-01, 99-00

1. Introduction

The equation of motion of an isolated, small, spherical particle submerged in a Newtonian fluid is well described by the equation proposed by Maxey and Riley [20], often referred to as the Maxey–Riley Equation, or, MRE. It reads

$$\begin{aligned} m_p \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = m_f \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} + \frac{1}{2} m_f \left(\frac{\mathrm{D}}{\mathrm{D}t}\left(\mathbf{u} + \frac{1}{10}a^2\nabla^2\mathbf{u}\right) - \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}\right) + C_D\left(\mathbf{u} - \mathbf{v} + \frac{1}{6}a^2\nabla^2\mathbf{u}\right) \\ + C_B \frac{\mathrm{d}}{\mathrm{d}t}\int_{t_0}^{t} \frac{1}{\sqrt{t-s}}\left(\mathbf{u} - \mathbf{v} + \frac{1}{6}a^2\nabla^2\mathbf{u}\right)\,\mathrm{d}s + (m_f - m_p)\mathbf{g} \end{aligned} \tag{1}$$

where $C_D = 6\pi a\mu$, $C_B = 6a^2\sqrt{\pi\rho_f\mu}$, m_p is the mass of the particle, m_f is the mass of the displaced fluid volume, a is the particle radius, ρ_f and μ are the density and dynamic viscosity of the fluid; and $\boldsymbol{g}$ is the acceleration due to gravity. The vector $\boldsymbol{v}$ is the velocity of the (point-)particle and $\boldsymbol{u}$ that of the surrounding fluid field, evaluated at the particle's center. $\mathrm{D}/\mathrm{D}t$ denotes the material derivative of the fluid. Equation (1), together with $\boldsymbol{v} = \mathrm{d}\boldsymbol{r}/\mathrm{d}t$ and the initial conditions $\boldsymbol{r}(t_0) = \boldsymbol{r}_0$ and $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$ form an initial value problem that must be solved to obtain the trajectory of the particle (where $\boldsymbol{r}$ is the particle's position vector).

The different terms on the right hand side of (1) have distinct physical interpretations and can be identified as: (from left to right) the force applied to the volume displaced by the particle in the undisturbed flow ($\boldsymbol{F}_U$), the added mass or virtual mass force ($\boldsymbol{F}_A$) [1], the Stokes drag ($\boldsymbol{F}_D$), the Basset–Boussinesq history force ($\boldsymbol{F}_H$) and the force due to the weight of the particle minus its buoyancy ($\boldsymbol{F}_W$).

Equation (1) can be used to simulate the dynamics of a number of particles suspended in fluid, provided that the particles are small enough, so that the flow around each of them can be accurately modelled by the

*Corresponding author
Email addresses: gcasas@cimne.upc.edu (G. Casas), aferrer@cimne.upc.edu (A. Ferrer), onate@cimne.upc.edu (E. Oñate)

[1] The form of the added mass force used here, with the material derivative applied to the field $\boldsymbol{u}$ corresponds to the more commonly applied form of this term, derived by Auton et al. [1] for inviscid flow. This form turns out to be also accurate for low and intermediate Reynolds numbers, as has been shown in several studies; see [21, 18, 29]. However, the difference with its alternative, involving $\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}$, is negligible in the range of validity of the MRE; see [20].

instationary Stokes equations; and that they spend most of the time far enough from each other to justify applying this single-particle theory [20]. Under these conditions, (1) accurately describes how the motion of the submerged particles is determined by the background flow field, $\boldsymbol{u}$.

Typically, $\boldsymbol{u}$ is calculated with a suitable numerical method, such as the finite element method, on a static mesh. Hence the relevant flow variables need to be somehow interpolated in order to have $\boldsymbol{u}$ and its needed derivatives defined exactly at the particles' centres. The motion of each particle can then be calculated by solving the MRE numerically, in a stepwise fashion, alternating advances in time with updates of the fluid flow at the particles' locations. A detailed review of this kind of methods, among others, is given by Loth [15]. A discussion of alternative interpolation methods can be found in [13]. Some relevant examples of applications of these simulation techniques are the study of turbulent dispersion of suspended particles, [26]; contaminants convection in cracks, [24]; liquid crystal growth, [5]; and transport in biological vessels, [23].

In this work, we focus on the solution of the MRE itself, avoiding any discussion about either the fluid calculation or the interpolation process. Instead, we make use of analytical fluid fields, which we directly impose at the particle locations. This simplifies the arguments, eliminating any influences from interpolation errors and from errors in the numerical solution of the Navier Stokes Equations.

Several studies have recently stressed its importance, particularly in liquid particulate flows [27, 7]. However, the inclusion of the history term in the numerical implementation of the Maxey–Riley equation is still widely regarded as impractical due to the large memory requirements and the corresponding overheads involved in the calculation of the associated integral. Indeed, the following two issues arise:

A The integral is defined over the totality of the particles' past trajectories. It is therefore necessary to keep track of an ever-increasing number of historical flow values (per particle) to perform the quadrature. Such number grows in proportion to the simulated duration.

B Standard quadrature methods perform poorly due to the singularity at the upper integration limit, requiring a large number of quadrature points per time unit to sufficiently reduce the quadrature error.

An attempt to alleviate the severity of A was put forward by Dorgan and Loth [8], with the so-called *window method*. This approach takes advantage of the decreasing influence of past flow conditions on the advancing present to avoid having to store the complete history of the particles. Unfortunately, the decay associated with the Basset kernel(the $1/\sqrt{t-s}$ prefactor of the integrand) is too slow at the very low particle Reynolds numbers [2] that concern us, going as the inverse of the square root of time [16]. This implies that the number of points to be recorded still has to be large, and in fact radically limiting the performance of the method if accuracy is to be preserved [9].

The recent developments by Daitche [6] and van Hinsberg et al. [12] have certainly improved the situation. The first of these works addressed B through the construction of higher order schemes. The other work addressed A with a kind of window model that approximates, rather than neglects, the tail contribution to the Basset integral. In this work, we build upon these two methods, combining them and further adding to the result. The focus is placed on the optimization problem that leads to the determination of the free parameters of the model of van Hinsberg et al. We analyse the performance of the resulting algorithm in a succession of steps, each adding a layer of complexity toward real-world applications. Consistently, we show its remarkable efficiency and accuracy. We are also concerned with the validity ranges and robustness of the method, about which we draw some generic recommendations. Finally, a worked out list of optimal parameters is included (Appendix E), ready to be used in numerical implementations for particle-laden flow simulations.

2. Window models and the Hinsberg method

2.1. Preliminaries

Once discretized in time, the Basset–Boussinesq history term, $\boldsymbol{F}_H$ will have been replaced by its finite difference counterpart. That is, a linear combination of the integral itself evaluated at different times.

[2] The particle Reynolds number is defined as $Re_p = aw/\nu$, where w is the modulus of the slip velocity. The MRE is derived under the assumption that $Re_p \ll 1$.

Therefore, the problem of how to evaluate such term becomes that of finding a way to calculate the integral itself, rather than its derivative, at any given point in time. Let us start by recalling the exact form of the integral:

$$\int_{t_0}^{t} \frac{1}{\sqrt{t-\tau}} \left(\mathbf{u} - \mathbf{v} + \frac{1}{6} a^2 \nabla^2 \mathbf{u} \right) \mathrm{d}\tau \tag{2}$$

which can be rewritten as

$$\mathbf{I}_B(t) \coloneqq \int_{t_0}^{t} K_B(t-\tau) \mathbf{f}(\tau) \, \mathrm{d}\tau \tag{3}$$

where $K_B(x) \coloneqq 1/\sqrt{x}$ defines the Basset kernel function and $\mathbf{f}(x)$ a differentiable vector field (assuming the unknowns to be smooth enough). Thus, the integral above can be understood as the convolution of $\mathbf{f}$ with the kernel K_B. Note that, with this notation, we have $\boldsymbol{F}_H = C_B \mathrm{d}\mathbf{I}_B(t)/\mathrm{d}t$. It is also possible to consider a different form of the Basset integral, where the derivative moves under the integral sign; see Appendix A. Specifically

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{I}_B(t) = \int_{t_0}^{t} K_B(t-\tau) \frac{\mathrm{d}\mathbf{f}(\tau)}{\mathrm{d}\tau} \, \mathrm{d}\tau + \mathrm{K_B(t-t_0)\mathbf{f}(t_0)} \quad \forall \mathrm{t} > \mathrm{t_0} \tag{4}$$

where the alternative form is on the right-hand side. The latter was precisely the one used in [12], though its second term vanished because the initial time was taken as $t_0 = -\infty$. Nonetheless, the same method can be applied to both situations; it is only necessary to correctly identify the field that must be considered: either $\mathbf{f}$, or its derivative [3]. As pointed out by [6], the first form is very convenient because it eliminates the need to compute either any additional derivatives or the second term in (4). It is for this reason that we have also adopted this form in this work.

In this section we present our algorithm for the integration of the MRE, describing it in detail. We start with the hybrid polynomial interpolation/analytic approach, which we use for the computation of the integral over the more recent history. Then, we introduce the method of van Hinsberg et al., which defines how the integral over the remaining history is calculated. Finally, we explain how both approaches are stitched together and implemented in the time integration scheme of the full MRE.

2.2. Hybrid polynomial interpolation/analytic approach

The idea of using a polynomial interpolation to approximate *only* the nonsingular part of the integrand ($\mathbf{f}$ in Equation (3)) was introduced in [12], who used linear polynomials and obtained a first-order accurate method. That is, they showed the quadrature error to scale with h^2, where h is the distance between successive quadrature points. This performance beat standard methods (for example, the second order Euler-McLaurin formula, of order one half) but also the fractional derivative approach used by Bombardelli et al. [2], which is first-order-accurate. The method consists in the following steps:

1. Subdivide the integration domain, $[t_0, t]$ in n subintervals $[t_0, t_0 + h], \ldots, [t_{n-1}, t_n]$, with $t = t_n$. The boundaries of these intervals are defined by all the scheme's intermediate time integration points up to the present time, t.

2. Interpolate $\mathbf{f}$ in the interior of each interval with a linear polynomial.

3. Replace $\mathbf{f}$ by its interpolated approximation back into the integral and find its primitive (it has a simple analytical formula).

4. Reorder the resulting expression as a weighed sum of the values $\mathbf{f}_0 \coloneqq \mathbf{f}(t_0), \ldots, \mathbf{f}_n \coloneqq \mathbf{f}(t_n)$. The result is the discretized version of $\mathbf{I}_B$.

5. Plug the expression into a suitable time integration scheme for the MRE. The resulting algorithm expresses the current force as a function of the unknowns at all the past (and current) time steps.

[3] Such derivative would involve a finite difference representation of the unknowns and the fluid field values under the integral sign, leading to a stencil involving the linear combination of quadratures at different points in time.

The method was recently extended to second and third order accuracy by Daitche [6], who employed a type of semi-implicit Adams-Bashforth scheme for the time-integration and provided a detailed analysis of its performance. We will thus hereafter to it as the Daitche method. Its second-order-accurate variant has been our specific choice in this work.

2.3. Addressing memory requirements: window methods

Even though Daitche's method achieves considerable improvements in accuracy, therefore allowing for larger time steps, its memory requirements are still large. In systems with many particles, the necessity of keeping track of the complete history of each of them becomes highly demanding. Indeed, let us for instance consider a system with 10^5 particles. Assuming we take a time step of 0.01s and a simulation duration of 10s (as done in [6]), the necessity of storing one vector per time step and particle leads to a total 10^8 vectors or, roughly, about 1 GB in memory (assuming a vector takes up 3×8 bytes) by the end of the simulation!. As mentioned in the introduction, the influence of past values decays as time passes, and window methods are design to take advantage of this fact to avoid having to store the complete history of the particles.

The simplest approach consists in neglecting the contribution of the particle's history that is older than some cut-off time $t - t_w$, where t is the current time. One refers to t_w as the *window time*, because only the history within $[t - t_w, t]$ is taken into account. That is, one considers

$$\int_{t_0}^{t} K_B(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \approx \int_{t-t_w}^{t} K_B(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \tag{5}$$

which is equivalent to

$$\int_{t_0}^{t-t_w} K_B(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \approx 0 \tag{6}$$

The integral on the right-hand side of (5) becomes the *window contribution*, and the integral one on left-hand side of (6) the *tail contribution*. The idea is to employ a suitable method, such as the Datiche method, to calculate the window contribution and to neglect the rest, thereby saving all the associated memory space (and avoiding the corresponding computations). As such, this method corresponds to what is known as the window method in literature, following its introduction by Dorgan and Loth [8]. These researchers applied it to the case of finite particle Reynolds numbers, taking advantage of the faster decaying nature of the history force kernel in this case [4], see for instance [16]. Here we apply the term *window method* generically to any method that records only a limited part of the particles' most recent history. The main weakness of the original window method is its poor accuracy, precisely due to the slow decay of the convolution kernel in time (as the inverse of the square root for the Basset kernel); because one is thus forced to increase t_w significantly to keep the tail truncation error low, at the cost of sacrificing potential memory savings.

A much more accurate window method was proposed by van Hinsberg et al. [12] (henceforth referred to as MAE for *Method of Approximation by Exponentials*). In it, the tail contribution is not neglected but rather *approximated*. Specifically, while in the window region the kernel is kept exactly equal to the Basset kernel, in the tail region it is only approximated by a different, special kernel. The advantage of this special kernel is that it leads to a recursive expression of the evolution of the Basset integral in time. That is, the evolution of the integral from one time step to the next can be expressed in terms of its value in the previous time step plus a contribution that depends only on the recent history. Despite its relatively recent development, the method has already found application in the study of particle-laden turbulence; see [4], [14] and [17].

2.4. The MAE: methodology

Let us review the method in sufficient detail. We begin by replacing the Basset kernel $K_B : (0, t-t_0] \to \mathbb{R}$ with the modified kernel $K : (0, t-t_0] \to \mathbb{R}$, defined by

$$K(\tau) = \begin{cases} K_B(\tau) & \text{if } \tau \le t_w \\ K_T(\tau) & \text{if } \tau > t_w \end{cases} \tag{7}$$

[4] The exact formulation includes an algorithm to approximately determine t_w so as to minimize the truncation error. But since we would like to have t_w as small as possible, we cannot in general apply it without running into a conflict

where $K_B(\tau) = 1/\sqrt{\tau}$, as before, and the tail kernel is defined by

$$K_T(\tau) = \sum_{i=1}^{m} a_i K_i(\tau) \tag{8}$$

with

$$K_i(\tau) = \alpha(t_i)\mathrm{e}^{\beta(t_i)\tau} \quad i = 1, \ldots, m \tag{9}$$

and

$$\alpha(t_i) = \sqrt{\frac{\mathrm{e}}{t_i}}, \quad \beta(t_i) = -\frac{1}{2t_i} \tag{10}$$

Which introduces $2m$ free parametric constants, $a_1, \ldots, a_m$ and $t_1, \ldots, t_m$. In other words: we replace the Basset kernel by a linear combination of exponentials at the tail. As for the form of the functions, van Hinsberg et al. reason as follows: Let us consider the unknown parameters $t_1, \ldots, t_m$ to be a set of points that belong to the interval $(0, \infty)$. Let us then impose the condition that the graph of each $K_1, \ldots, K_m$ be tangent to that of K_B at the these points $t_1, \ldots, t_m$. This condition looks to align the exponentials with the reference curve, so that a much smaller set of good candidate exponential approximants is considered. Precisely setting α and β as in (10) fulfils this purpose. The hope is that this choice will later facilitate the determination of the sets of parameters a_i and t_i.

Now, instead of (5), one has

$$\int_{t_0}^{t} K_B(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \approx \int_{t-t_w}^{t} K_B(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau + \int_{t_0}^{t-t_w} K(t-t_w)\mathbf{f}(\tau)\,\mathrm{d}\tau = \mathbf{F}_w(t) + \mathbf{F}_t(t) \coloneqq \mathbf{I} \tag{11}$$

where we assume $t_w \geq t_0$. Thus, the whole integral becomes the sum two terms: the *window contribution*, $\mathbf{F}_w(t)$, and the *tail contribution*, $\mathbf{F}_t(t)$. The former can be calculated, for example, with the polynomial interpolation/analytic approach. In particular a first-order accurate version of this method was used in [12]. On the other hand, $\mathbf{F}_t(t)$ is calculated as explained below.

The use of exponentials to approximate the tail of the kernel becomes the key that permits a recursive calculation of the tail integral, which leads to a radical decrease in memory requirements. It is not difficult to prove the following relations [12]

$$\begin{aligned} \mathbf{F}_t(t) &= \sum_{i=1}^{m} a_i \mathbf{F}_i(t) \\ \mathbf{F}_i(t) &= \mathbf{F}_{d,i}(t) + \mathbf{F}_{r,i}(t) \end{aligned} \tag{12}$$

where

$$\begin{aligned} \mathbf{F}_{d,i}(t) &= \int_{t-h-t_w}^{t-t_w} K_i(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \\ \mathbf{F}_{r,i}(t) &= \mathrm{e}^{\beta(t_i)h}\mathbf{F}_i(t-h) \end{aligned} \tag{13}$$

and where a suitable temporal discretization of $\mathbf{F}_{d,i}$ must be provided. Note how only recent values of the integrand are involved in the calculation of $\mathbf{F}_i(t)$; all the information regarding older values is extracted, in (13), *by referring to the old value of the integral itself.*

Of course, it is still left to define parameters a_i and t_i. Let us for this purpose consider the error associated with the approximation of K_B by K, which we would like to have as small as possible:

$$\left|\mathbf{F}_H(t) - \mathbf{F}_{H,K}(t)\right| = C_B \left| \int_{t_0}^{t} (K_B - K)(t-\tau)\frac{\mathrm{d}\mathbf{f}(\tau)}{\mathrm{d}\tau}\,\mathrm{d}\tau + (\mathrm{K_B} - \mathrm{K})(\mathrm{t} - \mathrm{t_0})\mathbf{f}(\mathrm{t_0}) \right| \tag{14}$$

where $\mathbf{F}_{H,K}$ is the approximate history force, obtained using $\mathbf{I}$ in place of $\mathbf{I}_B$ and $|\cdot|$ denotes the usual vector modulus. Note that we have employed the RHS form of Equation (4) also for the kernel K. This is possible,

given the regularity of K, see Appendix A. Furthermore, it is straightforward to generalize the bound given in [12], see Appendix B:

$$\left|\mathbf{F}_H(t) - \mathbf{F}_{H,K}(t)\right| \leq C_B \frac{\|\mathbf{f}\|_\infty}{\sqrt{t_w}} \left(\left| K_B(1) - \tilde{K}(1) \right| + \int_1^{\tilde{t}_0} \left| \frac{\mathrm{d}}{\mathrm{d}\tilde{t}} \left(K_B(\tau) - \tilde{K}(\tau) \right) \right| \mathrm{d}\tau \right) \tag{15}$$

where $\tilde{K}$ is obtained from K by replacing the t_i with their rescaled analogues, $\tilde{t}_i = t_i/t_w$ and $\tilde{t}_0 = (t - t_0)/t_w$ and $\|\cdot\|_\infty$ is defined by

$$\|\mathbf{f}\|_\infty = \inf \left\{ C : |\mathbf{f}| \leq C \; a.e. \right\} \tag{16}$$

In fact, these authors considered only the limit $t_0 \to -\infty$, at which the first term in the parenthesis in (15) vanishes. Their interest was therefore to find a good approximation for long simulation times. Indeed, by taking this limit in (15), we recover the bound derived by [12]:

$$\left|\mathbf{F}_H(t) - \mathbf{F}_{H,K}(t)\right| \leq C_B \frac{\|\mathbf{f}\|_\infty}{\sqrt{t_w}} \left(\left| 1 - \tilde{K}(1) \right| + \int_1^{\infty} \left| \frac{\mathrm{d}}{\mathrm{d}\tau} \left(K_B(\tau) - \tilde{K}(\tau) \right) \mathrm{d}\tau \right| \right) \tag{17}$$

Now, an important observation to be made at this point is that the bound in (17) is also a bound for all values $t_0 > -\infty$. This means it is legitimate to take it as a reference for real, finite-time simulations. Like Hinsberg et al. we now concentrate in this bound, with the idea of minimizing it.

Hinsberg et al. point out that the minimization of the quantity between parenthesis on the right hand side of the inequality (16) is not amenable to the standard Newton–Raphson algorithm. They propose the following substitute objective function, which is differentiable with respect to the unknown parameters a_i and t_i:

$$\left(1 - \tilde{K}(1)\right)^2 + \int_1^{\infty} \tau \left(\frac{\mathrm{d}}{\mathrm{d}\tau} \left(K_B(\tau) - \tilde{K}(\tau) \right) \right)^2 \mathrm{d}\tau \tag{18}$$

The first step in the optimization procedure suggested by the same researchers is to "make a reasonable choice for $\tilde{t}_i$"; and then to calculate the optimal a_i parameters by applying Newton–Raphson's algorithm. About the choice of the $\tilde{t}_i$, the only indications are to make the choice of $\tilde{t}_i$ so as to they cover a great range, making the separations smaller as the values become smaller. A single example of $\tilde{t}_i$ values was provided by [12], specifically, the 10-member set $\{0.1, 0.3, 1, 3, 40, 190, 1000, 6500, 50000\}$, for which the corresponding Newton–Raphson-averaged values a_i were also provided.

The method has proved remarkably effective in reducing the computational cost of the simulations; see [12], [22] as originally formulated. Nonetheless, it still leaves room for improvement. Specifically, the following issues are not covered:

a) the possibility of having a nonzero initial relative velocity, which is not taken into account

b) the unexplored possibility of using a different time step for the quadrature

c) the choice of the t_i, which is largely heuristic

Point a) we have briefly touched upon, showing that the method's bound is applicable in this case. In any case, we do not pursue here an adaptable method which should include the dependence in time of the kernel's free parameters. Point b) is altogether left for future work. Point c), however, is the main concern of this paper, and, in particular, the subject of Section 3.

2.5. Full algorithm

Let us finally detail the specifics of the numerical implementation. We have already mentioned that the full algorithm is obtained by combining the hybrid polynomial interpolation/analytic approach for the integration of the window term with the MAE to approximate the tail contribution. The first method is accurate, requiring fewer time-steps per unit simulated time; the second avoids having to store most of the particle's history, leading to important memory savings. The algorithm closely resembles the one by [6], to which we have applied the required modifications. Let us start by rewriting the MRE as

$$(m_p + \frac{1}{2} m_f) \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \boldsymbol{F}_{NH} + \boldsymbol{F}_H \tag{19}$$

where $\boldsymbol{F}_{NH}$ stands for the 'non-memory' forces, that is

$$\boldsymbol{F}_{NH} = \boldsymbol{F}_U^* + \boldsymbol{F}_D + \boldsymbol{F}_H + \boldsymbol{F}_W \tag{20}$$

and where now

$$\boldsymbol{F}_U^* = \boldsymbol{F}_U + \frac{1}{2} m_f \left(\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} + \frac{1}{10} a^2 \frac{\mathrm{d}}{\mathrm{d}t} \left(\nabla^2 \boldsymbol{u} \right) \right) \tag{21}$$

The added mass force $\boldsymbol{F}_A$ has thus been split in two: one part has been moved to the LHS of the MRE and the rest has been absorbed, along with $\boldsymbol{F}_U$, into $\boldsymbol{F}_U^*$, which now holds the non-historic contribution from the fluid acceleration. By integrating both sides of (19) over $(t, t+h)$, we obtain

$$\boldsymbol{v}(t+h) = \boldsymbol{v}(t) + \frac{1}{m_p + \dfrac{m_f}{2}} \left(\boldsymbol{H}(t+h) - \boldsymbol{H}(t) + \int_t^{t+h} \boldsymbol{F}_{NH} \, \mathrm{d}\tau \right) \tag{22}$$

where

$$\boldsymbol{H}(s) \coloneqq C_B \int_{t_0}^{t} \frac{1}{\sqrt{t-\tau}} \left(\boldsymbol{u} - \boldsymbol{v} + \frac{1}{6} a^2 \nabla^2 \boldsymbol{u} \right) \mathrm{d}\tau \tag{23}$$

The discretization of the integral term in (22) can be done for a variety of finite difference schemes. Here we have chosen the Adams–Bashforth Formulas, as in [6]. On the other hand, the $\boldsymbol{H}$-terms are partitioned according to the MAE, see (11), as

$$\boldsymbol{H}(s) = \boldsymbol{H}_w(s) + \boldsymbol{H}_t(s) \tag{24}$$

and where $\boldsymbol{H}_w(s)$ is calculated with the Daitche method, while $\boldsymbol{H}_t(s)$ is calculated according to (11), (13) and (12), see Section 2.4. The algorithmic details concerning the latter term can be found in the Appendix B. Furthermore, and similarly to what has been done with $\boldsymbol{F}_A$, the term $\boldsymbol{H}_{t,n+1}$, which also contains a term proportional to $\boldsymbol{v}_{n+1}$, is split and the latter is sent to the LHS of the equation, leaving the modified term $\boldsymbol{H}^*_{t,n+1}$ on the RHS. By doing so, the equations become *semi-implicit*. In the work of [12] it was found that the accuracy and stability of the resulting algorithm improved greatly, avoiding the need for extremely small time-steps. Our preliminary calculations indeed confirmed the same tendency. Denoting the time stepping index as k, the resulting first-order scheme (used in the initialization) reads

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k + h\boldsymbol{v}_k + \mathcal{O}(h^2) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \frac{h}{m_p + \frac{1}{2} m_f + 6a^2 \sqrt{\pi \rho \mu \phi h}\, \alpha_0^n} \left(\boldsymbol{F}_{NH,k} + \boldsymbol{H}^*_{t,\alpha,k+1} - \boldsymbol{H}_{\alpha,k} \right) + \mathcal{O}(h^2) \end{aligned} \tag{25}$$

while the second order version (used elsewhere) is

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k + \frac{h}{2} \left(3\boldsymbol{v}_k - \boldsymbol{v}_{k-1} \right) + \mathcal{O}(h^3) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \frac{h}{m_p + \frac{1}{2} m_f + 6a^2 \sqrt{\pi \rho \mu \phi h}\, \beta_0^n} \left(\frac{3}{2} \boldsymbol{F}_{NH,k} - \frac{1}{2} \boldsymbol{F}_{NH,k-1} + \boldsymbol{H}^*_{t,\beta,k+1} - \boldsymbol{H}_{\beta,k} \right) + \mathcal{O}(h^3) \end{aligned} \tag{26}$$

where the subscripts indicate the time-step at which each term must be evaluated and where and extra index (α or β) has been added to the $\boldsymbol{H}_t^*$ terms to indicate which formula (first or second order, see Appendix C) must be used in each case.

3. Optimization problem: determining the a_i and the t_i

In this section we generalize the optimization problem considered in [12] in order to fix the free parameters a_i and t_i. First, we pose the problem in a mathematically sound setting. We then present several alternative options to circumvent some of the difficulties brought about by the original formulation. Next, we explore the behaviour of the different options, which show significant differences in behaviour. Finally, we present the results for the different alternatives. The best ones (listed in Appendix E) are used in the subsequent chapters.

3.1. Posing the optimization problem

Let us start by making explicit the dependence of the functions involved. Re-expressing the modified kernel (7) in terms of a_i and t_i, we obtain

$$\tilde{K}_T(\tilde{\tau}, a_i, \tilde{t}_i) = \sum \frac{a_i}{\sqrt{\tilde{t}_i}} e^{\frac{1}{2}(1-\frac{\tilde{\tau}}{\tilde{t}_i})}. \tag{27}$$

and the Basset kernel is

$$K_B(\tilde{\tau}) = \frac{1}{\sqrt{\tilde{\tau}}} \tag{28}$$

For the sake of a compact notation, we define the kernel approximation error as the following function:

$$e_K(\tilde{\tau}, a_i, \tilde{t}_i) = K_B(\tilde{\tau}) - \tilde{K}_T(\tilde{\tau}, a_i, \tilde{t}_i). \tag{29}$$

We would like to minimize the error in the calculation of the history force, $\mathbf{F}_B$. However the force depends on the unknown relative flow and thus we must settle for minimizing its bound, given by Equation (17). This is equivalent to minimizing

$$I_1(a_i, \tilde{t}_i) = \left| e_K(1, a_i, \tilde{t}_i) \right| + \int_1^\infty \left| \frac{\partial}{\partial \tilde{\tau}} e_K(\tilde{\tau}, a_i, \tilde{t}_i) \right| \, \mathrm{d}\tilde{\tau}. \tag{30}$$

Anticipating the numerical difficulties associated with this cost function, we set up the minimization problem leaving the cost function unspecified so that alternative cost functions can be later considered as follows

$$\begin{array}{ll} \underset{a_i, \tilde{t}_i}{\text{minimize}} & I(a_i, \tilde{t}_i) \\ \text{subjected to:} & 0 \leq a_i \leq 1 \\ & 0 \leq \tilde{t}_i \end{array} \tag{31}$$

where i takes integer values from 1 to m and $I(a_i, \tilde{t}_i)$ stands for a general objective function. For example, one of the alternatives is to consider $I \equiv I_1$.

Let us discuss this set-up in some detail. Regarding the design variables a_i, the zero lower bound is imposed in order to avoid exponentials (with negative exponent) weighted by a negative value. This would give rise to concave functions with which to approximate a convex one (one over the square root), which is not in the spirit of the methodology explained in Section (2.4), which tries to restrict the set of exponentials to those resembling the Basset kernel as much as possible. On the other hand, the upper bound of parameter a_i is arbitrary. However, this bound helps the optimizer to find the solution in a smaller space. Of course, sub-optimal values could be expected by this simplification, but in our numerical experience, when we do not use the upper bound constraint, the optimal solution always fulfils it anyway.

Likewise, the positivity of $\tilde{t}_i$ is required to keep the values of the modified Basset kernel real. Furthermore, requiring $\tilde{t}_i > 1$ may seem reasonable. This variable represents the non-dimensional history time, thus taking values on the window part may appear as unnatural. However, we do not add this constraint in order to get better solutions. In fact, the numerical results explained in the following sections support this idea. [12] had also made this point.

The box constraints can be directly handled by means of the line search method [25]. The problem contains a small number of design variables ($2m$), which represent at most 20 unknowns for the cases considered. Thus, the complexity in solving the optimization problem will depend mainly on the nature of the cost function.

We next present the different cost functions that we have considered. I_1 is included along three additional alternatives leading to more tractable optimization problems.

Option A: I_1 cost function

The first option is defined by the use of I_1 itself (see 17), which we repeat here for completeness:

$$I_1(a_i, \tilde{t}_i) = \left| e_K(1, a_i, \tilde{t}_i) \right| + \int_1^\infty \left| \frac{\partial}{\partial \tilde{\tau}} e_K(\tilde{\tau}, a_i, \tilde{t}_i) \right| \, \mathrm{d}\tilde{\tau}. \tag{32}$$

Two terms are involved: the absolute value of the error at the point where the window and the tail meet $(t - t_w)$ plus the absolute value of the error derivative integrated along the whole tail. The second term can

be interpreted as the $W^{1,1}([1,\infty);\mathbb{R})$ semi-norm. The latter penalizes the outliers weakly, so the error is expected to be small in most of the points but remain large in few points.

Since the L^1 norm is linear with respect to its argumen, the convexity and non-linearity properties of the problem are essentially determined by the dependence of the error derivative with respect to $\tilde{t}_i$ (it also linear with respect to the a_i).

The abs-value function is not continuously differentiable, so a non straightforward treatment on the computation of the gradient will be required.

Option B: I_2 cost function

Alternatively, we can consider replacing the absolute-value function in the I_1 cost function by the square function. That is

$$I_2(a_i,\tilde{t}_i) = e_K(1,a_i,\tilde{t}_i)^2 + \int_1^\infty \left(\frac{\partial}{\partial\tilde{\tau}} e_K(\tilde{\tau},a_i,\tilde{t}_i)\right)^2 \mathrm{d}\tilde{\tau} \tag{33}$$

This option leads with a stronger penalization of the outliers, i.e., points with large errors in the derivative tend to be eliminated more easily. In this case, the second term in the cost function can be interpreted as the $H^1([1,\infty);\mathbb{R})$ semi-norm, or the $L^2([1,\infty);\mathbb{R})$ norm of the error derivative.

With the L^2 norm we gain convexity but we add nonlinearity to the already nonlinear dependence on $\tilde{t}_i$. However we add a quadratic non-linearity which is relatively weak. In addition, this modification also contributes to improve regularity (I_2 is continuously differentiable) and no special treatment of the gradient is required.

Option C: I_{2t} cost function

In this case, we replace the abs-value of the cost function by taking the square, to which we add an extra weight $\tilde{\tau}$.

$$I_{2t}(a_i,\tilde{t}_i) = e_K(1,a_i,\tilde{t}_i)^2 + \int_1^\infty \tilde{\tau}\left(\frac{\partial}{\partial\tilde{\tau}} e_K(\tilde{\tau},a_i,\tilde{t}_i)\right)^2 \mathrm{d}\tilde{\tau}. \tag{34}$$

This extra weight $\tilde{\tau}$ was included in [12] 'to correct for the change in norm'. With it the values at the end of the tail will be more strongly penalized than the values at the beginning of the tail. The convexity and non-linear aspects are very similar properties to those from Option B.

Option D: I_{2tH} cost function

This option can be understood as a restricted version of Option C, were the $\tilde{t}_i = \tilde{T}_i$ are given. That is

$$I_{2tH}(a_i) = I_{2t}(a_i,\tilde{T}_i) = e_K(1,a_i,\tilde{T}_i)^2 + \int_1^\infty \tilde{\tau}\left(\frac{\partial}{\partial\tilde{\tau}} e_K(\tilde{\tau},a_i,\tilde{T}_i)\right)^2 \mathrm{d}\tilde{\tau}. \tag{35}$$

So that the dependence with respect to $\tilde{t}_i$ has been removed. Basically, the $\tilde{t}_i$ values (represented by $\tilde{T}_i$) must be provided as data. This is the cost function that was explored in [12], where an increasing separation between successive values was suggested for the T_i. Its properties are similar to those of Options C, but since the space of possible solutions has been reduced, we should expect higher optimal costs in this case.

Note that after a few manipulations (detailed in Appendix C) and taking $x_i \coloneqq a_i$, the optimization problem can re-expressed in matrix notation as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^T A x - b^T x \\ \text{subjected to:} & 0 \le x \le 1. \end{array} \tag{36}$$

where A and b are also detailed in Appendix C. Equation (36) thus has the form of standard quadratic programming problem. More specifically, with a quadratic cost function and box constraints. This kind of problem can be approached effectively by standard optimization algorithms [25].

3.2. Exploring the character of the cost functions

In this section, we explore the behaviour of the different options introduced above as a function of the variables a_i and $\tilde{t}_i$ at low dimensionality ($m = 2$), to facilitate visualization.

Figure 1 shows the cost function for Options A and C in terms of a_1 and $\tilde{t}_1$. Note the discontinuity on the derivative of the cost function in Figure 1a. In contrast, Figure 1b shows the smoothness of the cost function I_{2t}, which greatly facilitate the search for minima. Its graph is otherwise very close to that of I_1 (Option A). We will show that the gain in regularity makes up for the difference.

Note that the cost function is, in both cases non-convex, specifically with respect to $\tilde{t}_1$. Therefore local minima are expected to appear. Indeed, in practice we observed the appearance of multitude of local minima, whose number grew very quickly with dimensionality. Nonetheless, for this low-dimensional case it is possible to exactly determine the optimum point, which is marked in Figure 1.

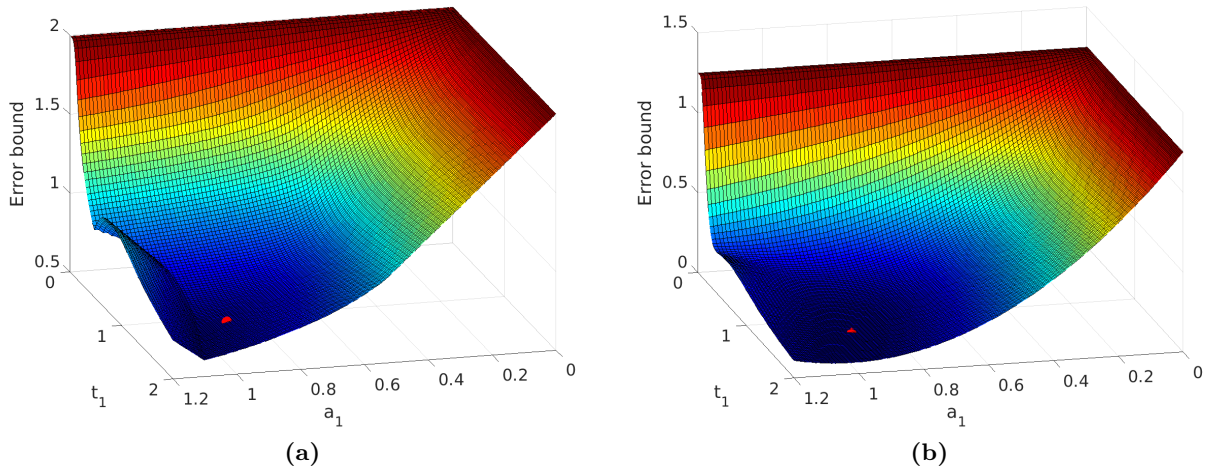


Figure 1: *Error bound for one exponential approximation of the Basset kernel versus a_1 and t_1 for both Options A and C. The respective minima are marked with a red dot.*

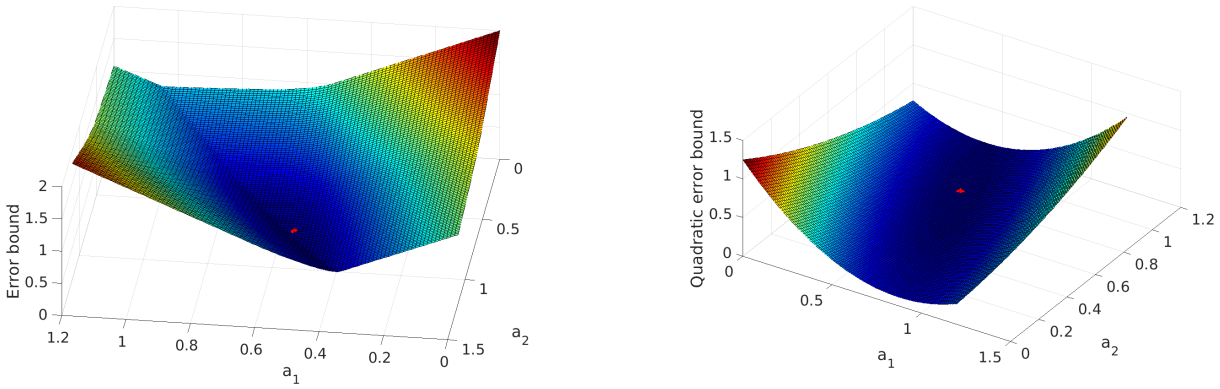
Next we study the dependence of the different cost function with respect as a_1 and a_2 are varied, while $\tilde{t}_1^*$ and $\tilde{t}_2^*$ are held fixed to their optimal values. Note from Figure 2 the convexity of both cost functions with respect to the dependence with respect to the a_i, as compared to $\tilde{t}_i$. Certainly, to consider only the dependence on a_i makes the optimization problem much more tractable, especially as the dimensionality grows.

Finally, let us look at the dependence on $\tilde{t}_1$ and $\tilde{t}_2$ alone. Figure 3 shows a contour plot of the I_1 bound in terms of $\tilde{t}_1$ and $\tilde{t}_2$ where, this time, a_1^* and a_2^* are being held fixed (at their optimal values). The strong gradients and the nonlinearity of the dependence on the $\tilde{t}_i$ can be clearly appreciated. The cost function is almost flat close to the optimum, but away from it the gradient grows very quickly. This could indicate that the selection of the $\tilde{t}_i$ might not need to be determined with extremely high precision. However, obtaining a fair approximation should be essential. Indeed, we have found in practice this to be the case (see next section).

3.3. Numerical solution of the optimization problem

In this work we have developed an in-house code in Matlab © and solved the problem with a standard PC (3.40GHz processor in a 64-bit architecture). We have nonetheless taken advantage of symbolic coding mode in the Matlab environment to calculate all the derivatives and integrations.

Furthermore, We have also employed the optimization toolbox of Matlab. This tool makes combined use of a variety of methods (Newton-Rapshon, Quasi-Newton and steepest descent, among others) for constraint optimization (box constraints in our case). Continuous optimization algorithms helped to obtain the real (local) minima. Furthermore, their convergence is normally faster than the alternative genetic algorithms.



(a) **(b)**

Figure 2: *Error bound for two exponentials as a function of a_1 and a_2, with fixed (optimal) $\tilde{t}_1^*$ and $\tilde{t}_2^*$ for Options A and C. Options C and D are equivalent in this case.*

Specifically, we have used the interior point algorithm (primal-dual Newton-Raphson), which considers both the primal and dual variables simultaneously (see, e.g., [25]).

Our numerical experiments show that the problem is full of local minima, thus the strategy of initial point seeding becomes crucial. For each case m we must determine a_i and t_i, with $i = 1, \ldots, m$. We propose the following heuristic initial point strategy, which has worked well up to $m = 10$, and consists in the following (An upper index m is used to refer to the case the parameters correspond to).

1. Take a_i^m randomly generated in $(0, 1]$.
2. For $m = 1$, take $\tilde{t}_1^1$ as also randomly generated in $(0, 1]$.
3. For $m = 2, \ldots, 10$, define

$$\tilde{t}_i^m = \begin{cases} \tilde{t}_i^m & \text{if } i = 1 \\ \frac{\tilde{t}_{i-1}^{m-1} + \tilde{t}_i^{m-1}}{2} & \text{if } i = 2, \ldots, m-1 \\ 10\tilde{t}_{m-1}^{m-1} & \text{if } i = m \end{cases} \tag{37}$$

This initial strategy is used when solving Options A, B and C. In combination with it, we also employed the Global-search toolbox of Matlab, which is a type of multi-start technique for Global Optimization. This tool granted the possibility of exploring many different initial points and was specifically used with the a_i variables.

3.3.1. Some particularized remarks for the different options

For Option A the gradient is discontinuous in some regions. So in order to have a robust algorithm, we computed the gradient by perturbations, though at a high computational cost. For Options B and C, the gradient could be computed analytically (symbolically), leading to faster performance. For option D, the gradient is also computed analytically, taking the t_i as specified in 3.3.

3.4. Results

Figure 4, shows the minimized a_i and $\tilde{t}_i$ values of the upper bound for the different options. We have also included the single result reported in [12]. Case C seems to slightly outperform the others, although option A is very close, except perhaps for the last point, where the computational cost was already very high. note that this implies that, at least for the points where $I_{2t} < I1$, either only local minima were found or convergence had not been achieved when using I_1 as the objective function. The complete list of optimized values a_i and $\tilde{t}_i$ corresponding to the I_1 and I_{2t} bounds can be found in Appendix E. Let us now make a few comments about these results, looking at each individual option.

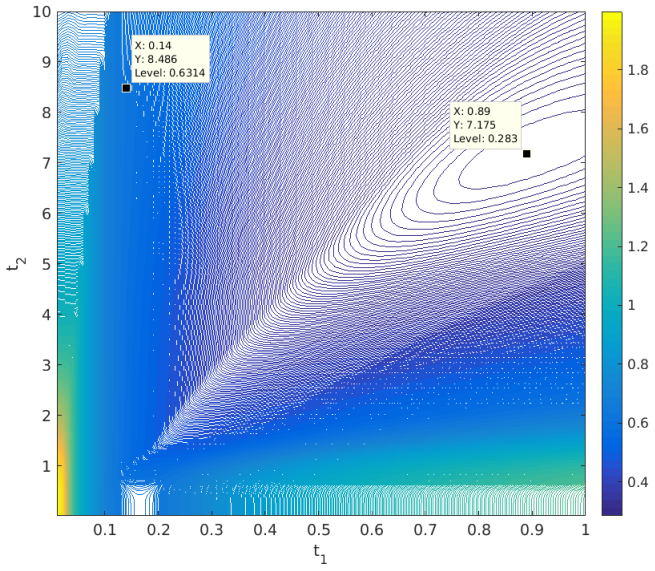


Figure 3: *Contour plot of the I_1 bound in terms of $\tilde{t}_1$ and $\tilde{t}_2$. The dependence with respect to the t_i is clearly visible.*

For Option A (I_1), for which the gradient is obtained by perturbations, the algorithm suffered significantly for $m > 4$. This was due large number of computations needed to evaluate the gradient, which increases exponentially with m when employing perturbations. That is, very sub-optimal solutions were found when the initial values where not set adequately. Precisely this fact justified the convenience of trying large number of initial values and, consequently, the use of Global-search with its associated computational costs. The results come out second-best (after Option C), even though this alternative is the only one that uses the original objective function, I_1.

For Option B and C (I_2 and I_{2t}), the computational cost per initial point was significantly less, thanks to having an analytical expression for the gradient. Still, finding appropriate initial values remained very demanding. We again observed a strong dependence on initial values and a huge number of local minima significantly hampered the search. Nonetheless, thanks to a reduced cost in computing the gradient, a much larger number of initial points could be employed. As a result, Option C came out slightly on top of Option B and clearly beat Option A and Option D by a considerable margin.

Regarding Option D, number of design variables is halved and, consequently, the number of possible initial values decreases. Though there are local minima, a much smaller number of them were found in this case, which considerably sped up the computations. Nonetheless, the optimized costs were significantly higher that for the other options. Note how, encouragingly, with our proposed strategy 3.3 a very similar cost is achieved for $m = 10$ compared to that of the parameter set reported in [12].

In general, the bounds monotonically decrease as the number of exponentials increases for all the options, as expected. By looking at a particular examples (see Section 4), we will see that the actual error also follows the same trend, although not as robustly.

Despite all the technology applied, the problem starts to become unaffordable for $m > 10$ for Options A, B and C. The high computational cost is basically caused by the number of required initial guesses, which strongly increases with m. This affects all the Options, except Option D, where the problem becomes less severe and the process could actually be carried on further. As we will see, this will not turn out to be necessary in many applications.

.

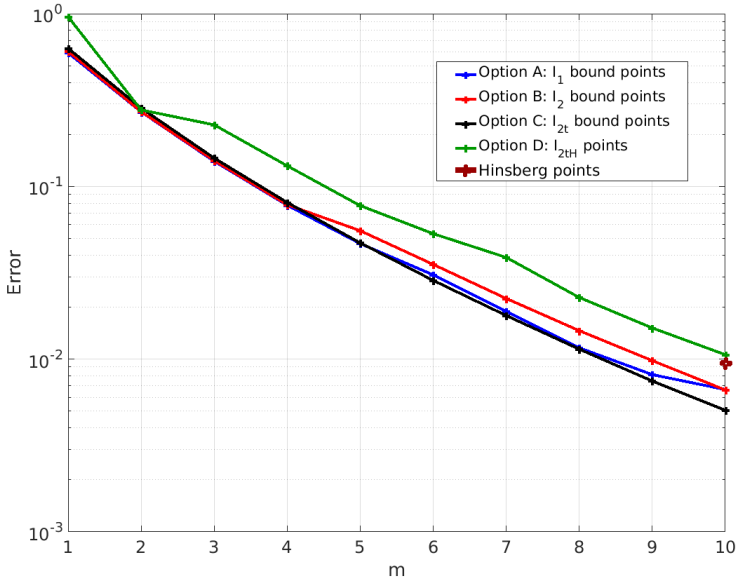


Figure 4: *Resulting values of the I_1 function for the optimized paramters using each of the different alternatives. The values provided by van Hinsberg et al. for $m = 10$ are also shown for comparison.*

4. Performance

In this section we test the performance of the MAE using the optimized a_i and t_i values from the previous section. We start a with very elementary example, aimed only at measuring the quadrature error. Next, we consider a single-particle example with analytical solution to benchmark the accuracy of the full scheme. The last example features a long-term, 10 000 particle simulation with which we show the remarkable efficiency of the method.

4.1. First benchmark: an integral with analytical solution

The error bound (17) can indeed be used for conservative predictions about the expected error when using the MAE. Nonetheless, we wish investigate how this bound in fact relates to the actual error, but of course this can only be done for particular cases. Consider the convolution of the sine function, denoted $\sin * K_B(t)$, a commonly chosen example with analytical solution; see [2, 12, 6][5]. Its physical significance is that of a sphere being forced to oscillate in an otherwise quiescent fluid. The error can be expressed, see (17), as

$$E(t) \coloneqq \left|\sin * \left(K_B - \tilde{K}\right)(t)\right| \leq \frac{1}{\sqrt{t_w}} \left(\left|1 - \tilde{K}(1)\right| + \int_1^\infty \left| \frac{\mathrm{d}}{\mathrm{d}\tau} \left(K_B(\tau) - \tilde{K}(\tau)\right) \mathrm{d}\tau \right| \right) \tag{38}$$

since $\|\cos\|_\infty = 1$. We have computed the bound numerically, by partitioning the integral in (38) in two parts: an integral over the region where the argument of the absolute value may change and an integral over the rest, where the sign is given by that of $\mathrm{d}K_B/\mathrm{d}\tau$ (which has a slower asymptotic decay than $\mathrm{d}K/\mathrm{d}\tau$ as $t \to \infty$). The first part can be calculated (to very high accuracy) using a standard quadrature method, while the second one has an analytical formula. In order to obtain a representative measure of the error, we evaluate the integral over $[-\infty, 2\pi - \phi_i]$, with ϕ_i sampled at 40, evenly distributed, points in $[0, 2\pi]$ and we take the mean absolute error within the sample [6]. We denote this error as $E_{2\pi}(t_w, m)$, making explicit its dependence on both t_w and m. The results are shown in Figure 5, where three different sets of parameters

[5] Actually, the convolution of the cosine was considered by [2] and by [12].

[6] Note that the convolution of the sine function is periodic, with the same fundamental period as the sine, as it is readily seen with a change of variable.

are considered: the single list given in [12] ($m = 10$), resulting from the optimization of I_{2tH}; the set obtained from the optimization of I_1, for $m = 1, \ldots, 10$; and the set obtained from the optimization of I_{2t}, again for $m = 1, \ldots, 10$ (see Section 3). For the sake of clarity, we include a single set of results in Figure 5a where we have picked $t_w = 2\pi/10$. Figure 5b shows the analogous results for a range of window times, including the single curve from Figure 5a.

Figure 5 shows how indeed the errors fall significantly below their optimized upper bounds, up to more than an order of magnitude so. Overall, it seems that both the I_1 and the I_{2t} methods achieve comparable results, while the parameters by van Hinsberg et al. come out as a bit less accurate. Nonetheless, this difference turns out not to be consistently significant, despite our inclusion of the t_i parameters as variables in the optimization problem. This indicates that the trial-and-error strategy used by these authors yielded a close-to-optimal result.

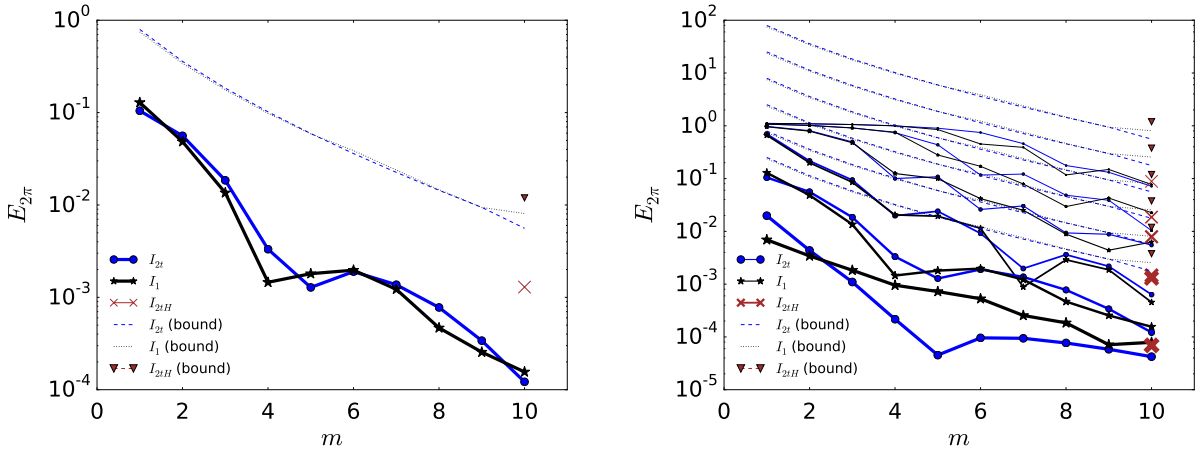


Figure 5: *Error produced by the approximation $K \approx K_B$ in calculating $\sin * K_B(t)$ for three different sets of values a_i and t_i, corresponding to different cost functions: Option A, Option C and the point given by van Hinsberg et al.. The resulting errors are accompanied by their predicted upper bounds. A window time of $t_w = 0.2\pi$ has been considered in Figure 5a. Figure 5b shows the analogous results for different values of t_w corresponing to $t_w = 2\pi 10^k$ for $k = -5, \ldots, 0$ (with thicker lines corresponding to larger values of t_w).*

In theory, the greater the window, the smaller the error should be, because the approximation of the kernel is done over a smaller portion of the total domain. This tendency is expected to be especially pronounced for small values of t_w, close to the singularity of K_B at $\tau = t$. There, the well-behaved exponentials have difficulty keeping close to the curve for as it diverges. This is indeed what can be seen in Figure 5b. The time window ranges 1×10^{-5} to 1 fractions of a period and is represented by using greater thickness on the error curves corresponding to larger windows. The tendency of the curves to go horizontal for increasing m signals the breakdown of the kernel approximation for extremely small values of t_w. Note that, for t_w smaller than 1×10^{-3}, none of our optimal a_i, t_i sets manage to keep the error under the 1×10^{-2} mark.

However, a small t_w requires an even smaller integration time step. So as long as the time step is kept large enough, such small time windows become unnecessary, because the memory cost can be afforded. This is why the most effective solution is to use a high accuracy scheme, such as the second or third Daitche schemes for the integration of the window in the MAE.

A final remark about Figure 5b: note that although the expected monotonous behaviour of the error with respect to variations in t_w is mostly realized, there can be exceptions: for $m = 7$ there is a crossing between adjacent sets of curves. This fact highlights the complexity of the relations that govern the method.

A different way to characterize the accuracy gains obtained by increasing the number of exponentials is represented by Figure 6. It shows the relation between the number of exponentials, m and the minimum window time necessary to attain a desired level of accuracy. On the horizontal axis we represent m; on the

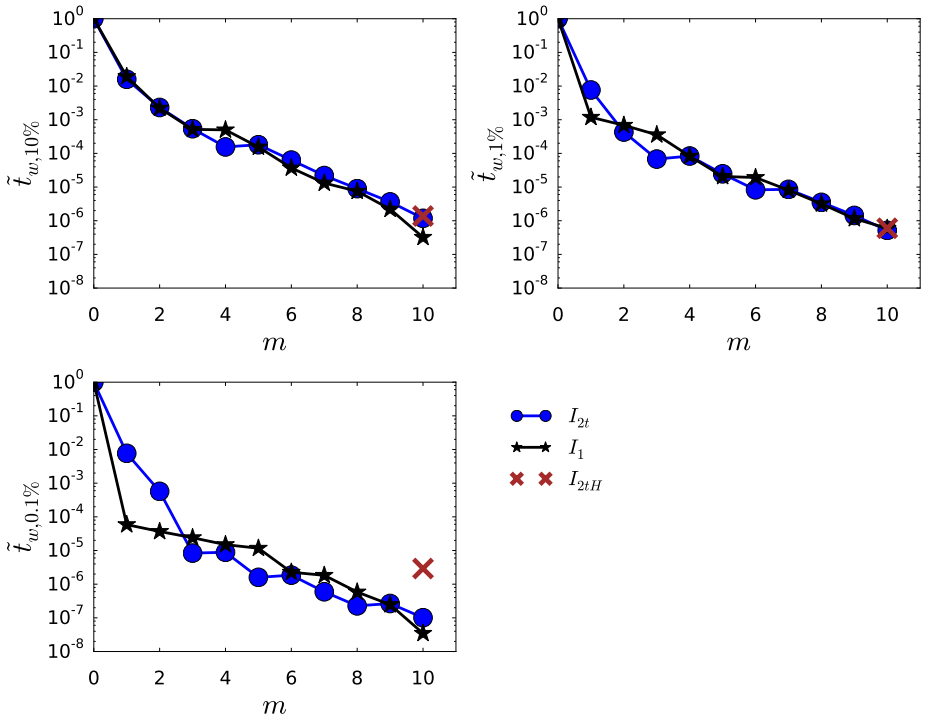


Figure 6: *Minimal window time necessary to obtain an error $E_{2\pi} = 1\%$, as a function of the number of exponentials. The time is normalized by the minimal time window required when $m = 0$.*

vertical axis, the normalized window time $\tilde{t}_{w,1\%}(m)$. The latter is defined as

$$\tilde{t}_{w,1\%}(m) \coloneqq \frac{t_{w,1\%}(m)}{t_{w,1\%}(0)} \tag{39}$$

where

$$t_{w,1\%}(m) \coloneqq t_w \quad \text{such that} \quad E_{2\pi}(t_w, m) = 0.01 \tag{40}$$

and similarly we define $t_{w,10\%}$ and $t_{w,0.1\%}$. In other words, Figure 6 addresses the question *if $t_{w,x\%}(0)$ is the minimal time window necessary to have less than $x\%$ error when using the standard window method, then what fraction of $t_{w,x\%}(0)$ is instead required when using the MAE?*. The answer will depend on the values of m and x. Strictly speaking, the existence of a unique solution is not even guaranteed, since that would require the strict monotonicity of the dependence of $E_{2\pi}$ on m, which we have seen to only hold approximately. Nonetheless we have applied the bisection algorithm to find such solutions, producing Figure 6. Note the immense memory cutoffs that are generated by the MAE. For instance, suppose the accuracy goal set to one percent. Then just by including a single exponential approximant, the interval of the particles history that must be tracked is reduced by a factor 10^3, by just using the I_1-optimized a_1 and t_1. Or else by more than 10^6 if ten exponentials are considered.

Let us now consider the joint effect of the quadrature algorithm and the kernel approximation, using the same test flow as above. For that, the initial time is taken as $t_0 = 0$, the final time is set to 10. The time interval $[0, 10]$ is initially partitioned in eighty parts. This amount is successively doubled, defining the range of values of the time step, h. For simplicity, the error is now measured at single point ($t = 10$); that is $E(10)$. Figure 7 shows the performance of the algorithm described in Section 2.5 for a fixed $t_w = 1$ and $m = 10$. The method initially follows the third order slope of the second-order accurate Daitche quadrature algorithm. But then, as soon as the time integration error becomes dominated by the error due to the approximation $K \approx K_B$, its accuracy stagnates, rendering further reductions in h futile.

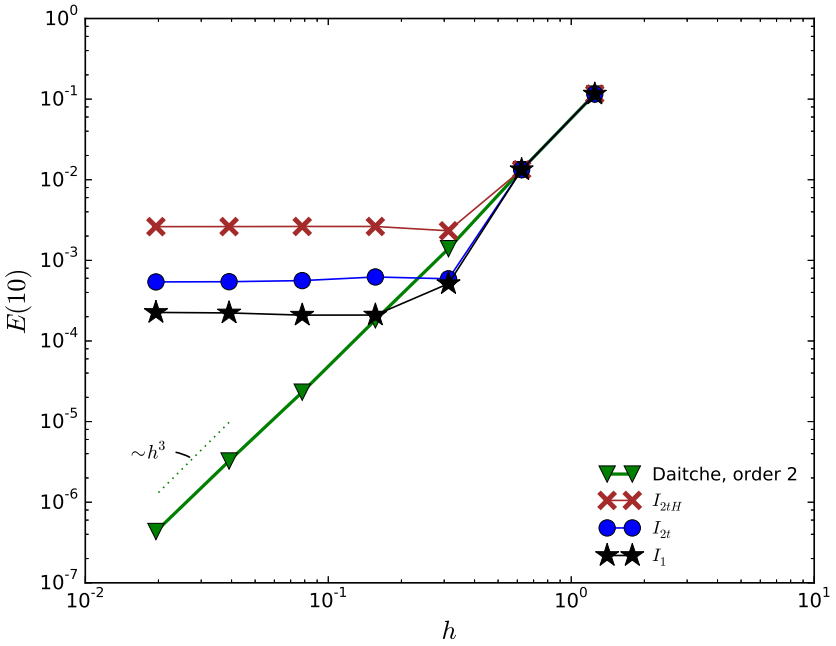


Figure 7: *Error vs time step of Daitche's method as compared with the MAE for different optimized values t_i and a_i. The curves reach a plateau as soon as the kernel approximation error exceeds the quadrature error.*

4.2. Second benchmark: Candelier's solution

There are a limited number of known closed-form particular solutions for the Maxey-Riley equation. Some of these can be found in the works cited in [10]. Fortunately, the solution obtained by Candelier et al. [3] includes the effect of the Basset force, as well as all the other forces, though the Faxén terms do not contribute in this case. The solution corresponds to the trajectory of a particle that sediments under gravity in an infinite container which is rotating as a rigid solid around a fixed axis. The same benchmark was considered in [6] (see the same work for the input parameters). Figure 8 shows two spiral trajectories resulting from two different versions of the solution by Candelier and co-workers. The inner spiral corresponds to the full solution, including all the terms in the MRE. The outer spiral is obtained by neglecting the Basset contribution, while retaining the remaining effects. The asymptotic behaviour of the radial coordinate around the rotation axis is in both cases of exponential form, of which the exponent's coefficient is modified by the presence of Basset force. This asymptotic solution becomes a very good approximation only after a few turnover times [3]. The exponential form suggests that this test is especially demanding, as it should tend to amplify any systematic inaccuracies over time.

For the purpose of measuring the accuracy of the algorithm, let us define the error function as

$$E(t) = \frac{r(t) - r_{\text{num}}(t)}{r(t)} \tag{41}$$

where $r(t) = \sqrt{x(t)^2 + y(t)^2}$ is the exact radial coordinate and r_{num} its numerical approximation. We consider $E(100)$ as the measure of the error. This measure has a very similar behaviour to the max-norm over all the time steps. This is because the error tends to accumulate in a monotone way, leading to the maximum value occurring at the final point in most cases; see [6]. For instance, the error introduced by neglecting the Basset force at $t = 100$ is $\frac{r(100) - r_{\text{no Basset}}(t)}{r(100)} > 14$.

Figure 9 shows the error for different values of m for the MAE as compared to the bare, second order Daitche method. The optimal points were those obtained with the cost function I_1, since this set gave slightly more accurate results for all values of m in this example. We observe that, despite a few permutations do

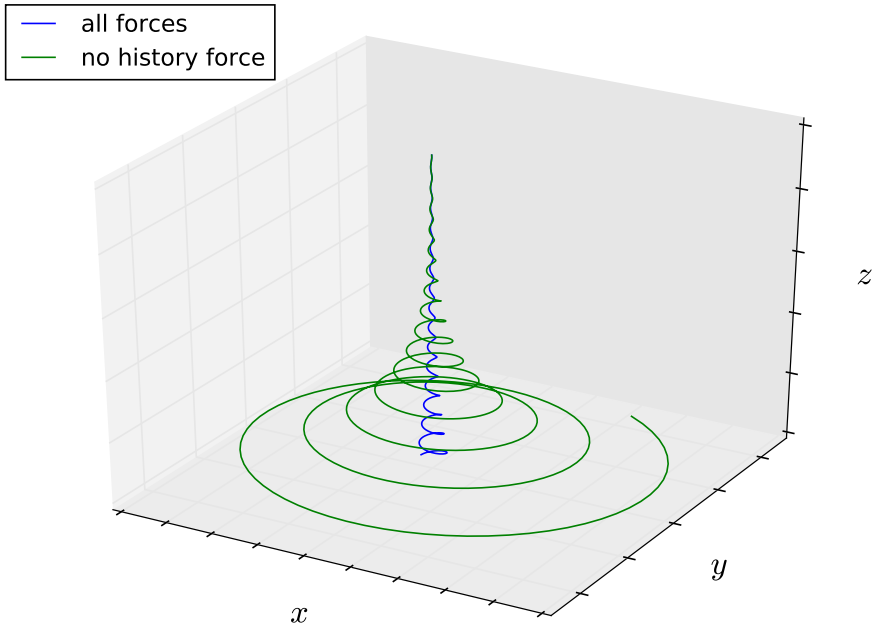


Figure 8: *Trajectories described by the particle in the Candelier et al. benchmark after* 100s *(*$50/\pi$ rotation periods*). The mass of fluid is rotating around the axis* $x = y = 0$*. The curve with the largest maximum radial coordinate corresponds to the solution without Basset force, while the other does include the effect.*

occur between curves with successive m values, in general terms increasing m yields an increase in accuracy which, on average, amounts to about half an order of magnitude each successive increase.

But, *how many exponentials should be used?*. Based on accuracy, the safest answer should be *as many as possible*, or at least, *as many as necessary*. However, taking efficiency into consideration might complicate the matter. Indeed, adding additional exponentials to the kernel has two detrimental effects on efficiency. First of all, it implies a proportional increase in the total number of operations needed to compute the tail contribution. But more importantly, each new exponential requires an extra vector to be kept in memory per particle, i.e., the value of $\mathbf{F}_i$ in the previous step, see (13). Of course it is still possible to increase the time window, thus improving the effectiveness of a fixed number of exponentials, as Figure 5 suggests. But again, this increase also implies a raise in the memory demand. The situation is summarized in Figures 10, where the error is plotted against the total number of bytes to be kept in memory per particle. The number of bytes is estimated by assuming that 24 bytes are taken up by each vector, as

$$\text{number of bytes} = 24(m + 1 + t_w/h) \tag{42}$$

where the time step h is kept constant at 2.5×10^{-3}, close to the saturation time step (below which no further gains are obtained, see Figure 9). This corresponds to an error of around 10^{-3} for 8, 9 or 10 exponentials, according to the same figure. The window time is successively doubled, starting at $t_w = 5h$. Again, Figure 9 shows that only initial increases in the memory demand (via increase in the number of exponentials or the value of t_w) yield significant gains in accuracy. Furthermore, the gains in accuracy per byte are greater when investing them into more exponentials rather than into additional t_w (except perhaps for the very smallest values of t_w). In other words, within the analysed ranges, it pays off to take t_w as small as possible, while using as many exponentials as necessary.

4.2.1. Some remarks concerning the choice of t_w

It is not quite clear how small is 'as small as possible'. The smallest value considered in Figure 10 is $t_w = 0.05$, or $t_w \approx 0.008$ periods. This t_w corresponds to an error of around 10^{-3}, based on Figure 7. Making

it smaller might very quickly lead to important approximation errors. Furthermore, since the time step is only a fifth of that amount, only five points enter the window region. Any further decrease in t_w would surely damage accuracy, while not significantly reducing the memory requirements, according to (42), with $m = 10$.

Based on this line of reasoning (and our numerical experience), we conclude that ten points are a reasonable option, which we have adopted by default in conjunction to $m = 10$. This translates into 21 vectors to be stored per particle. Indeed, for a fixed integration time step (which is typically determined by other factors, such as the overall time scheme or the fluid time resolution) the accuracy gain is quite substantial compared to, say, only five points, while still avoiding important memory sanctions. We do recommend this combination as a starting point in other simulations, as long as the frequency of the relative motion is low enough. That is, making sure that at least a few time steps fall within the period corresponding to the highest frequency modes in the motion. For instance, in a turbulent flow, this frequency is that of the Kolmogorov microscales, unless some external, high frequency force is acting on the particle. We test this 10-point, $m = 10$ combination in the following benchmark.

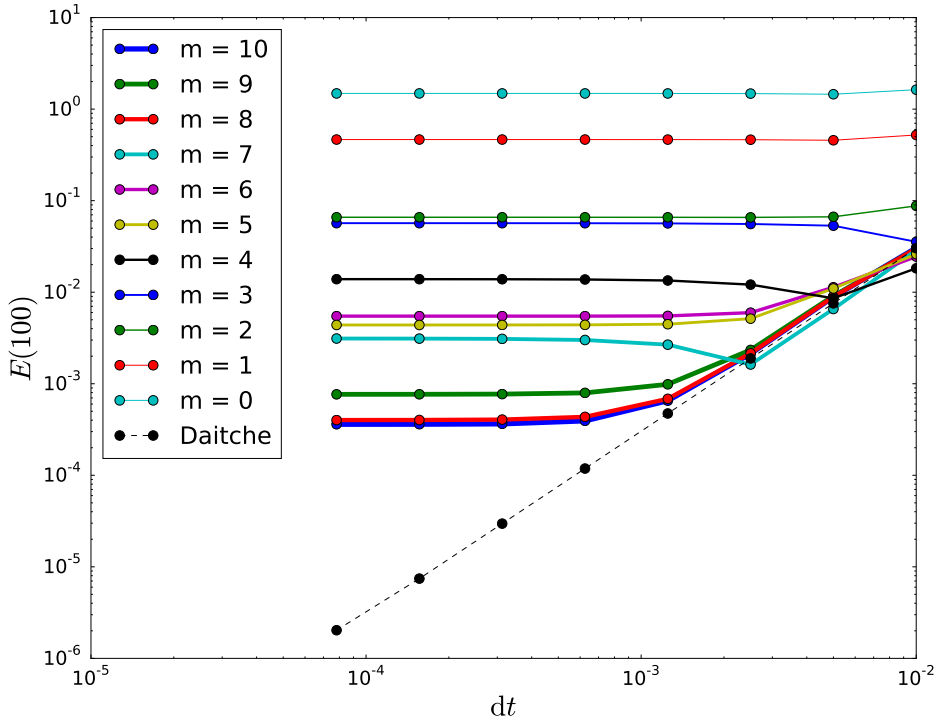


Figure 9: *Average relative error of the particle position (final time $t = 100s$) for different time-steps. Both the raw second-order accurate method of Daitche and the corresponding MAE alternative for different number of exponentials, m. The window time is taken as $t = 0.1s$*

4.3. Third benchmark: Sedimentation through synthetic vortices

We now wish to test the efficiency of the full algorithm. For that, we will consider the benchmark test that was considered in Guseva et al. [11] in their work about the influence of the history force on the dynamical properties of a system of sedimenting particles. These authors studied a synthetic bidimensional flow that is a transient variant of the classic cellular flow field previously studied in, e.g., [19]. The flow field is given by

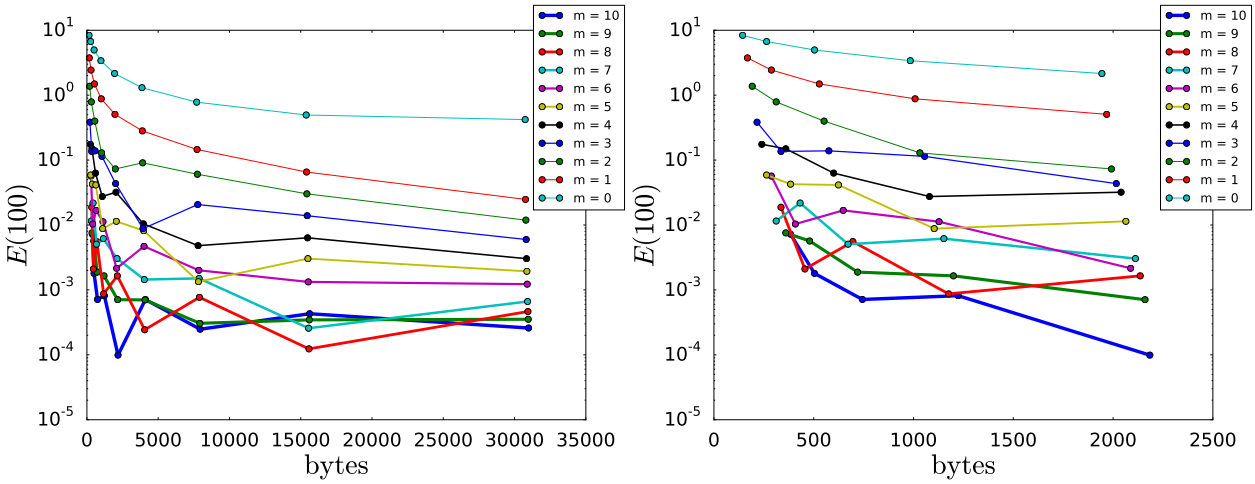


Figure 10: *Relative error of the particle position at the final time ($t = 100s$) for different memory loads per particle. Figure 10b is a zoomed-in version of Figure 10a, showing the intial part with the lesser amount of bytes.*

[7]

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = U\left(1 + k\sin\left(\omega\frac{U}{L}\right)\right)\begin{pmatrix} \sin\left(\frac{x_1}{L}\right)\cos\left(\frac{x_2}{L}\right) \\ \cos\left(\frac{x_1}{L}\right)\sin\left(\frac{x_2}{L}\right) \end{pmatrix} \tag{43}$$

where x_i are Cartesian coordinates and U and L are the characteristic velocity and length-scales of the flow; ω controls the frequency of the temporal evolution of the flow and k its amplitude. Such flow covers the Cartesian plane with $2\pi L$-diameter vortices, each one flowing in the opposite sense than the ones directly next to it.

Guseva and co-workers were interested in studying the long-term evolution of a number of particles subject to the above flow in a double-periodic domain containing four vortices. They monitored 10 000 particles for up to 120 periods employing the first-order accurate version of the Daitche algorithm. They did not extend the simulation duration any longer, since, as they put it: 'A longer interval is not possible to choose due to the numerical cost of recording the history force with small time step for this number of particles'.

We attempted to reproduce one of their examples using the MAE with our optimal points and use it as a test for efficiency. The values chosen for the input parameters are consistent with [11] and roughly correspond to the typical conditions in the phenomenon known as *marine rain* [8]. This term refers to the sedimentation of small agglomerates of mainly organic mater from the surface to the deep ocean, subject to the turbulence found in the upper layers due to the action of wind; see [11] and the references therein. Their values are summarized in Table 1. While these are in dimensional form, they correspond to the same non-dimensional variables used by [11].

Figure 11 shows the position of ten thousand particles suspended in the cellular flow at three different time instants. The particles where initially placed in a uniform lattice covering the whole domain, as was done in [11]. The effect of the history force is apparent: With $\boldsymbol{F}_H \equiv \mathbf{0}$ the particles become confined to a set of four curves. In contrast, when including all forces, full bands with particles are still visible at the latest stage, although the bands' edges do become more sharply marked. These qualitative trends are consistent

[7] The flow described in [11] differs from the one described here in that, in it, the arguments of the sine and cosine functions are pre-multiplied by π. We however only managed to obtain similar results to the ones reported in their work upon removing this π factor. We suspected that the difference was simply due to a misprint. In contacting the authors, this possibility was given credibility and thus we have assumed this to be the case. Note that, in particular, this modification changes the meaning of L, which is no longer equal to the diameter of a vortex, but rather 2π times smaller.

[8] Except perhaps for the peculiar value for the gravity (see Table 1, which is the only value not given in dimensional form by [11], although its value is determined by the nondimensional parameters used therein)

with [11].

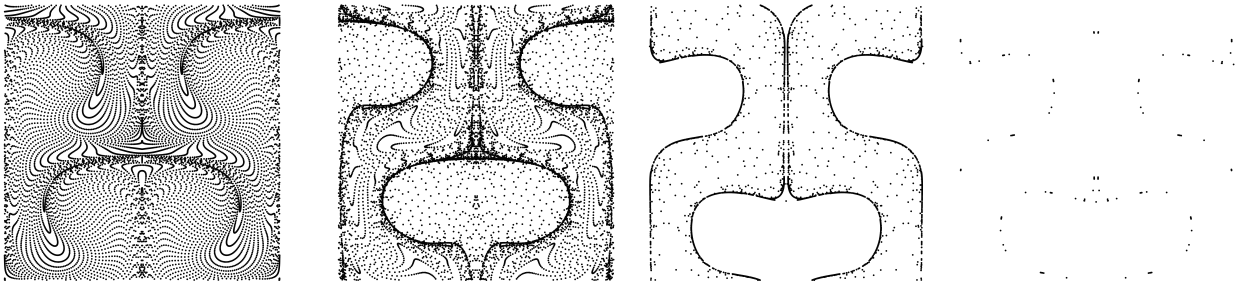
However, we must point out some differences as well. Take the configuration corresponding to $t = 80\,\mathrm{s}$ (120 periods). For the case of no history, the configuration reported by [11] (Fig. 4, (c)) looks much more similar to our configuration at $t = 20\,\mathrm{s}$ than the one at $t = 80\,\mathrm{s}$. For the case with all forces, our configuration is closer to the one reported by Guseva (Fig. 4, (d)), although some differences can be found still. In particular, for $t = 80\,\mathrm{s}$, a dark spot lying on the axis of symmetry of the figure signals an accumulation of about fifty particles, which is not marked in [11] (note also its counterpart near the top-right that corresponds to the top pair of vortices, with another fifty particles). Furthermore the particles concentrate in areas with much sharper edges in our case, again as if a faster convergence was taking place. We tried different integration schemes, combined with the first-order version of the Daitche quadrature but always obtained similar results. After much scrutiny, we could not find the reason for such divergence and thus we leave the question open for other researchers to settle.

We now turn to the comparison between the second and third rows in Figure 11. Comparing the top two rows, no visible effects arise due to the use of the window method at any of the recorded times. However, the total computational time is greatly decreased by this, as is apparent from Figure 12, where the elapsed time per unit simulated time is represented. The steady increase in memory resources and the number of computations to be performed translates into ever higher costs for the Daitche method. The MAE, instead, stabilizes after a few time-steps, once the window region is completely filled. Note that in our implementation the cost is of the same order as for the case of neglecting the history force altogether.

Admittedly, our implementation is mounted on top of a 3D discrete element code, where neighbour search has been switched off. The flexibility of the application sacrifices some efficiency and thus one cannot conclude the latter statement to be the case in more optimized implementations. Indeed, more detailed analyses with the help of profilers show that a vast majority of the time spent in computing the hydrodynamic forces is still being spent with the history force. Further analysis and reductions of the history force cost are thus in order. However we have demonstrated that the MAE, as described, already *critically improves the situation*. The perspective of routinely including the Basset force in the numerical implementation of the Maxey–Riley seems now much closer.

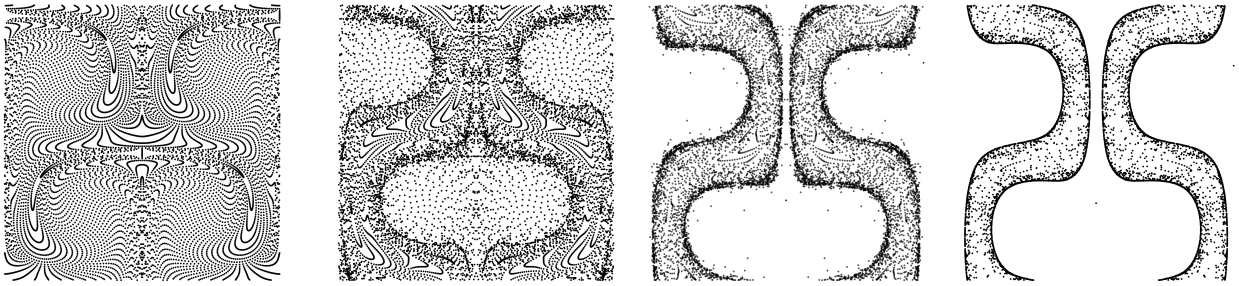
Parameter	**Value**	**Description**
g	$8.624\,\mathrm{m/s^2}$	gravity acceleration
	Flow parameters	
U	$0.3\,\mathrm{m\,s^{-1}}$	characteristic velocity
L	$2\pi\,\mathrm{m}$	characteristic length
k	2.72	trans. amplitude parameter
ω	π	trans. frequency parameter
ρ_f	$1000\,\mathrm{kg/m^3}$	fluid density
ν	$1 \times 10^{-6}\,\mathrm{m^2/s}$	fluid kinematic viscosity
	Particles parameters	
a	$3.9685 \times 10^{-4}\,\mathrm{m}$	particle radius
ρ_p	$1500\,\mathrm{kg/m^3}$	particle density

Table 1: *Physical parameters considered in the cellular flow example.*



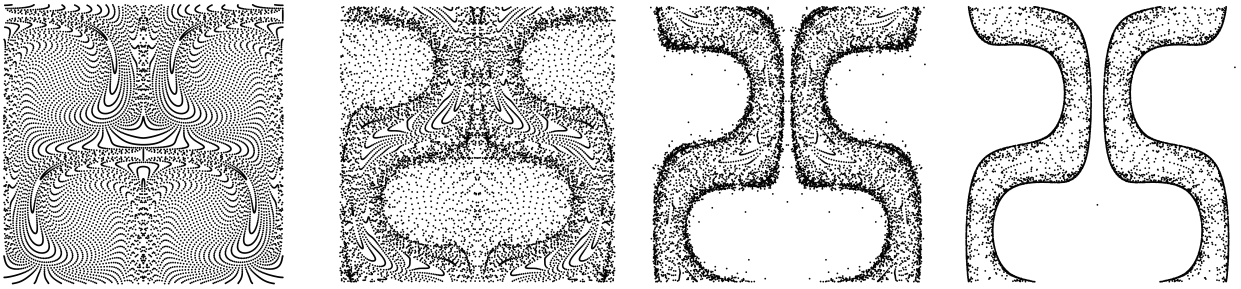
(a) *t=1* **(b)** *t=5* **(c)** *t=20* **(d)** *t=80*

No history force, Daitche



(e) *t=1* **(f)** *t=5* **(g)** *t=20* **(h)** *t=80*

All forces, Daitche



(i) *t=1* **(j)** *t=5* **(k)** *t=20* **(l)** *t=80*

All forces, MAE (m = 10)

Figure 11: *Position of the 10000 particles at different times in double-periodic spatial representation.*

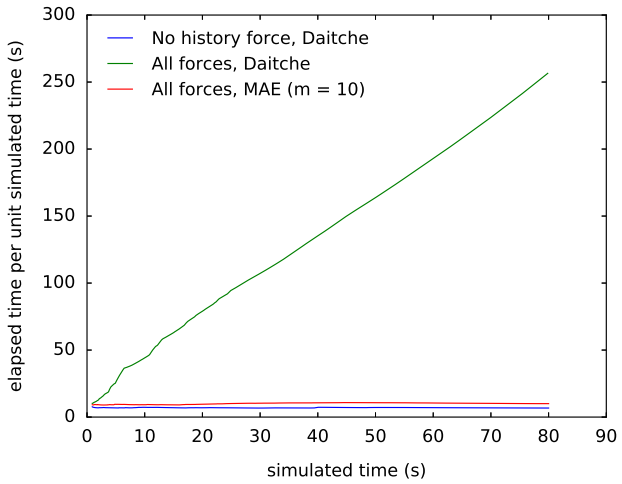


Figure 12: *Evolution of the wall-clock time spent in seconds, per unit simulated second, as a function of the simulation duration for the Daitche method, MAE and neglecting the history force. All the runs were performed with the same time step on the same PC (serial implementation).*

5. Concluding Remarks & Outlook

We have presented a detailed algorithm that combines the higher order quadrature scheme presented in [6] (Daitche method) with the window method presented in [12] (MAE).

The MAE includes several free parameters that must be determined. In the original formulation, half of the parameters (the a_i) where determined by solving an optimization problem, while the other half (the t_i) had to be fixed based on heuristic arguments. We have generalized the original formulation by including all the parameters in the optimization.

It turns out that the resulting problem is significantly more challenging. To address it, We have combined advanced optimization techniques with relatively large computational resources, producing the list of parameters summarized in Appendix E. By extending the problem, we have introduced a strategy that could inspire future research to continue refining and/or extending our list.

In any case, with the current list of parameters the overall algorithm already yields very high performance. We have shown this in several tests, where the method displayed remarkable accuracy. We have demonstrated that the reductions in the memory requirements can consistently be expected to exceed three orders of magnitude, even when the accuracy requirements are strict. As a consequence, the computational cost of including the history force has not changed the order of magnitude of the overall performance of the code in a long simulation with 10 000 particles. We find this result remarkable, since it is an extended view that the inclusion of this force dramatically hinders performance.

6. Acknowledgments

The authors are very grateful to Ksenia Guseva, who patiently answered all our questions.
We acknowledge the financial support to CIMNE via the CERCA Programme / Generalitat de Catalunya.

References

[1] Auton, T.R., Hunt, J.C.R., Prud'Homme, M., 1988. The force exerted on a body in inviscid unsteady non-uniform rotational flow. Journal of Fluid Mechanics 197, 241–257.

[2] Bombardelli, F.a., González, A.E., Niño, Y.I., 2008. Computation of the Particle Basset Force with a Fractional-Derivative Approach. Journal of Hydraulic Engineering 134, 1513–1520.

[3] Candelier, F., Angilella, J.R., Souhar, M., 2004. On the effect of the Boussinesq-Basset force on the radial migration of a Stokes particle in a vortex. Physics of Fluids 16, 1765–1776.

[4] Chong, K., Kelly, S.D., Smith, S., Eldredge, J.D., 2013. Inertial particle trapping in viscous streaming. Physics of Fluids 25, 1–21.

[5] Coimbra, C.F.M., Rangel, R.H., 2001. Spherical particle motion in harmonic stokes flows. AIAA Journal 39, 1673–1682.

[6] Daitche, A., 2013. Advection of inertial particles in the presence of the history force: Higher order numerical schemes. Journal of Computational Physics 254, 93–106. 1210.2576.

[7] Daitche, A., 2015. On the role of the history force for inertial particles in turbulence. Journal of Fluid Mechanics 782, 567–593. arXiv:1501.04770v1.

[8] Dorgan, A.J., Loth, E., 2007. Efficient calculation of the history force at finite Reynolds numbers. International Journal of Multiphase Flow 33, 833–848.

[9] Elghannay, H.A., Tafti, D.K., 2016. Development and validation of a reduced order history force model. International Journal of Multiphase Flow 85, 284–297.

[10] Farazmand, M., Haller, G., 2015. The MaxeyRiley equation: Existence, uniqueness and regularity of solutions. Nonlinear Analysis: Real World Applications 22, 98–106. arXiv:1310.2450v3.

[11] Guseva, K., Feudel, U., Tél, T., 2013. Influence of the history force on inertial particle advection: Gravitational effects and horizontal diffusion. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 88, 1–11. arXiv:1309.1878v1.

[12] van Hinsberg, M.A.T., ten Thije Boonkkamp, J.H.M., Clercx, H.J.H., 2011. An efficient, second order method for the approximation of the Basset history force. Journal of Computational Physics 230, 1465–1478. 1008.0833.

[13] van Hinsberg, M.A.T., Thije Boonkkamp, J.H.M., Toschi, F., Clercx, H.J.H., 2012. On the Efficiency and Accuracy of Interpolation Methods for Spectral Codes. SIAM Journal on Scientific Computing 34, B479–B498. 1201.4060.

[14] Joly, A., Moulin, F., Violeau, D., Astruc, D., 2012. Diffusion in grid turbulence of isotropic macro-particles using a Lagrangian stochastic method: Theory and validation. Physics of Fluids 24, 1–25.

[15] Loth, E., 2000. Numerical approaches for motion of dispersed particles, droplets and bubbles. Progress in Energy and Combustion Science 26, 161–223.

[16] Lovalenti, P.M., Brady, J.F., 1993. The hydrodynamic force on a rigid particle undergoing arbitrary time-dependent motion at small Reynolds number. Journal of Fluid Mechanics 256, 561–605.

[17] Machielsen, M., 2015. Enhanced settling velocity of small particles in homogeneous isotropic turbulence. Ph.D. thesis.

[18] Magnaudet, J., Rivero, M., Fabre, J., 1995. Accelerated flows past a rigid sphere or a spherical bubble. Part 1. Steady straining flow. Journal of Fluid Mechanics 284, 97–135.

[19] Maxey, M.R., 1987. The motion of small spherical particles in a cellular flow field. Physics of Fluids 30, 1915–1928.

[20] Maxey, M.R., Riley, J.J., 1983. Equation of motion for a small rigid sphere in a nonuniform flow. Physics of Fluids 26, 883–889. arXiv:1011.1669v3.

[21] Mei, R., Adrian, R.J., 1992. Flow past a sphere with an oscillation in the free-stream velocity and unsteady drag at finite Reynolds number. Journal of Fluid Mechanics 237, 323–341.

[22] Moreno-Casas, P.A., Bombardelli, F.A., 2016. Computation of the Basset force: recent advances and environmental flow applications. Environmental Fluid Mechanics 16, 193–208.

[23] Mukherjee, D., Padilla, J., Shadden, S.C., 2016. Numerical investigation of fluid–particle interactions for embolic stroke. Theoretical and Computational Fluid Dynamics 30, 23–39.

[24] Nizkaya, T., Angilella, J.R., Buès, M.A., 2014. Inertial focusing of small particles in wavy channels: Asymptotic analysis at weak particle inertia. Physica D: Nonlinear Phenomena 268, 91–99.

[25] Nocedal, J., Wright, S., 2006. Numerical Optimization. Springer Series in Operations Research and Financial Engineering, Springer New York.

[26] Olivieri, S., Picano, F., Sardina, G., Iudicone, D., Brandt, L., 2014. The effect of the Basset history force on particle clustering in homogeneous and isotropic turbulence. Physics of Fluids 26, 041704. arXiv:1401.5309v2.

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

[28] Tao, T., . Texts and Readings in Mathematics 38. Springer Singapore.

[29] Wakaba, L., Balachandar, S., 2007. On the added mass force at finite Reynolds and acceleration numbers. Theoretical and Computational Fluid Dynamics 21, 147–153.

Appendix A. Alternative expression for the history force

We will first proof the relationship for kernels that are bounded in $[t_0, t]$. We have that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{t_0}^{t} K(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau &= K(0)\,\mathbf{f}(t) + \int_{t_0}^{t}\frac{\mathrm{d}}{\mathrm{d}t}K(t-\tau)\,\mathbf{f}(\tau)\,\mathrm{d}\tau \\ &= K(0)\,\mathbf{f}(t) - \int_{t_0}^{t}\frac{\mathrm{d}}{\mathrm{d}\tau}\left(K(t-\tau)\,\mathbf{f}(\tau)\right)\,\mathrm{d}\tau + \int_{t_0}^{t} K(t-\tau)\,\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{f}(\tau)\,\mathrm{d}\tau \\ &= K(t-t_0)\,\mathbf{f}(t_0) + \int_{t_0}^{t} K(t-\tau)\,\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{f}(\tau)\,\mathrm{d}\tau\end{aligned} \tag{A.1}$$

Now, when K is not defined at $\tau = t$, as in the case of the Basset kernel, the derivation above must be altered. We proceed by constructing a sequence of kernels that limit at the singular kernel to derive an analogous result. Consider

$$K_n : [t_0, \infty) \to \mathbb{R}, \quad t \mapsto K\left(t + \frac{1}{n}\right) \tag{A.2}$$

where n is a positive integer. The sequence of K_n for all positive integers converges pointwise to the desired kernel K. The result we are after will easily follow for K if we can move the limit sign from the integrand on the LHS expression in (A.1) to outside the derivative, as we will show. Note that we are not interested in validity of the formula at exactly $t = t_0$, since the impulse due to this infinite value at the initial time is zero anyway. Thus, mathematically, we want to show that for any $t > t_0$

$$\lim_{n\to\infty}\frac{\mathrm{d}}{\mathrm{d}t}\int_{t_0}^{t} K_n(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau = \frac{\mathrm{d}}{\mathrm{d}t}\int_{t_0}^{t}\lim_{n\to\infty} K_n(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \tag{A.3}$$

First, the limit can me moved outside the integral by the Dominated Convergence theorem. All we must prove is that there exists an integrable function that is greater or equal to all the integrands in the sequence. But

$$\left|K_n(t-\tau)\mathbf{f}(\tau)\right| \le K(t-\tau)\left\|\mathbf{f}\right\|_\infty \tag{A.4}$$

and the RHS is an integrable function, because

$$\int_{t_0}^{t} K\left\|\mathbf{f}\right\|_\infty\,\mathrm{d}\tau = 2\left\|\mathbf{f}\right\|_\infty\sqrt{t-t_0} \tag{A.5}$$

On the other hand, to show that the limit can be moved outside the derivative as well, we use the following theorem [28, Theorem 3.7.1]:

Theorem 1. *Let F_n define a sequence of differentiable functions in a closed interval $\mathrm{I} = [t_1, t_2]$, for $n \geq 1$, and let F_n' be the corresponding sequence of derivatives, also defined in* I. *Suppose that F_n' converges uniformly to some function G, also defined in* I. *Suppose also that there exists a point t_3 where the limit $\lim_{n\to\infty} F_n(t_3)$ exists. Then $\{F_n\}$ converges uniformly to a differentiable function F, and its derivative is G.*

We want to apply the theorem to the sequence of functions

$$F_n(t) : \mathrm{I} \to \mathbb{R}, \quad t \mapsto \int_{t_0}^{t} K_n(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \tag{A.6}$$

where t_1 is an arbitrary number in I. We have already shown that

$$\lim_{n\to\infty} F_n = \int_{t_0}^{t} K(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau =: F(t) \tag{A.7}$$

To particularize Theorem 1 to these choices of F and F_n it is enough to show that the following hold:

a) Each F_n is differentiable in I.

b) There exists a t_3 in I, such that

$$\lim_{n\to\infty} \int_{t_0}^{t_3} K_n(t_3-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \tag{A.8}$$

exists.

c) The sequence $\left\{F_n'\right\}$ converges uniformly to some function, G, defined in I.

The requirement a) follows immediately by the differentiability of the integrands. The requirement b) follows from the existence of the Basset force of all $t > t_0$. Finally, let us consider

$$\begin{aligned} \left\|F_m' - F_n'\right\| &= \left\| \mathbf{f}(t_0)\left(K_m(t-t_0) - K_n(t-t_0)\right) + \int_{t_0}^{t} \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\tau}\left(K_m(t-\tau) - K_n(t-\tau)\right)\,\mathrm{d}\tau \right\| \leq \\ &\leq \|\mathbf{f}\|_{W^{1,\infty}} \left| \frac{1}{\sqrt{t-t_0+\frac{1}{m}}} - \frac{1}{\sqrt{t-t_0+\frac{1}{n}}} + 2\left(\sqrt{t-t_0+\frac{1}{m}} - \sqrt{t-t_0+\frac{1}{n}} + \sqrt{\frac{1}{m}} - \sqrt{\frac{1}{n}}\right)\right| \end{aligned} \tag{A.9}$$

which tends to zero as m and n tend to zero (assuming the derivative of $\mathbf{f}$ to be well behaved). So the sequence is uniformly Cauchy and, thus, uniformly convergent to some function G, defined in I. We have proven that c) also holds. We can now apply Theorem 1 and write

$$\begin{aligned} F'(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\int_{t_0}^{t} K(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau = G(t) = \lim_{n\to\infty} F_n'(t) = \\ &= \lim_{n\to\infty} K_n(t-t_0)\mathbf{f}(t_0) + \int_{t_0}^{t} K_n(t-\tau)\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}(\tau)\,\mathrm{d}\tau \\ &= K(t-t_0)\mathbf{f}(t_0) + \int_{t_0}^{t} K(t-\tau)\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}(\tau)\,\mathrm{d}\tau \end{aligned} \tag{A.10}$$

for all $t \in I$, where in the last equality we have used the Dominated Convergence theorem again to move the limit under the integral sign.

In summary, we have proved that the formula is valid in an interval $I = [t_1, t_2]$, as long as $t_1 > t_0$; thus proving Equation (4), also for the Basset kernel.

Appendix B. Error bound for the kernel approximation

Let us define

$$e_K \coloneqq K_B - K \tag{B.1}$$

We want to establish an upper bound for (14). Ignoring the constant C_B, we have that the RHS of this equation is

$$\begin{aligned} I &\coloneqq \left| e_K(t-t_0)\mathbf{f}(t_0) + \int_{t_0}^{t-t_w} e_K(t-\tau)\frac{\mathrm{d}\mathbf{f}(\tau)}{\mathrm{d}\tau}\,\mathrm{d}\tau \right| = \\ &= \left| e_K(t-t_0)\mathbf{f}(t_0) + \int_{t_0}^{t-t_w} \frac{\mathrm{d}}{\mathrm{d}\tau}\left(e_K(t-\tau)\mathbf{f}(\tau)\right) - \mathbf{f}(\tau)\frac{\mathrm{d}}{\mathrm{d}\tau}e_K(t-t_0)\,\mathrm{d}\tau \right| = \\ &= \left| e_K(t_w-t_0)\mathbf{f}(t-t_w) - \int_{t_0}^{t-t_w} \mathbf{f}(\tau)\frac{\mathrm{d}}{\mathrm{d}\tau}e_K(t-t_0)\,\mathrm{d}\tau \right| \end{aligned} \tag{B.2}$$

Applying the change of variables $\tilde{\tau} = (t-\tau)/t_w$, we obtain

$$I = \frac{1}{\sqrt{t_w}} \left| \tilde{e}_K(1)\mathbf{f}(t-t_w) + \int_1^{\tilde{t}_0} \mathbf{f}(t-t_w\tilde{\tau})\frac{\mathrm{d}}{\mathrm{d}\tilde{\tau}}\tilde{e}_K(\tilde{\tau})\,\mathrm{d}\tilde{\tau} \right| \tag{B.3}$$

where $\tilde{t}_0 = (t-t_0)/t_w$ and

$$\tilde{e}_K = K_B - \tilde{K} \tag{B.4}$$

with $\tilde{K}$ defined as

$$\tilde{K}(\tilde{\tau}) = \begin{cases} K_B(\tilde{\tau}) & \text{if } \tilde{\tau} \le t_w \\ \tilde{K}_T(\tilde{\tau}) = \sum_{i=1}^m a_i \tilde{K}_i(\tilde{\tau}) & \text{if } \tilde{\tau} > t_w \end{cases} \tag{B.5}$$

and

$$\tilde{K}_i(\tilde{\tau}) = \alpha \left(\frac{t_i}{t_w}\right) \mathrm{e}^{\beta\left(\frac{t_i}{t_w}\right)\tilde{\tau}} \quad i = 1, \ldots, m \tag{B.6}$$

In other words, $\tilde{K}$ is obtained from K by substituting all the t_i by their normalized counterparts. Now, from (B.3) and (B.1) we can immediately write

$$I \le \frac{\|\mathbf{f}\|_\infty}{\sqrt{t_w}} \left(\left| K_B(1) - \tilde{K}(1) \right| + \int_1^{\tilde{t}_0} \left| \frac{\mathrm{d}}{\mathrm{d}\tau}\left(K_B(\tau) - \tilde{K}(\tau)\right) \right| \mathrm{d}\tau \right) \tag{B.7}$$

where we have changed the dummy integration variable back to τ.

Appendix C. Algorithmic details

Daitche method formulae

By following the steps outlined in Section 2.2 using Lagrange polynomials of a particular order, one obtains particular versions of the method. The generic quadrature formula for any order reads

$$\int_{t_0}^{t} \frac{\boldsymbol{f}(\tau)}{\sqrt{t-\tau}}\,\mathrm{d}\tau \approx \sqrt{h} \sum_{j=0}^{n} \mu_j^n \boldsymbol{f}(\tau_{n-j}) \tag{C.1}$$

where $\tau_i = t_0 + ih$ (h is the time step) and where the use of an extra index indicating the polynomial order of the interpolation has been avoided. Instead, we use Daitche's nomenclature, that is we replace μ_j^n with α_j^n (first order), with β_j^n (second order) when particularizing. The formulas for the α_j^n and β_j^n can be found in the paper by [6].

Tail contribution formulae

We must discretize the term

$$\mathbf{F}_{d,i}(t) = \int_{t-\Delta t-t_w}^{t-t_w} K_i(t-\tau)\mathbf{f}(\tau)\,\mathrm{d}\tau \tag{C.2}$$

The integral on the right-hand side of can be approximately computed using the same technique as in Daitche's method, since the kernel K_i, now of exponential form, also leads to analytically computable integrals when convoluted with polynomials. This was, in fact, the approach followed by [12], who considered the first-order-polynomial version. The first-order version of C.1. The result is

$$\boldsymbol{F}_{d,i}(t) = \frac{\mathrm{e}^{\beta t_w}}{\beta^2 h}\left(\boldsymbol{f}(\tau_{-1})\left(\mathrm{e}^{\beta h} - h\beta - 1\right) + \boldsymbol{f}(\tau_0)\left(\mathrm{e}^{\beta h}(\beta h - 1) + 1\right)\right) + \mathcal{O}(h^2) \tag{C.3}$$

where now $t - t_w = t_0$ and where $\tau_{-i} = t - t_w - ih$. (we assume here that t_w is taken as multiple of h) and where the explicit dependence of $\beta(t_i)$ on t_i has been omitted for brevity. This formula is equivalent to the one presented in [12].

The extension to order two can also be done in the same manner. However, the resulting formula is numerically very unstable due to cancellation errors and thus we followed an alternative path. Since the kernels K_i are well behaved around $t_n - t_w$, it is possible to use a standard quadrature method there. By interpolating, not only $\boldsymbol{f}$, but the product $K_i\boldsymbol{f}$, with second-order polynomials one obtains the following quadrature rule

$$\begin{aligned}\boldsymbol{F}_{d,i}(t) = \frac{\mathrm{e}^{\beta t_w}}{\beta^3 h^2}\Bigg(&\boldsymbol{f}_i(\tau_{-1})\left(\mathrm{e}^{\beta h}(2 - 3\beta h + 2\beta^2 h t^2) + \beta h - 2\right)\\ &+\boldsymbol{f}_i(\tau_0)\left(-4\mathrm{e}^{\beta h}(1-\beta h) - 2\beta^2 h^2 + 4\right) + \boldsymbol{f}_i(\tau_1)\left(\mathrm{e}^{\beta h}(2-\beta h) - \beta^2 h^2 - 2\right)\Bigg) + \mathcal{O}(h^3)\end{aligned} \tag{C.4}$$

where $\boldsymbol{f}_i(\tau) \coloneqq K_i(\tau_n - \beta)\boldsymbol{f}(\tau)$.

Appendix D. Quadratic character of the I_{2tH} problem

In this Appendix, we outline the quadratic character of the minimization problem when I_{2tH} bound is taken as the cost function. The $\tilde{t}_i$ values of I_{2tH} bound must be given and we represent them by $\tilde{T}_i$ in I_{2tH} as

$$I_{2tH}(a_i) = I_{2t}(a_i, \tilde{T}_i) = e_K(1, a_i, \tilde{T}_i)^2 + \int_1^\infty \tilde{\tau}\left(\frac{\partial \mathrm{e_K}(\tilde{\tau}, \mathrm{a_i}, \tilde{\mathrm{T}}_\mathrm{i})}{\partial \tilde{\tau}}\right)^2 \mathrm{d}\tilde{\tau}. \tag{D.1}$$

The kernel error with respect to a_i can be written as

$$e_K(\tilde{\tau}, a_i, \tilde{t}_i) = K_B(\tilde{\tau}) - \tilde{K}(\tilde{\tau}, a_i, \tilde{t}_i) = \frac{1}{\sqrt{\tilde{\tau}}} - \sum \frac{a_i}{\sqrt{\tilde{t}_i}} e^{\frac{1}{2}(1-\frac{\tilde{\tau}}{\tilde{t}_i})} = K_B(\tilde{\tau}) - \sum \tilde{K}(\tilde{\tau}, 1, \tilde{t}_i)a_i. \tag{D.2}$$

Thus, the first term of I_{2tH} bound can be expressed, using Einstein notation, as

$$\begin{aligned} e_K(1, a_i, \tilde{T}_i)^2 &= \left(K_B(1) - \tilde{K}(1,1,\tilde{t}_i)a_i\right)^2 \\ &= a_i\left(\tilde{K}(1,1,\tilde{t}_i)\tilde{K}(1,1,\tilde{t}_j)\right)a_j - 2K_B(1)\tilde{K}(1,1,\tilde{t}_j)a_j + K_B^2(1). \end{aligned} \tag{D.3}$$

By defining the design variables as $x = a_i$, the $A_{ij}^{(1)}$ matrix, the $b_j^{(1)}$ vector and $c^{(1)}$ as

$$\begin{aligned} A_{ij}^{(1)} &= 2\tilde{K}(1,1,\tilde{t}_i)\tilde{K}(1,1,\tilde{t}_j) \\ b_j^{(1)} &= 2K_B(1)\tilde{K}(1,1,\tilde{t}_j) \\ c^{(1)} &= K_B^2(1) \end{aligned} \tag{D.4}$$

the first term of I_{2tH} is readily rewritten in matrix notation, as

$$e_K(1, a_i, \tilde{T}_i)^2 = \frac{1}{2} x A^{(1)} x - b^{(1)} x + c^{(1)}. \tag{D.5}$$

Similarly, the second term of I_{2tH} has the following form

$$\begin{aligned} \int_1^\infty \tilde{\tau} \left(\frac{\partial \mathrm{e_K}(\tilde{\tau}, \mathrm{a_i}, \tilde{\mathrm{T}}_\mathrm{i})}{\partial \tilde{\tau}} \right)^2 \mathrm{d}\tilde{\tau} &= \int_1^\infty \tilde{\tau} \left(\frac{\partial K_B(\tilde{\tau})}{\partial \tilde{\tau}} - \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_i)}{\partial \tilde{\tau}} a_i \right)^2 \mathrm{d}\tilde{\tau} = a_i \left(\int_1^\infty \tilde{\tau} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_i)}{\partial \tilde{\tau}} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_j)}{\partial \tilde{\tau}} \mathrm{d}\tilde{\tau} \right) a_j \\ &= -\left(2 \int_1^\infty \tilde{\tau} \frac{\partial K_B(\tilde{\tau})}{\partial \tilde{\tau}} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_j)}{\partial \tilde{\tau}} \mathrm{d}\tilde{\tau} \right) a_j + \int_1^\infty \tilde{\tau} \left(\frac{\partial K_B(\tilde{\tau})}{\partial \tilde{\tau}} \right)^2 \mathrm{d}\tilde{\tau}. \end{aligned} \tag{D.6}$$

Again defining $A_{ij}^{(t)}$ matrix, the $b_j^{(t)}$ vector and $c^{(t)}$ as

$$\begin{aligned} A_{ij}^t &= 2 \int_1^\infty \tilde{\tau} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_i)}{\partial \tilde{\tau}} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_j)}{\partial \tilde{\tau}} \mathrm{d}\tilde{\tau} \\ b_j^{(t)} &= 2 \int_1^\infty \tilde{\tau} \frac{\partial K_B(\tilde{\tau})}{\partial \tilde{\tau}} \frac{\partial \tilde{K}(\tilde{\tau}, 1, \tilde{t}_j)}{\partial \tilde{\tau}} \mathrm{d}\tilde{\tau} \\ c^{(t)} &= \int_1^\infty \tilde{\tau} \left(\frac{\partial K_B(\tilde{\tau})}{\partial \tilde{\tau}} \right)^2 \mathrm{d}\tilde{\tau} \end{aligned} \tag{D.7}$$

The second term of the I_{2tH} bound can be compactly expressed as

$$\int_1^\infty \tilde{\tau} \left(\frac{\partial \mathrm{e_K}(\tilde{\tau}, \mathrm{a_i}, \tilde{\mathrm{T}}_\mathrm{i})}{\partial \tilde{\tau}} \right)^2 \mathrm{d}\tilde{\tau} = \frac{1}{2} x A^{(t)} x - b^{(t)} x + c^{(t)}. \tag{D.8}$$

Collecting the first and the second terms, we obtain the final expression for the I_{2tH} bound as

$$I_{2tH} = \frac{1}{2} x \left(A^{(1)} + A^{(t)} \right) x - \left(b^{(1)} + b^{(t)} \right) x + \left(c^{(1)} + c^{(t)} \right) \tag{D.9}$$

which stands for a standard quadratic minimization as

$$I_{2tH} = \frac{1}{2} x A x - b x. \tag{D.10}$$

where A and b are defined as $A = A^{(1)} + A^{(t)}$ and $b = b^{(1)} + b^{(t)}$. Note that $c = c^{(1)} + c^{(t)}$ is suppressed from the expression since it does not play any role in the minimization problem.

Appendix E. Optimal a_i and t_i values

We provide the optimal pairs a_i and $\tilde{t}_i$ for each of the cost functions I_1 (Table E.2) and I_{2t} (Table E.3). Note the resulting increasing distance between successive $\tilde{t}_i$ values, as van Hinsberg et al. [12] suggested to use. Note also that we allow in $m = 1$ the possibility of having $a_1 > 1$.

$m=1$		$m=2$		$m=3$		$m=4$		$m=5$	
a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$
1.046347992	1.581186674	0.566192817	0.717656182	0.440072204	0.482318894	0.374397988	0.365083559	0.3450551877	0.3227320427
		0.864298391	8.925153279	0.538287204	3.324763126	0.421322343	1.820334739	0.3762685526	1.4017593843
				0.807797346	38.928376132	0.517872275	11.809488351	0.4383511621	7.3543952717
						0.761539469	127.109159354	0.5502868981	52.9058339347
								0.7701813938	699.4337431732

$m=6$		$m=7$		$m=8$		$m=9$		$m=10$	
a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$
0.3227460255	0.2894856389	0.2931405176	0.2413624327	0.2718360249	0.2192620346	0.2570818336	0.1878604572	0.2520642358	0.1878604572
0.3446901326	1.1312690586	0.3053190176	0.8199848671	0.2685924185	0.662026818	0.2610118588	0.5420260992	0.254913066	0.5306382498
0.3924441164	5.1207861657	0.3394616674	3.0838532791	0.2871214552	2.0706383247	0.2799238451	1.6534881587	0.2638832071	1.5524873935
0.471576099	29.6345412934	0.3924532926	13.8047974118	0.3249589764	7.2825402363	0.3051985477	5.5204876302	0.2666445191	4.6517443725
0.5990063177	256.64908268	0.4794140412	80.9779742728	0.3805886345	31.0062809826	0.3418149337	20.8847203692	0.2806268115	14.2413555446
0.7695849793	4254.1241751139	0.5546383969	696.8320792921	0.4469592071	169.6857783353	0.3892337642	93.9005719593	0.344914608	50.7413819742
		0.6207864425	6133.2449027098	0.5474439544	1226.001409491	0.4655655296	532.1532341216	0.4566204962	263.7561507819
				0.7637048975	17271.9375778519	0.6107696402	4683.3937018005	0.5663046247	2146.211201895
						0.784623916	93277.7129340798	0.6253574036	26744.590748687
								0.6932526975	348322.670028861

Table E.2: a_i and t_i optimal values for I_1 cost function.

$m=1$		$m=2$		$m=3$		$m=4$		$m=5$	
a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$
0.9384724434	1.4300340551	0.5470597552	0.6666835275	0.430797005	0.4521461414	0.3714051613	0.3505056162	0.3335736291	0.2904610289
		0.8449767491	8.3424872407	0.5319402016	3.0597097311	0.4221306386	1.7525741335	0.3629331173	1.203691574
				0.8046471493	36.769402335	0.5248827638	11.6528756138	0.4197252519	5.9370324806
						0.7814317902	136.8864124598	0.520201698	39.1450598115
								0.7661038702	452.8226228869

$m=6$		$m=7$		$m=8$		$m=9$		$m=10$	
a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$	a_i	$\tilde{t}_i$
0.3065928563	0.2504309713	0.2869667584	0.2229567355	0.2695926115	0.1998084724	0.2560766303	0.1826244466	0.2467020831	0.1711374102
0.3243480187	0.9103056758	0.297777436	0.7379950193	0.2751628513	0.6094217589	0.2580812359	0.5231166336	0.2464749444	0.4695384557
0.3615932545	3.7204976994	0.3249218804	2.6583099103	0.2954155007	1.9746292865	0.2739535236	1.5665809982	0.2597178682	1.3336047234
0.418122689	18.2727422761	0.3631687423	10.9237321521	0.3228159111	7.0527307887	0.2950977014	5.0639463936	0.2773405882	4.038729849
0.5168085735	119.760302387	0.420482447	54.149026921	0.3602702642	28.6942745173	0.3224941082	18.0664363914	0.2995010019	13.2686834339
0.7551149413	1369.9016377844	0.5207711634	360.6375769122	0.4159293673	140.1890961279	0.3598005017	73.4054449448	0.3282822047	48.3505553197
		0.7554318595	4254.1243411105	0.5121568839	911.2555045811	0.4151331109	357.9494752882	0.3678821811	202.2013044128
				0.7402280446	10263.3419763251	0.5104760265	2319.7684648904	0.4276240337	1029.0899279619
						0.7348997012	25980.6116922192	0.5335800139	7177.8752909387
								0.7652665389	93277.7373733731

Table E.3: a_i and t_i optimal values for I_{2t} cost function.