ON THE EFFICIENCY AND ACCURACY OF INTERPOLATION METHODS FOR SPECTRAL CODES[*]

M. A. T. VAN HINSBERG[†], J. H. M. TEN THIJE BOONKKAMP[‡], F. TOSCHI[§], AND H. J. H. CLERCX[¶]

Abstract. In this paper a general theory for interpolation methods on a rectangular grid is introduced. By the use of this theory an efficient B-spline-based interpolation method for spectral codes is presented. The theory links the order of the interpolation method with its spectral properties. In this way many properties like order of continuity, order of convergence, and magnitude of errors can be explained. Furthermore, a fast implementation of the interpolation methods is given. We show that the B-spline-based interpolation method has several advantages compared to other methods. First, the order of continuity of the interpolated field is higher than for other methods. Second, only one FFT is needed, whereas, for example, Hermite interpolation needs multiple FFTs for computing the derivatives. Third, the interpolation error almost matches that of Hermite interpolation, a property not reached by other methods investigated.

Key words. interpolation, B-spline, three-dimensional, Hermite, Fourier, spline

AMS subject classifications. 65T40, 65D05

DOI. 10.1137/110849018

1. Introduction. In recent years many studies on the dynamics of inertial particles in turbulence have focused on the Lagrangian properties; see the review by Toschi and Bodenschatz [1]. For particles in turbulence, but also in many other applications in fluid mechanics, interpolation methods play a crucial role as fluid velocities, rate of strain, and other flow quantities are generally not available at the location of the particles, while these quantities are needed for the integration of the equations of motion of the particles.

When a particle is small, spherical, and rigid its dynamics in nonuniform flow is governed by the Maxey–Riley (MR) equation [2]. An elaborate overview of the different terms in the MR equation and their numerical implementation can be found in the paper by Loth [3] and a historical account was given in a review article by Michaelides [4]. The evaluation of the hydrodynamic force exerted on the particles requires knowledge of the fluid velocity, its time derivative, and gradients at the particle positions and turns out to be rather elaborate. First, the Basset history

†Department of Physics, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands (M.A.T.v.Hinsberg@tue.nl).

‡Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands (tenthije@win.tue.nl).

§CNR, Istituto per le Applicazioni del Calcolo, Via dei Taurini 19, 00185 Rome, Italy. Current address: Departments of Physics and Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands (F.Toschi@tue.nl).

¶Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands. Current address: Department of Physics, Eindhoven University of Technology, PO Box 513, 5600MB Eindhoven, The Netherlands (H.J.H.Clercx@tue.nl).

force is computationally very expensive. However, a significant reduction of cpu-time can be obtained by fitting the diffusive kernel of the Basset history force with exponential functions, as recently shown by Van Hinsberg, ten Thije Boonkkamp, and Clercx [5]. Second, the interpolation step itself can be very time consuming and memory demanding. Especially for light particles, which have a mass density similar to the fluid density (which is, for example, sediment transport in estuaries and phytoplankton in oceans and lakes), most terms in the MR equation cannot be ignored and therefore also the first derivatives of the fluid velocity are needed [6]. For this reason simulations of light particles are computationally expensive, while simulations of heavy particles are less expensive. In order to achieve convergence of the statistical properties (probability distribution functions, correlation functions, multiparticle statistics, particle distributions) many particles are needed, and this calls for fast and accurate interpolation methods. Therefore, our aim is to reduce the computation time for the evaluation of the trajectories of light particles substantially and make the algorithm competitive with the fast algorithms for the computation of trajectories of heavy particles in turbulence.

The incompressible Navier–Stokes equations are used to describe the turbulent flow field. In turbulence studies the Navier–Stokes equations are often solved by means of dealiased pseudospectral codes because of the advantage of exponential convergence of the computed flow variables. Therefore, we will focus here on interpolation methods for spectral codes. Yeung and Pope [7] and Balachandar and Maxey [8] have investigated such kinds of methods already in the late 1980s.

There are many interpolation methods available [9]. We are interested in those interpolation methods which are characterized by the following properties. First, the method must be accurate, and thus we need a high order of convergence. Second, the interpolant must have a high order of continuity C^p, with p the order of continuity. Third, the method must be fast, i.e., computationally inexpensive. A very simple interpolation method is linear interpolation. This method is very fast, but unfortunately this method is relatively inaccurate and has a low order of convergence. High order of convergence can be reached by employing Lagrange interpolation [10]. This interpolation method has the drawback that it still has a low order of continuity for the interpolant. A solution for this problem is Catmull–Rom splines [11]. Here, the interpolant has a higher order of continuity, but the order of convergence has decreased. A method that has both a high order of convergence and a high order of continuity is Hermite interpolation [12, p. 327]. The major disadvantage of this method is that the derivatives of the function to be interpolated are also needed, and these are calculated by additional fast Fourier transforms (FFTs), making this method computationally expensive. A remedy to this is B-spline interpolation [12, pp. 330–331], which has a high order of convergence and errors comparable with the ones of Hermite interpolation. Furthermore, this method has a higher order of continuity compared with the other methods mentioned above. Normally, the transformation to the B-spline basis is an expensive step, but by making use of the spectral code it can be executed in Fourier space, which makes it inexpensive. By executing this step in Fourier space the method can be optimized, resulting in smaller errors. We believe that the proposed combination of B-spline interpolation with a spectral code makes the method favorable over other traditional interpolation methods.

Besides exploring the advantages of B-spline interpolation we focus on necessary conditions allowing general three-dimensional (3D) interpolations to be efficiently executed as successive one-dimensional (1D) interpolations. These conditions also carry over desired properties (like order of convergence and order of continuity) from the

1D interpolation to the 3D equivalent. Further, we provide a fast, generic algorithm to interpolate the function and its derivatives using successive 1D interpolations.

We provide expressions for the interpolation errors in terms of the Fourier components. For this we use Fourier analysis where the interpolation method is represented as a convolution function. By doing this, the errors can be calculated as a function of the wave number. This gives insight into the behavior of interpolation, especially which components are dominant in the interpolation.

The present study may also be useful for many other applications. Examples include the computation of charged particles in a magnetic field [13, 14], as well as digital filtering and applications in medical imaging [9, 15]. In the latter case interpolations are used to improve the resolution of images. Many efforts have been taken to find interpolation methods with optimum qualities [9]. Still, it is a very active field of research. Besides the optimization of interpolation algorithms (accuracy, efficiency), the impact of different interpolation methods on physical phenomena like particle transport has been investigated in many studies [16, 17, 18, 19, 20, 21, 22].

In section 2 we introduce the general framework and explain some 1D interpolation methods. In section 3 the framework is generalized to 3D interpolation, and a generic algorithm is proposed for the implementation of the interpolation in section 4. A Fourier analysis of the interpolation operator is discussed in section 5. In section 6 the Fourier analysis is extended to Hermite interpolation and a proof of minimal errors is given. In section 7 our B-spline-based interpolation method is introduced, and is compared with three other methods (including Hermite interpolation) in section 8. Finally, concluding remarks are given in section 9.

2. Interpolation methods. We present a general framework to discuss the various interpolation methods. The goal of any interpolation method is to reconstruct the original function as closely as possible. Since in many applications also some derivatives of the original function are needed, we focus on reconstructing them as well. We start with 1D interpolation and subsequently, in section 3, generalize our framework to the 3D case.

Let $u(x)$ be a 1D function that needs to be reconstructed with $x \in [0, 1]$. In practice we only have the values of u on a uniform grid, with grid spacing Δx and knots at positions x_j, with $0 \leq j \leq (\Delta x)^{-1}$. After interpolation, the function $\widetilde{u}$ is obtained which is an approximation of u. Now let $\mathcal{I}$ be the interpolation operator, so $\widetilde{u} = \mathcal{I}[u]$.

When u has periodic boundary conditions, it can be expressed in a Fourier series as

$$(2.1) \qquad u(x) = \sum_{k \in \mathbb{Z}} \hat{u}_k \phi_k(x), \qquad \phi_k(x) = e^{2\pi i k x},$$

with i the imaginary unit and k the wave number. As the grid spacing is finite, a finite number of Fourier modes can be represented by the grid. From now on we consider u to have a finite number of Fourier modes, so

$$(2.2) \qquad u(x) = \sum_{k=-k_{\max}}^{k_{\max}} \hat{u}_k \phi_k(x),$$

where $k_{\max}$, related to Δx, is the maximum wave number. As we add a finite number of continuously differentiable Fourier modes ϕ_k we have $u \in C^\infty(0, 1)$, a property

which can be used when constructing the interpolation method. In principle u could be reconstructed at any point by the use of Fourier series; however, in practice this would be far too time consuming and instead an interpolation is performed. $\widetilde{\phi}_k$ is defined as the interpolant of ϕ_k, i.e., $\widetilde{\phi}_k = \mathcal{I}[\phi_k]$. We restrict ourselves to linear interpolation operators, i.e., $\mathcal{I}[\alpha_1 u_1 + \alpha_2 u_2] = \alpha_1 \mathcal{I}[u_1] + \alpha_2 \mathcal{I}[u_2]$ with $\alpha_1, \alpha_2 \in \mathbb{C}$. This property can be used to write $\widetilde{u}(x)$ as

$$(2.3) \qquad \widetilde{u}(x) = \sum_{k=-k_{\max}}^{k_{\max}} \hat{u}_k \widetilde{\phi}_k(x).$$

We focus on reconstructing u by piecewise polynomial functions of degree $N-1$. For each interval (x_j, x_{j+1}) with $0 \le j < (\Delta x)^{-1}$ we have

$$(2.4) \qquad \widetilde{u}(x) = \sum_{i=1}^{N} a_i x^{i-1} = \mathbf{a}^T \bar{\mathbf{x}}, \qquad x \in (x_j, x_{j+1}), \qquad \bar{\mathbf{x}} = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^{N-1} \end{pmatrix}.$$

Here, the vector $\mathbf{a}$ depends on the interval under consideration and $\mathbf{a}^T$ denotes the transpose of $\mathbf{a}$. The degree of the highest polynomial function for which the interpolation is still exact is denoted by n. In this way we get the restriction $n \le N-1$. We consider the reconstruction of u between the two neighboring grid points x_j and x_{j+1}. Without loss of generality we can translate and rescale x so that the interval $[x_j, x_{j+1}]$ becomes the unit interval $[0, 1]$.

For Hermite interpolation the values of $\widetilde{u}$ and of its derivatives, up to the order $N/2 - 1$ (N must be even), must coincide with those of u at $x = 0$ and $x = 1$, i.e.,

$$(2.5) \qquad \frac{\mathrm{d}^l \widetilde{u}}{\mathrm{d}x^l}(0) = \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(0), \qquad \frac{\mathrm{d}^l \widetilde{u}}{\mathrm{d}x^l}(1) = \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(1), \qquad l = 0, 1, \ldots, \frac{N}{2} - 1.$$

If the derivatives are known, then $n = N - 1$. When the derivatives are not known exactly on the grid they have to be approximated by finite difference methods, as done by Catmull–Rom splines [11]. Unfortunately, this method is less accurate than Hermite interpolation and $n = N - 2$.

The general framework will be illustrated with cubic Hermite interpolation for which $N = 4$. So the interpolation uses the function value and the first derivative in the two neighboring grid points to construct the interpolation polynomial. We have chosen this method because it is very accurate. Moreover, the second derivative, which is a piecewise linear function, gives minimal errors with respect to the L^2-norm. This property is further discussed in section 6.

First, the discrete values of u and possible derivatives which are given on the grid are indicated with the vector $\mathbf{b}$. In general we have

$$(2.6) \qquad \mathbf{b} = \mathbf{f}[u],$$

where the linear operator $\mathbf{f}$ depends on the interpolation method and maps a function onto an N-vector. Second, the coefficients a_i of the monomial basis need to be computed. Because $\mathcal{I}$ and $\mathbf{f}$ are linear operators, we can write without loss of generality

$$(2.7) \qquad \mathbf{a}^T = \mathbf{b}^T \mathbf{M}.$$

Here, $\mathbf{M}$ is the matrix that defines the interpolation method. For illustration, $\mathbf{f}$ and $\mathbf{M}$ for cubic Hermite interpolation are given by

$$(2.8) \qquad \mathbf{f}[u] = \begin{pmatrix} u(0) \\ \frac{\mathrm{d}u}{\mathrm{d}x}(0) \\ u(1) \\ \frac{\mathrm{d}u}{\mathrm{d}x}(1) \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Finally, substituting relation (2.7) in (2.4) gives

$$(2.9) \qquad \mathcal{I}[u](x) = \widetilde{u}(x) = \mathbf{a}^T \bar{\mathbf{x}} = \mathbf{b}^T \mathbf{M} \bar{\mathbf{x}}.$$

In many applications derivatives are also needed. In order to compute the lth derivative of $\widetilde{u}$, the monomial basis functions should be differentiated l times. In general this can be done by multiplying $\mathbf{a}$ by the differentiation matrix $\mathbf{D}$ l times, so

$$(2.10) \qquad \mathbf{a}^{(l)T} = \mathbf{a}^T \mathbf{D}^l, \qquad \mathbf{D} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \ddots & & & \vdots \\ 0 & 2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & N-1 & 0 \end{pmatrix},$$

where $\mathbf{a}^{(l)}$ contains the coefficients for the lth derivative, obtaining

$$(2.11) \qquad \frac{\mathrm{d}^l \widetilde{u}}{\mathrm{d}x^l}(x) = \mathbf{a}^{(l)T} \bar{\mathbf{x}} = \mathbf{b}^T \mathbf{M} \mathbf{D}^l \bar{\mathbf{x}} = \mathbf{b}^T \mathbf{M}^{(l)} \bar{\mathbf{x}},$$

with $\mathbf{M}^{(l)} = \mathbf{M}\mathbf{D}^l$. Note that the matrix $\mathbf{D}$ is nilpotent, since $\mathbf{D}^l = \mathbf{0}$ for $l \geq N$, implying that at most $N-1$ derivatives can be approximated.

In conclusion, we presented a framework that is able to describe interpolation methods, which can be used to implement the interpolation methods in a straightforward way. In section 4 it is used to generate fast algorithms for the implementation of the method.

3. 3D interpolation. In this section the 1D interpolation methods of section 2 are extended to the 3D case. Now the scalar field u depends on the vector $\mathbf{x}$ and a 3D interpolation needs to be carried out. As before, the interpolated field is given by $\widetilde{u}$ and $\mathcal{I}_3$ is the 3D equivalent of $\mathcal{I}$, so $\widetilde{u} = \mathcal{I}_3[u]$. The 3D field u can be represented by a 3D Fourier series like

$$(3.1) \qquad u = \sum_{\mathbf{k}} \hat{u}_{\mathbf{k}} \phi_{\mathbf{k}}(\mathbf{x}),$$

where $\phi_{\mathbf{k}}$ is given by

$$(3.2) \quad \phi_{\mathbf{k}}(\mathbf{x}) = e^{2\pi i \mathbf{k} \cdot \mathbf{x}} = \phi_{k_x}(x) \phi_{k_y}(y) \phi_{k_z}(z), \qquad \mathbf{k} = (k_x, k_y, k_z), \quad \mathbf{x} = (x, y, z),$$

and ϕ_k is defined by (2.1). Again we restrict ourselves to linear interpolation operators; therefore $\widetilde{u}$ can be written as

$$(3.3) \qquad \widetilde{u}(\mathbf{x}) = \sum_{\mathbf{k}} \hat{u}_{\mathbf{k}} \widetilde{\phi}_{\mathbf{k}}(\mathbf{x}),$$

with $\widetilde{\phi}_{\mathbf{k}}$ the interpolant of $\phi_{\mathbf{k}}$, i.e., $\widetilde{\phi}_{\mathbf{k}} = \mathcal{I}_3[\phi_{\mathbf{k}}]$.
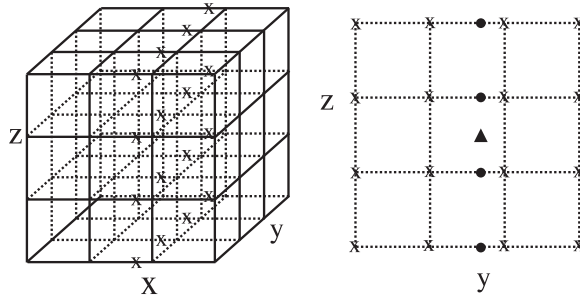
FIG. 3.1. *Graphical description of the 3D Lagrange interpolation, using three steps of 1D interpolations for the case $N = 4$. First, N^2 1D interpolations are carried out in the x-direction (crosses). Second, N interpolations are carried out in the y-direction (dots in the right figure), and from these N results finally one interpolated value is derived in the z-direction (triangle).*

The 3D interpolation for a scalar field is carried out applying three times 1D interpolations; see Figure 3.1. The interpolation consists of three steps in which the three spatial directions are interpolated one after each other. The order in which the spatial directions are interpolated does not matter. Building the 3D interpolation out of 1D interpolations saves computing time. It can be done for all interpolation methods as long as the following two conditions are met. First, the operator $\mathcal{I}_3$ must be linear. Second, the condition

$$(3.4) \qquad \widetilde{\phi}_{\mathbf{k}} \equiv \mathcal{I}_3\left[\phi_{\mathbf{k}}\right] = \mathcal{I}_3\left[\phi_{k_x}\phi_{k_y}\phi_{k_z}\right] = \mathcal{I}\left[\phi_{k_x}\right]\mathcal{I}\left[\phi_{k_y}\right]\mathcal{I}\left[\phi_{k_z}\right] = \widetilde{\phi}_{k_x}\widetilde{\phi}_{k_y}\widetilde{\phi}_{k_z},$$

must be satisfied, which is the case for almost all interpolation methods. Property (3.4) can be used to prove that properties of the 1D operator $\mathcal{I}$ carry over to the 3D operator $\mathcal{I}_3$, for example, the order of convergence and the order of continuity.

Next, relations (2.9) and (2.11) are extended to the 3D case. As before, we start with storing some values of u (given by the spectral code) and possible derivatives in the third-order tensor $\mathbf{B}$. In the same fashion as relation (2.6) one gets

$$(3.5) \qquad \mathbf{B} = \mathbf{f}_z\left[\mathbf{f}_y\left[\mathbf{f}_x[u]\right]\right],$$

where one element of tensor $\mathbf{B}$ is defined like

$$(3.6) \qquad B_{i_1 i_2 i_3} = \mathbf{f}_z\left[\mathbf{f}_y\left[\mathbf{f}_x[u]_{i_1}\right]_{i_2}\right]_{i_3}.$$

$\mathbf{f}_x$, $\mathbf{f}_y$, and $\mathbf{f}_z$ are similar to operator $\mathbf{f}$ but now working in a specified direction. For Hermite interpolation they are given by

(3.7)

$$\mathbf{f}_x[u] = \begin{pmatrix} u(0,y,z) \\ \frac{\partial u}{\partial x}(0,y,z) \\ u(1,y,z) \\ \frac{\partial u}{\partial x}(1,y,z) \end{pmatrix}, \quad \mathbf{f}_y[u] = \begin{pmatrix} u(x,0,z) \\ \frac{\partial u}{\partial y}(x,0,z) \\ u(x,1,z) \\ \frac{\partial u}{\partial y}(x,1,z) \end{pmatrix}, \quad \mathbf{f}_z[u] = \begin{pmatrix} u(x,y,0) \\ \frac{\partial u}{\partial z}(x,y,0) \\ u(x,y,1) \\ \frac{\partial u}{\partial z}(x,y,1) \end{pmatrix}.$$

The interpolation is carried out in a way similar to that sketched in Figure 3.1. Similarly to (2.9), $\widetilde{u}(\mathbf{x})$ can be represented as

$$(3.8) \qquad \mathcal{I}_3[u](\mathbf{x}) = \widetilde{u}(\mathbf{x}) = \mathbf{B} \bar{\times}_1 (\mathbf{M}\bar{\mathbf{x}}) \bar{\times}_2 (\mathbf{M}\bar{\mathbf{y}}) \bar{\times}_3 (\mathbf{M}\bar{\mathbf{z}}),$$

where $\mathbf{M}$ is still the matrix for 1D interpolation, $\bar{\mathbf{y}}$ and $\bar{\mathbf{z}}$ are defined like $\bar{\mathbf{x}}$, which is given by relation (2.4). Further, $\bar{\times}_n$ denotes the n-mode vector product [23], like

$$(3.9) \qquad \mathbf{A} = \mathbf{B} \bar{\times}_n \mathbf{f}, \qquad A_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_{\mathcal{N}}} = \sum_{i_n} B_{i_1 \cdots i_{\mathcal{N}}} f_{i_n},$$

where $\mathcal{N}$ denotes the order of tensor $\mathbf{B}$. In this way tensor $\mathbf{A}$ is one order less than tensor $\mathbf{B}$. Because we employ three of these n-mode vector products, the third-order tensor $\mathbf{B}$ reduces to a scalar. Furthermore, each of these n-mode vector products corresponds to an interpolation in one direction; see also Figure 3.1. For a general derivative one gets

$$(3.10) \qquad \frac{\partial^{i+j+k}\widetilde{u}}{\partial x^i \partial y^j \partial z^k}(\mathbf{x}) = \mathbf{B} \bar{\times}_1 \left(\mathbf{M}^{(i)}\bar{\mathbf{x}} \right) \bar{\times}_2 \left(\mathbf{M}^{(j)}\bar{\mathbf{y}} \right) \bar{\times}_3 \left(\mathbf{M}^{(k)}\bar{\mathbf{z}} \right).$$

Note that the matrix $\mathbf{M}$ does not necessarily have to be the same for the different directions x, y, and z. One could choose different interpolation methods when, for example, Chebyshev polynomials are used in one direction. In this case the grid is nonuniform in this direction, and therefore not all interpolation methods can be used.

Finally, when the scalar field $u(\mathbf{x})$ becomes a vector field $\mathbf{u}(\mathbf{x})$, the three components of $\mathbf{u}$ can be interpolated separately. This can be written in short by a fourth-order tensor $\mathbf{B}$ where the last dimension contains the three components of $\mathbf{u}$. In this way the equations for the new tensor $\mathbf{B}$ remain the same as given above.

4. Implementation. Relations (3.8) and (3.10) provide a good starting point for an efficient implementation of the interpolation. We focus on interpolating a 3D vector field $\mathbf{u}(\mathbf{x})$ and on calculating all its first derivatives (which are needed in many applications such as the computation of the trajectories of inertial particles). The matrices $\mathbf{M}$ and $\mathbf{M}^{(1)}$ only need to be computed once, which can be done prior to interpolation. Next, the vectors $\bar{\mathbf{x}}$, $\bar{\mathbf{y}}$, and $\bar{\mathbf{z}}$ have to be computed, which only needs to be done once for each location at which interpolation is performed. In Table 4.1 we keep track of all the computed quantities. Here, the computational costs for evaluating all the components is shown where one flop denotes one multiplication with one addition. We show the number of flops for the general case and for $N = 4$. The main idea is to reduce the order of the tensors as soon as possible in order to generate an efficient method.

In order to determine how efficient the algorithm is, one can compare the computational costs against a lower bound. The lower bound we use is related to the size of $\mathbf{B}$, which is $3N^3$ for a vector field $\mathbf{u}$. In order to be able to use all the information in tensor $\mathbf{B}$, $3N^3$ flops are needed. For large N one finds that the algorithm of Table 4.1 is only a factor 2 less efficient than this lower bound.

We also compare our algorithm with the one proposed by Lekien and Marsden [24], which uses Hermite interpolation with $N = 4$. Our algorithm has fewer restrictions and shows a slightly better computational performance (for $N = 4$). The algorithm of Lekien and Marsden consists of two steps. First, they calculate the coefficients for the polynomial basis. Second, the values at the desired location are calculated. They claim that their method is beneficial when the derivatives are needed or the

TABLE 4.1
Algorithm for interpolation, with an estimate of the computational costs.

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

interpolation needs to be done multiple times for the same interval, because only the second step needs to be executed multiple times. Our method does not have the first step, therefore it has no restrictions; nevertheless the computation of the values and the first derivatives is slightly faster than for Lekien and Marsden, even when considering only the second step. The total costs of their second step is bounded by $12N^3$ flops (4 times $3N^3$ flops, for the computation of the values and the first derivatives). From Table 4.1 we can conclude that our method needs less fewer for the same computations.

5. Fourier analysis. In this section the interpolation operator $\mathcal{I}$ is expressed in terms of a convolution. In this way properties of the interpolation method like the order of continuity of the interpolated field and the magnitude of the errors can be shown in the Fourier domain. We start with the interpolation of 1D functions, and subsequently it can be extended to the 3D case.

Before we start with the derivation, we rescale the variable x by dividing it by Δx, so that the new grid spacing equals unity. From now on we work with the rescaled grid where $x \in [0, m]$ and $m = (\Delta x)^{-1}$, so $x_j = j$ for $0 \le j \le m$. Furthermore we introduce the dimensionless wave number $\kappa = k\Delta x$, and ϕ_κ is similarly defined as ϕ_k; see (2.1). For Hermite interpolation the derivation is somewhat more complex because also the derivatives are used and therefore it is postponed to section 6. We focus on interpolation methods where $\mathbf{f}[u]$ contains the values of u at the N nearest grid points x_j of x with local ordering. Thus $b_j = u(x_j)$ and x_j is given by

$$(5.1) \qquad x_j = \left\lfloor x - \frac{N}{2} + j \right\rfloor, \qquad j = 1, 2, \ldots, N,$$

where $\lfloor \cdot \rfloor$ denotes the nearest lower integer. The interpolation methods can be described by the matrix $\mathbf{M}$, with elements $M_{j,i}$; see relation (2.9). This relation can also be written as

$$(5.2) \qquad \widetilde{u}(x) = \sum_{j=1}^{N} C_j\left(x - x_j\right) u\left(x_j\right),$$

with x_j defined by (5.1) and where C_j is given by

$$(5.3) \qquad C_j\left(x + \frac{N}{2} - j\right) = \begin{cases} \sum_{i=1}^{N} M_{j,i} x^{i-1} & \text{for } 0 \le x < 1, \\ 0 & \text{elsewhere.} \end{cases}$$

Relation (5.2) can be rewritten by using the sifting property of the delta function, like

$$(5.4) \qquad \widetilde{u}(x) = \sum_{j=1}^{N} C_j\left(x - x_j\right) \int_{-\infty}^{\infty} u(y)\delta\left(y - x_j\right) \mathrm{d}y.$$

This can be further reformulated by subtracting the argument of the delta function from the argument of C_j, as

$$\widetilde{u}(x) = \int_{-\infty}^{\infty} u(y) \sum_{j=1}^{N} \delta\left(y - x_j\right) C_j(x - y)\mathrm{d}y$$

$$(5.5) \qquad = \int_{-\infty}^{\infty} u(y)D(y)C(x - y)\mathrm{d}y,$$

with $C(x)$ and $D(x)$ given by

$$(5.6) \qquad C(x) = \sum_{j=1}^{N} C_j(x), \qquad D(x) = \sum_{i \in \mathbb{Z}} \delta(x - i).$$

In relation (5.5) the delta functions can be replaced by the function D, which is a train of delta functions because the functions C_j only have a support of length unity; see (5.3). Finally, the interpolation can be written like

$$(5.7) \qquad \widetilde{u} = (uD) * C,$$

with $*$ denoting the convolution product. Here, the convolution function C depends on the interpolation matrix $\mathbf{M}$; see Figure 5.1.

As a consequence of relation (5.7), if the function C is continuous up to the pth derivative, then $\widetilde{u}$ is also continuous up to the pth derivative. Even stronger, the order of continuity of the function C is equal to the order of continuity of $\widetilde{u}$. Furthermore, by the use of relation (5.7) exact interpolation can be constructed as well.[1]

In the remainder of this section we will discuss the interpolation error. Before proceeding we need to prove the following theorem.

THEOREM. $\langle e_\kappa, e_\lambda \rangle_2 = 0$ for $\kappa \ne \lambda$. Here e_κ is the error in mode κ, $e_\kappa = \widetilde{\phi}_\kappa - \phi_\kappa$; similarly, $e_\lambda = \widetilde{\phi}_\lambda - \phi_\lambda$ and $\langle \cdot \rangle_2$ is the inner product related to the L^2-norm $\| \cdot \|_2$ defined by

$$(5.8) \quad \langle f, g \rangle_2 = \int_0^m f(x)g^*(x)\mathrm{d}x, \qquad \|f\|_2^2 = \langle f, f \rangle_2 = \int_0^m f(x)f^*(x)\mathrm{d}x.$$

The asterisk (*) denotes complex conjugation.

---

[1] Exact interpolation can be accomplished by $\mathcal{F}[C](k) = 1$ for $-0.5 \le k \le 0.5$ and zero elsewhere. In this way only the original Fourier component is filtered out of the spectrum. Note that in this case C has infinite support.
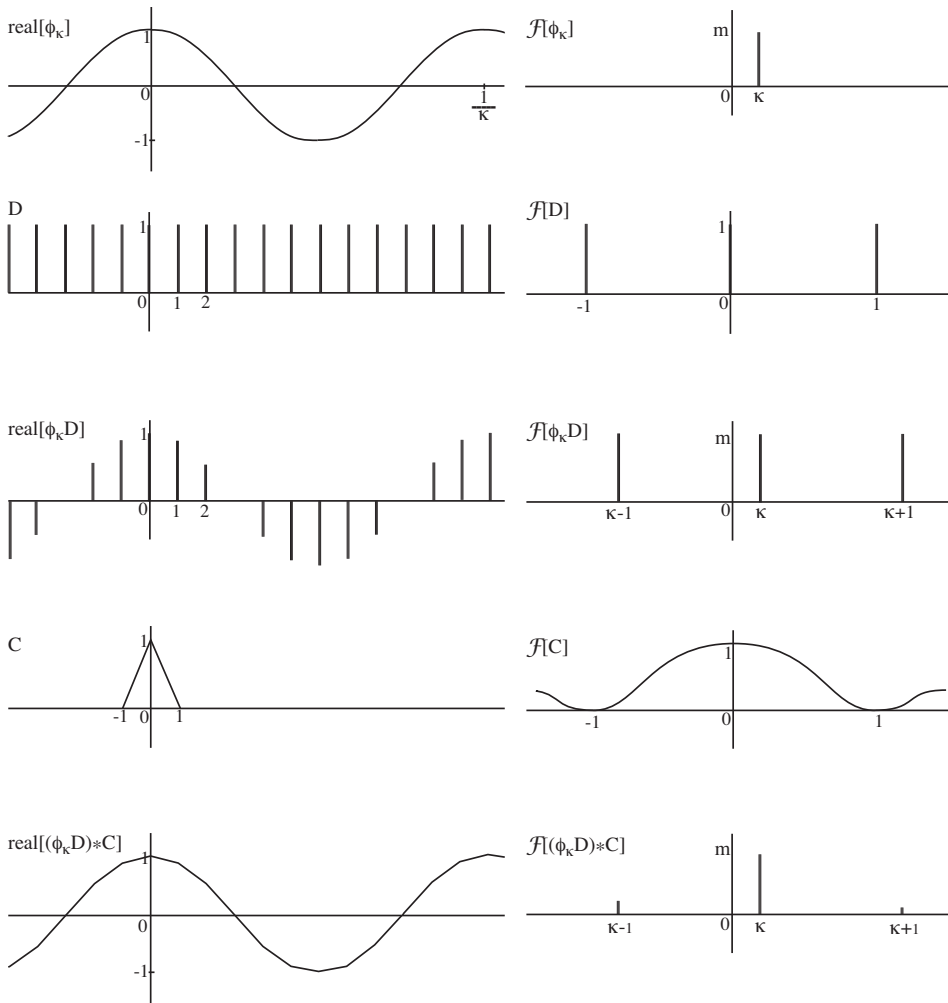
FIG. 5.1. *Sketch of linear interpolation as a convolution. The pins represent delta functions with the height equal to its prefactor. On the left side is a visualization in real space and on the right side in Fourier space.*

Proof. We start with replacing u by ϕ_κ in relation (5.7), i.e.,

$$(5.9) \qquad \widetilde{\phi}_\kappa = \mathcal{I}[\phi_\kappa] = (\phi_\kappa D) * C.$$

Second, we take the Fourier transform of $\widetilde{\phi}_\kappa$, for some fixed κ_0, i.e.,

$$\mathcal{F}\left[\widetilde{\phi}_{\kappa_0}\right](k) = \mathcal{F}\left[(\phi_{\kappa_0} D) * C\right](k) = \left(\mathcal{F}[\phi_{\kappa_0}] * \mathcal{F}[D]\right)(k)\mathcal{F}[C](k)$$

$$(5.10) \qquad = m \sum_{i\in\mathbb{Z}} \delta\big(k - (i + \kappa_0)\big)\mathcal{F}[C](i + \kappa_0),$$

which results in a train of delta functions with the prefactor given by $\mathcal{F}[C]$ (see Figure 5.1), and $\mathcal{F}[\cdot]$ denotes the Fourier transform given by

$$(5.11) \qquad \mathcal{F}[g](k) := \int_{-\infty}^{\infty} g(x)e^{-2\pi i k x}\mathrm{d}x.$$

For linear interpolation these functions are shown in Figure 5.1. Next, $\langle e_\kappa, e_\lambda \rangle_2$ can be written as $\langle e_\kappa, e_\lambda \rangle_2 = \left\langle \widetilde{\phi}_\kappa, \widetilde{\phi}_\lambda \right\rangle_2 - \left\langle \widetilde{\phi}_\kappa, \phi_\lambda \right\rangle_2 - \left\langle \phi_\kappa, \widetilde{\phi}_\lambda \right\rangle_2 + \left\langle \phi_\kappa, \phi_\lambda \right\rangle_2$. Trivially $\left\langle \phi_\kappa, \phi_\lambda \right\rangle_2 = 0$ for $\kappa \neq \lambda$. Furthermore, $\widetilde{\phi}_\kappa$ consists of a discrete set of Fourier components; see relation (5.10). Using this relation, one can show that no common Fourier components exist for $\widetilde{\phi}_\kappa$ and $\widetilde{\phi}_\lambda$ or ϕ_λ for $\kappa \neq \lambda$. Therefore $\left\langle \widetilde{\phi}_\kappa, \widetilde{\phi}_\lambda \right\rangle_2 = 0$, $\left\langle \widetilde{\phi}_\kappa, \phi_\lambda \right\rangle_2 = 0$, and $\left\langle \phi_\kappa, \widetilde{\phi}_\lambda \right\rangle_2 = 0$ for $\kappa \neq \lambda$, implying $\langle e_\kappa, e_\lambda \rangle_2 = 0$ as claimed.

COROLLARY. *The orthogonality is important to estimate errors. When the error in u is computed as $\|\widetilde{u} - u\|_2^2$, it can be rewritten as $\|\widetilde{u} - u\|_2^2 = \sum_\kappa \hat{u}_\kappa^2 \|e_\kappa\|_2^2$, which allows easy and straightforward computation of the errors.*

Next, the error in one Fourier component is calculated. In this derivation we make use of the fact that $\widetilde{\phi}_\kappa$ can be written by a sum of Fourier components; see Figure 5.1 and relation (5.10). The relative error in one Fourier component can be written as

$$\frac{\left\| \widetilde{\phi}_\kappa - \phi_\kappa \right\|_2^2}{\|\phi_\kappa\|_2^2} = \frac{\|e_\kappa\|_2^2}{m} = \frac{1}{m}\left\| -\phi_\kappa + \sum_{i \in \mathbb{Z}} \mathcal{F}[C]\,(\kappa + i)\,\phi_{\kappa+i} \right\|_2^2$$

$$(5.12) \qquad\qquad = \left(\mathcal{F}[C]\,(\kappa) - 1 \right)^2 + \sum_{i \neq 0} \left(\mathcal{F}[C]\,(\kappa + i) \right)^2.$$

From this expression one can see that the error can be computed directly from $\mathcal{F}[C]$. The same can be done for the error in the lth derivative: $e_\kappa^{(l)} = \widetilde{\phi}_\kappa^{(l)} - \phi_\kappa^{(l)}$. The idea is to take the derivatives of the individual Fourier components, which results in

$$(5.13) \qquad \frac{\left\| e_\kappa^{(l)} \right\|_2^2}{\left\| \phi_\kappa^{(l)} \right\|_2^2} = \left(\mathcal{F}[C]\,(\kappa) - 1 \right)^2 + \sum_{i \neq 0} \left(\frac{\kappa + i}{\kappa} \right)^{2l} \left(\mathcal{F}[C]\,(\kappa + i) \right)^2.$$

The extension to the 3D case is rather straightforward and is therefore not reported here. The basic idea is to create 3D functions by multiplying the 1D components; this can be done for all functions and the basic equations remain the same.

6. Hermite interpolation. In this section we extend the theory of section 5 to Hermite interpolation. We also show some special properties that hold for Hermite interpolations. We especially examine the case $N = 4$. For this case the second derivative becomes a piecewise linear function. Comparison with the actual second derivative shows that this piecewise linear function is optimal with respect to the L^2-norm.

Analogously to (5.2), Hermite interpolation with even N can be written as

$$(6.1) \qquad \widetilde{u}(x) = \sum_{j=0}^{1} \sum_{l=1}^{N/2} C_{j,l}\,(x - x_j)\, \frac{\mathrm{d}^{l-1} u}{\mathrm{d} x^{l-1}}\,(x_j),$$

where $C_{j,l}$ and x_j are given by

$$C_{j,l}(x-j) = \begin{cases} \sum_{i=1}^{N} M_{l+j\frac{N}{2},i}x^{i-1} & \text{for } 0 \le x < 1, \\ 0 & \text{elsewhere,} \end{cases} \quad l \in 1,2,\ldots,\frac{N}{2},$$

$$(6.2) \qquad x_j = \lfloor x \rfloor + j, \qquad\qquad\qquad\qquad\qquad\qquad j \in 0,1.$$

Again following steps similar to those in section 5, $\widetilde{u}(x)$ can be rewritten as

$$\widetilde{u}(x) = \sum_{j=0}^{1}\sum_{l=1}^{N/2} C_{j,l}(x-x_j) \int_{-\infty}^{\infty} \frac{\mathrm{d}^{l-1}u}{\mathrm{d}x^{l-1}}(y)\delta(y-x_j)\mathrm{d}y$$

$$(6.3) \qquad = \sum_{l=1}^{N/2} \int_{-\infty}^{\infty} \frac{\mathrm{d}^{l-1}u}{\mathrm{d}x^{l-1}}(y)D(y)C_l(x-y)\mathrm{d}y,$$

where D is given by relation (5.6) and C_l is given by $C_l(x) = C_{0,l}(x) + C_{1,l}(x)$. In short, $\widetilde{u}$ can be written as

$$(6.4) \qquad \widetilde{u} = \sum_{l=1}^{N/2} \left(\frac{\mathrm{d}^{l-1}u}{\mathrm{d}x^{l-1}}D\right) * C_l,$$

similar to relation (5.7). Here one can see that for Hermite interpolation multiple convolution functions C_l are needed which correspond to the derivatives and the function itself. Replacing u by ϕ_κ in (6.4) gives

$$(6.5) \qquad \widetilde{\phi_\kappa} = \mathcal{I}[\phi_\kappa] = (\phi_\kappa D) * \sum_{l=1}^{N/2} (2\pi i\kappa)^{l-1} C_l.$$

In this way we find an expression similar to relation (5.9), where C has to be replaced by $\sum_{l=1}^{N/2}(2\pi i\kappa)^{l-1}C_l$. In conclusion, relation (5.12) and (5.13) can still be used.

PROPERTY. *For the error in the first derivative we have the following property:*

$$(6.6) \qquad \left\langle e^{(1)},1\right\rangle_2 = 0,$$

where the inner product $\langle\cdot,\cdot\rangle_2$ is defined on the unit interval, i.e.,

$$(6.7) \qquad \langle f,g\rangle_2 = \int_0^1 f(x)g^*(x)\mathrm{d}x.$$

Furthermore the error in the lth derivative, $e^{(l)}$, is given by

$$(6.8) \qquad e^{(l)} = \frac{\mathrm{d}^l\widetilde{u}}{\mathrm{d}x^l}(x) - \frac{\mathrm{d}^l u}{\mathrm{d}x^l}(x).$$

Proof. One can rewrite part of the interpolation conditions for Hermite interpolation (2.5) in the following way

$$(6.9) \qquad \widetilde{u}(1)-\widetilde{u}(0) = u(1)-u(0) \quad\Leftrightarrow\quad \int_0^1 \frac{\mathrm{d}\widetilde{u}}{\mathrm{d}x}\mathrm{d}x = \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\mathrm{d}x.$$

Here two interpolation conditions give one new condition which is equivalent to relation (6.6).

COROLLARY. *Property* (6.6) *shows that the error in the first derivative does not have a constant component. Therefore the constant component is exact with respect to the L^2-norm.*

PROPERTY. *For the error in the second derivative in case of $N = 4$ we have*

$$(6.10) \qquad \left\langle e^{(2)}, 1 \right\rangle_2 = 0, \qquad \left\langle e^{(2)}, x \right\rangle_2 = 0.$$

Proof. One can rewrite the interpolation conditions (2.5) for $N = 4$ in the following way:

$$\widetilde{u}(1) - \widetilde{u}(0) - \widetilde{u}'(0) = u(1) - u(0) - u'(0) \Leftrightarrow \int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2} \mathrm{d}x \mathrm{d}\alpha = \int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \mathrm{d}x \mathrm{d}\alpha,$$

$$(6.11) \qquad \widetilde{u}'(1) - \widetilde{u}'(0) = u'(1) - u'(0) \Leftrightarrow \int_0^1 \frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2} \mathrm{d}x = \int_0^1 \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \mathrm{d}x.$$

The first relation in (6.10) follows immediately from the second condition in (6.11). The second relation in (6.10) is derived in the following way:

$$\int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2} \mathrm{d}x \mathrm{d}\alpha = \int_0^1 \int_0^\alpha \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \mathrm{d}x \mathrm{d}\alpha,$$

$$\alpha \int_0^\alpha \left(\frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x) \right) \mathrm{d}x \bigg|_{\alpha=0}^{\alpha=1} - \int_0^1 \left(\frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2}(\alpha) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(\alpha) \right) \alpha \mathrm{d}\alpha = 0,$$

$$(6.12) \qquad \int_0^1 \left(\frac{\mathrm{d}^2 \widetilde{u}}{\mathrm{d}x^2}(x) - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x) \right) x \mathrm{d}x = 0.$$

Here, the first step is integration by parts and the second step uses the second relation of (6.11).

COROLLARY. *Relation* (6.10) *implies that $e^{(2)}$ does not have a constant component, nor a linear component. For $N = 4$ the second derivative is a linear function, and this means that there is no better approximation in the L^2-norm of this second derivative with a piecewise linear function. This makes Hermite interpolation very interesting as a reference case, because we now have proven that the error is minimal for this case.*

7. B-spline interpolation. In this section we start with recalling B-spline interpolation. The idea is to create an interpolant that is as smooth as possible. Later it is shown how the pseudospectral code can be used to efficiently execute this interpolation method. Furthermore, the interpolation method is optimized to create small errors in the L^2-norm. We start with giving the B-spline convolution functions, after which their matrix representation is given, and finally the transformation to the B-spline basis functions is derived.

In a spectral code FFTs are applied to transform data from real space to Fourier space and backwards. These FFTs are the most expensive step in the simulation, and therefore we want to keep the number of FFTs needed minimal. This is the reason why Hermite interpolation is not a good option, since extra FFTs are needed for the computation of the derivatives. An alternative is B-spline interpolation.
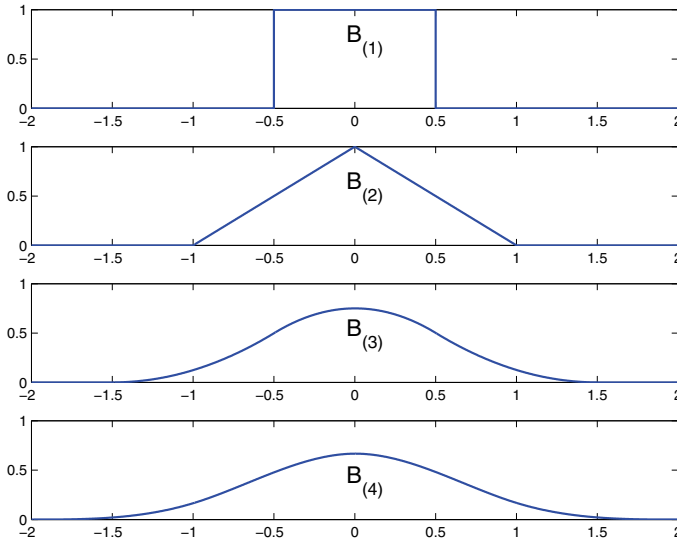
FIG. 7.1. *First four uniform B-splines functions.*

We require a high order of continuity of the interpolant. The highest order of continuity that can be obtained for the interpolant with piecewise polynomial functions of degree $N-1$ is C^{N-2}. In this way the interpolant still matches the original function $u(x)$ at the grid points x_j. Moreover, one can immediately see that $n = N-1$, where n is the highest degree of a polynomial test function for which the interpolation is still exact. This high level of continuity can be achieved by using B-spline functions [12, pp. 330–331]. The first four uniform B-spline basis functions $B_{(N)}$ are shown in Figure 7.1. These functions can be generated by means of convolutions in the following way:

$$B_{(1)}(x) = \begin{cases} 1 & \text{for } -0.5 \leq x < 0.5, \\ 0 & \text{elsewhere,} \end{cases}$$
$$B_{(2)} = B_{(1)} * B_{(1)},$$
$$B_{(3)} = B_{(2)} * B_{(1)},$$
$$\vdots$$

(7.1)
$$B_{(N)} = B_{(N-1)} * B_{(1)}.$$

These functions have the property that the Nth function is of degree $N-1$ and is C^{N-2}. Furthermore, the B-spline basis functions have local support of length N. The B-spline functions can be seen as convolution functions C introduced in section 5 and have a matrix representation. The relation between the functions $B_{(N)}$ and the matrix representation is similar to relations (5.3) and (5.6) and is given by

$$B_{(N)}(x) = \sum_{j=1}^{N} B_{(N),j}(x),$$

(7.2)
$$B_{(N),j}\left(x + \frac{N}{2} - j\right) = \begin{cases} \sum_{i=1}^{N} M_{(N),j,i} x^{i-1} & \text{for } 0 \leq x < 1, \\ 0 & \text{elsewhere.} \end{cases}$$

The matrix representation for the first four B-spline functions is as follows [25]:

$$\mathbf{M}_{(1)} = (1),$$

$$\mathbf{M}_{(2)} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix},$$

$$\mathbf{M}_{(3)} = \frac{1}{2!} \begin{pmatrix} 1 & -2 & 1 \\ 1 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix},$$

(7.3) $$\mathbf{M}_{(4)} = \frac{1}{3!} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In general we have [25]

(7.4) $$M_{(N),j,i} = \frac{1}{(N-1)!} Q_{N-1}^{N-i} \sum_{s=j}^{N} (-1)^{s-j} Q_N^{s-j} (N-s)^{N-i}, \qquad i,j = 1,2,\ldots,N,$$

with Q_n^i given by

(7.5) $$Q_n^i = \frac{n!}{i!(n-i)!} = \begin{pmatrix} n \\ i \end{pmatrix}.$$

We still need to express $u(x), x \in \mathbb{Z}$, in terms of B-spline basis functions and thus find the transform from real space to the B-spline basis. Because the inverse transform from the B-spline basis to real space is somewhat easier, we start with this transformation first. From now on we omit the subindex (N). The coefficients of the B-spline basis are called $u_B(x)$, and $u(x)$ can be derived from it by the discrete convolution $*_D$ in the following way: $u = u_B *_D B_D$. Here, B_D is given by

(7.6) $$B_D(x) = \begin{cases} B(x) & \text{for } x < \frac{m}{2}, \\ B(x-m) & \text{for } x \geq \frac{m}{2}, \end{cases} \qquad x = 0, 1, \ldots, m-1,$$

and the discrete convolution is given by

(7.7) $$(g *_D h)(x) = \sum_{y=0}^{m} g(y) h\big((x-y) \bmod m\big), \qquad x = 0, 1, \ldots, m-1.$$

Next, the inverse B_D^{-1} needs to be determined, where B_D^{-1} is defined by $B_D^{-1} *_D B_D = \delta_D$, with δ_D the discrete delta function, given by

(7.8) $$\delta_D(x) = \begin{cases} 1 & \text{for } x = 0, \\ 0 & \text{else,} \end{cases} \qquad x = 0, 1, \ldots, m-1.$$

Using the inverse B_D^{-1}, we can find $u_B(x)$ by the discrete convolution $u_B(x) = u(x) *_D B_D^{-1}(x)$.

Using a spectral code, the discrete convolution can be evaluated in Fourier space, and it reduces to a multiplication by constants $c(k)$. These multiplication constants

can be computed beforehand and no convolutions need to be evaluated. In this way
one gets

$$\mathcal{F}_D\left[u_B\right](k) = \mathcal{F}_D[u](k)\mathcal{F}_D\left[B_D^{-1}\right](k) = \frac{\mathcal{F}_D[u](k)}{\mathcal{F}_D\left[B_D\right](k)}$$

$$(7.9) \qquad\qquad\qquad = c(k)\mathcal{F}_D[u](k),$$

where the discrete Fourier transform $\mathcal{F}_D$ is given by

$$(7.10) \qquad \mathcal{F}_D[f](k) = \sum_{x=0}^{m-1} f(x)e^{-2\pi i x k/m}, \qquad k = 0, 1, \ldots, m-1.$$

The values of $c(k)$ can be determined in a straightforward manner as suggested above
using $\mathcal{F}_D[B_D]$, but a more optimal choice for $c(k)$ can be made in the following way.
We minimize the L^2-norm of the error, and for this we use relation (5.12) and require

$$(7.11) \qquad\qquad \frac{\mathrm{d}}{\mathrm{d}c(k)} \left\| \phi_\kappa - c(k)\widetilde{\phi}_\kappa \right\|_2^2 = 0,$$

with $\kappa = k\Delta x$. This implies

$$(7.12) \qquad\qquad c(k) = \frac{\mathcal{F}[B](\kappa)}{\sum_{i\in\mathbb{Z}}\left(\mathcal{F}[B](\kappa+i)\right)^2}.$$

In three dimensions (7.9) becomes

$$(7.13) \qquad\qquad \mathcal{F}_D[u_B](\mathbf{k}) = c\left(k_x\right)c\left(k_y\right)c\left(k_z\right)\mathcal{F}_D[u](\mathbf{k}).$$

Concluding, we propose an interpolation method for pseudospectral codes where
the interpolation matrix $\mathbf{M}$ is given by (7.4). Further a multiplication in Fourier space
is executed like (7.13) where the coefficients can be determined from (7.12). The
coefficients can be computed beforehand, and therefore no extra FFTs are needed,
making this method very efficient.

8. Comparison of the interpolation methods. In this section four different
interpolation methods are compared.[2] The criteria we are interested in are the fol-
lowing. First, the method must be fast, which is needed because many interpolations
will usually be carried out. Second, as we are using a spectral code, exponential con-
vergence is expected, and in order to meet this accuracy the interpolation methods
must have high order of convergence. Furthermore, as the original function is C^∞,
the interpolated function must have a high order of continuity as well. Finally, the
method must have small overall errors. In this way, also the derivatives of the inter-
polated field are still accurate enough. We investigate the wave number–dependent
error without looking at a specific flow configuration.

The methods that are investigated are the following. First, we have Lagrange in-
terpolation where a polynomial function of degree $N-1$ passes through N points [10].
Second, we have investigated Catmull–Rom splines [11]. Third, Hermite interpolation
is considered and finally our newly proposed B-spline interpolation method is used.
In Table 8.1 some properties of the interpolation methods are reported. All the inter-
polation methods use piecewise polynomial functions of degree $N-1$ to reconstruct
the field.

---

[2]Please contact the authors in order to get access to the computational code for B-spline inter-
polation.

TABLE 8.1
Overview of the interpolation methods investigated. For all methods the degree of the polynomial function is equal to $N - 1$.

Method	n	Order of continuity	FFT	Comment
Lagrange interpolation	$N-1$	0	1	for even N
		-1	1	for odd N
Spline interpolation [11]	$N-2$	$(N-2)/2$	1	only even N
Hermite interpolation	$N-1$	$(N-2)/2$	$(N/2)^3$	only even N
B-spline based interpolation	$N-1$	$N-2$	1	all N
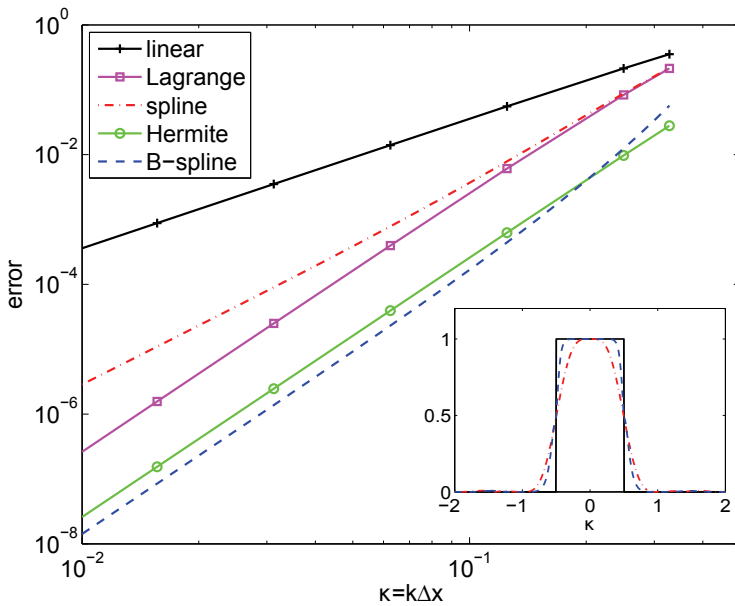
FIG. 8.1. *Relative interpolation error for the Fourier mode; see* (5.12). *For all methods $N = 4$, except for linear interpolation, which has $N = 2$. The subfigure shows the Fourier transform of two interpolation kernels (spline and B-spline), where the solid line represents exact interpolation.*

In order to estimate errors we use relation (5.12) to find wave number–dependent errors. For the four interpolation methods these errors are shown in Figure 8.1. In this figure linear interpolation is added as a reference case. In our case $k_{max}\Delta x = \frac{1}{3}$ in order to avoid aliasing during the computation of the nonlinear term [26]. If this problem were not present, k_{max} could be increased until $k_{max}\Delta x = \frac{1}{2}$. From Figure 8.1 also the order of convergence can be determined and it is found equal to $n + 1$ (the lowest degree of a polynomial function for which the interpolation is not exact), in agreement with Table 8.1.

In order to avoid extra FFTs the interpolated field can be differentiated, as done in relation (3.10), and the error can be computed by means of (5.13). The interpolation errors of the first and second derivatives are shown in Figure 8.2. Here linear interpolation is executed on the derivatives themselves to give a comparison of how accurate the interpolation methods are. One can see, for example, that the second
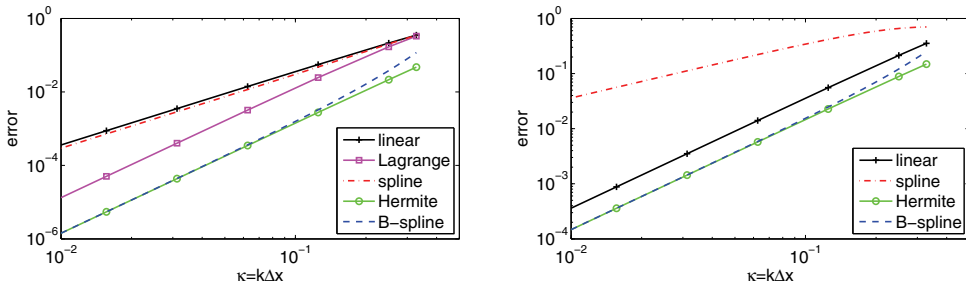
FIG. 8.2. *Relative interpolation error for the first (left) and second derivative (right). Here the linear interpolation is taken on the first and second derivatives, whereas the other methods are taken on the function itself and then differentiated afterwards. Again all methods are taken with $N = 4$, except for linear interpolation, which has $N = 2$.*

derivative is still better approximated by Hermite interpolation (with $N = 4$) than by the linear interpolation executed on the second derivative.

When comparing the interpolation methods one can see that all interpolation methods have a weak point on one of our criteria except for the B-spline-based method. The Lagrange interpolation, for example, is only C^0 continuous for even N and even discontinuous for odd N. Furthermore, the overall error is relatively high compared to the other methods. The spline interpolation has already a better order of continuity but it has a lower order of convergence. Also the overall error is relatively high compared to the other methods. Hermite interpolation on the other hand has an excellent overall error, especially for the second derivative; see section 6. The main disadvantage of this method is that multiple FFTs are needed, which is very time consuming. The B-spline-based interpolation does not have this problem. The time it takes to execute the multiplication in Fourier space can be neglected compared with one FFT. Furthermore, this method reaches a much higher order of continuity compared to the other methods. When looking at the overall errors in Figures 8.1 and 8.2 one can see that they almost match that of Hermite interpolation. This is especially interesting for the second derivative because we have proven that there can not be a better approximation. Note that this second derivative is still continuous for the B-spline interpolation, whereas for Hermite interpolation it is not.

9. Conclusions. We have introduced a general framework for interpolation methods on a rectangular grid. Making use of this framework, we propose an algorithm for fast evaluation of the interpolation in three dimensions. This can easily save considerable computing time compared with other algorithms. It is shown that the computation time needed for this algorithm is close to a theoretical lower bound.

A spectral theory about these interpolation methods is presented, with which the spectral properties of the interpolation methods can be studied. Here basic properties of the interpolation method are shown such as the order of continuity and the order of convergence. Furthermore, errors can be calculated for all Fourier components and also for its derivatives. By the use of this theory a novel B-spline based interpolation method is introduced for application in conjunction with spectral codes, for example, to investigate the dynamics of almost neutrally buoyant particles in turbulence.

Finally, the interpolation methods for spectral codes are compared. The B-spline-based interpolation method has several advantages compared with traditional methods. First, the order of continuity of the interpolated field is higher than that of Hermite interpolation and the other methods investigated. Second, only one FFT

needs to be done, whereas Hermite interpolation needs multiple FFTs for computing the derivatives. Third, the interpolation error matches almost that of Hermite interpolations which is not reached by the other methods investigated. The proposed B-spline interpolation is thus the preferred candidate for particle tracking algorithms applied for turbulent flow simulations.

REFERENCES

[1] F. Toschi and E. Bodenschatz, *Lagrangian properties of particles in turbulence*, Annu. Rev. Fluid Mech., 41 (2009), pp. 375–404.
[2] M. R. Maxey and J. J. Riley, *Equation of motion for a small rigid sphere in a nonuniform flow*, Phys. Fluids, 26 (1983), pp. 883–889.
[3] E. Loth, *Numerical approaches for motion of dispersed particles, droplets and bubbles*, Prog. Energ. Combust., 26 (2000), pp. 161–223.
[4] E. E. Michaelides *Hydrodynamic force and heat/mass transfer from particles, bubbles, and drops—the Freeman Scholar Lecture*, J. Fluids Eng., 125 (2003), pp. 209–238.
[5] M. A. T. van Hinsberg, J. H. M. ten Thije Boonkkamp, and H. J. H. Clercx, *An efficient, second order method for the approximation of the Basset history force*, J. Comput. Phys., 230 (2011), pp. 1465–1478.
[6] M. van Aartrijk and H. J. H. Clercx, *The dynamics of small inertial particles in weakly stratified turbulence*, J. Hydro-Envir. Res., 4 (2010), pp. 103–114.
[7] P. K. Yeung and S. B. Pope, *An algorithm for tracking fluid particles in numerical simulations of homogeneous turbulence*, J. Comput. Phys., 79 (1988), pp. 373–416.
[8] S. Balachandar and M. R. Maxey, *Methods for evaluating fluid velocities in spectral simulations of turbulence*, J. Comput. Phys., 83 (1989), pp. 96–125.
[9] T. M. Lehmann, C. Gönner, and K. Spitzer, *Survey: Interpolation methods in medical image processing*, IEEE Trans. Med. Imag., 18 (1999), pp. 1049–1075.
[10] J. D. Faires and R. L. Burden, *Numerical Methods*, PWS, Boston, MA, 1993.
[11] E. Catmull and R. Rom, *A class of local interpolating splines*, in Computer Aided Geometric Design, R. E. Barnhill and R. F. Reisenfeld, eds., Academic Press, New York, 1974, pp. 317–326.
[12] M. T. Heath, *Scientific Computing*, McGraw-Hill, New York, 2005.
[13] G. D. Reeves, R. D. Belian, and T. A. Fritz, *Numerical tracking of energetic particle drifts in a model magnetosphere*, J. Geophys. Res., 96 (1991), pp. 13997–14007.
[14] F. Mackay, R. Marchand, and K. Kabin, *Divergence-free magnetic field interpolation and charged particle trajectory integration*, J. Geophys. Res., 111 (2006), A06208.
[15] H. S. Hou and H. C. Andrews, *Cubic splines for image interpolation and digital filtering*, IEEE Trans. Acoust. Speech Signal Processing, ASSP-26, (1978), pp. 508–517.
[16] H. Homann, J. Dreher, and R. Grauer, *Impact of the floating-point precision and interpolation scheme on the results of DNS of turbulence by pseudo-spectral codes*, Comput. Phys. Commun., 177 (2007), pp. 560–565.
[17] J. Choi, K. Yeo, and C. Lee, *Lagrangian statistics in turbulent channel flow*, Phys. Fluids, 16 (2004), pp. 779–793.
[18] C. Marchioli, V. Armenio, and A. Soldati, *Simple and accurate scheme for fluid velocity interpolation for Eulerian-Lagrangian computation of dispersed flows in 3D curvilinear grids*, Comput. Fluids, 36 (2007), pp. 1187–1198.
[19] C. Marchioli, A. Soldati, J. G. M. Kuerten, B. Arcen, A. Tanière, G. Goldensoph, K. D. Squires, M. F. Cargnelutti, and L. M. Portela, *Statistics of particle dispersion in direct numerical simulations of wall-bounded turbulence: Results of an international collaborative benchmark test*, Int. J. Multiphase Flow, 34 (2008), pp. 879–893.
[20] G. B. Jacobs, D. A. Kopriva, and F. Mashayek, *Towards efficient tracking of inertial particles with high-order multidomain methods*, J. Comput. Appl. Math., 206 (2007), pp. 392–408.
[21] G. E. Karniadakis and J. S. Hesthaven, *Spectral interpolation in non-orthogonal domains: Algorithms and applications*, J. Engrg. Math., 56 (2006), pp. 201–202.
[22] C. C. Lalescu, B. Teaca, and D. Carati, *Implementation of high order spline interpolations for tracking test particles in discretized fields*, J. Comput. Phys., 229 (2010), pp. 5862–5869.

[23] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[24] F. LEKIEN AND J. MARSDEN, *Tricubic interpolation in three dimensions*, Internat. J. Numer. Methods Engrg., 63 (2005), pp. 455–471.

[25] K. QIN, *General matrix representations for B-splines*, Vis. Comput., 16 (2000), pp. 177–186.

[26] C. CANUTO, M. HUSSAINI, A. QUARTERONI, AND T. ZANG, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin, New York, 1988; second revised printing, 1990.