J. Plasma Phys. (2017), *vol.* 83, 905830407 © Cambridge University Press 2017
doi:10.1017/S0022377817000599

Fast and spectrally accurate evaluation of gyroaverages in non-periodic gyrokinetic-Poisson simulations

J. Guadagni[1] **and A. J. Cerfon**[1,†]

[1]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA

(Received 5 May 2017; revised 7 July 2017; accepted 10 July 2017)

We present a fast and spectrally accurate numerical scheme for the evaluation of the gyroaveraged electrostatic potential in non-periodic gyrokinetic-Poisson simulations. Our method relies on a reformulation of the gyrokinetic-Poisson system in which the gyroaverage in Poisson's equation is computed for the compactly supported charge density instead of the non-periodic, non-compactly supported potential itself. We calculate this gyroaverage with a combination of two Fourier transforms and a Hankel transform, which has the near optimal run-time complexity $O(N_\rho(P+\hat{P})\log(P+\hat{P}))$, where P is the number of spatial grid points, $\hat{P}$ the number of grid points in Fourier space and N_ρ the number of grid points in velocity space. We present numerical examples illustrating the performance of our code and demonstrating geometric convergence of the error.

Key words: intense particle beams, magnetized plasmas, plasma simulation

1. Introduction

Plasmas in which the characteristic collision time is long compared to the time scales of the processes of interest must usually be described with kinetic equations. In general, phase space is six-dimensional, so solving such equations numerically is computationally costly. For phenomena that are slow compared to the period of gyromotion of the particles, significant reduction in computational time and complexity can be obtained by averaging the equations over the gyromotion – an operation known as gyroaveraging. The formal theory for this averaging procedure is called gyrokinetic theory (Rutherford & Frieman 1968; Taylor & Hastie 1968; Catto & Tsang 1977; Catto 1978; Antonsen & Lane 1980; Frieman & Chen 1982; Brizard & Hahm 2007). It has been widely and successfully implemented in a large number of numerical codes, mostly for the study of magnetic confinement fusion and astrophysical plasmas (Candy & Waltz 2003; Garbet *et al.* 2010; Numata *et al.* 2010; Görler *et al.* 2011). Many recent discoveries in plasma physics relied heavily on gyrokinetic simulations produced by these codes, particularly for problems in which kinetic turbulence plays a central role.

Despite the remarkable achievements of the solvers mentioned above, there remain questions regarding the numerical implementation of the gyrokinetic equations that

† Email address for correspondence: cerfon@cims.nyu.edu

are not entirely resolved yet. In this article, we address one of them, namely the fast and high-order accurate numerical evaluation of gyroaveraged quantities in settings in which the periodicity of the physical quantities cannot be assumed. The issue can be summarised as follows. It is well-known that in Fourier space the gyroaveraging operation reduces to a multiplication by the Bessel function J_0. This fact is conveniently used by solvers which assume periodicity of the physical quantities, called local codes, turning gyroaveraging into a fast and high-order accurate operation. It is, however, not straightforward to use this fact in non-periodic settings due to difficulties associated with the evaluation of the Fourier transform in these situations (Crouseilles *et al.* 2010; Steiner *et al.* 2015). Consequently, two alternative approaches have been followed. A first approach is to replace the Bessel function with a Padé expansion approximation. The associated expansion for the gyroaveraging operation in Fourier space can then be transformed back to real space, and the gyroaverage is the solution of a tractable partial differential equation (Sarazin *et al.* 2005; Steiner *et al.* 2015). The Padé approximation approach has the advantage of being fast, but is well known to cause an overdamping of small scales, which limits its accuracy (Steiner *et al.* 2015). A more common approach is to evaluate directly the gyroaverage integral through numerical quadrature, relying on interpolation on points along the gyroring of the function which is gyroaveraged (Jolliet *et al.* 2007; Crouseilles *et al.* 2010; Görler *et al.* 2011; Steiner *et al.* 2015). By choosing high-order interpolation schemes, high-order accuracy can be achieved with this method, but with a larger computational cost than with a Padé based approach (Steiner *et al.* 2015).

In this article, we present a different strategy to calculate the gyroaveraged electrostatic potential, which leads to a spectrally convergent numerical scheme and a nearly optimal computational complexity, namely $O(N_\rho(P+\hat{P})\log(P+\hat{P}))$, where P is the number of grid points in the spatial domain, $\hat{P}$ the number of grid points in Fourier space and N_ρ the number of grid points in velocity space. Our approach relies on a reformulation of the gyrokinetic-Poisson system in which Poisson's equation is not solved for the electrostatic potential Φ but instead for the gyroaverage of Φ at fixed guiding centre position $\boldsymbol{R}$, which we call χ. In that modified gyrokinetic-Poisson system, the only gyroaverage which needs to be evaluated numerically is the gyroaverage of the charge density, which is a compactly supported function. We are thus able to compute the Fourier transform of the charge density, and rely on the standard multiplication by the Bessel function J_0 in Fourier space and the Hankel transform to evaluate its gyroaverage. At that point, the desired spectral convergence and optimal run-time complexity follow immediately from the adoption of well-known spectrally accurate algorithms with nearly optimal complexity for the calculation of the forward and inverse Fourier transforms and of the Hankel transform.

The structure of the article is as follows. In § 2, we present the gyrokinetic-Poisson system we are interested in and derive our reformulation of the system of equations, which allows the use of a Fourier representation for the calculation of the gyroaverage even for non-periodic settings. In § 3, we describe our numerical scheme by decomposing the gyroaverage operation into more fundamental mathematical operations. Note that our scheme relies on the sequential implementation of very well-known algorithms, which are open-source and readily available for implementation by any user. In § 4, we focus on functions which can be gyroaveraged analytically and compare the analytic results with our numerical results to demonstrate the spectral convergence of the numerical error. We summarise our work in § 5 and suggest directions for future work.

2. New formulation for the gyrokinetic-Poisson equations

In this section, we introduce the gyrokinetic-Poisson system of equations we will consider in the remainder of this article, and derive a reformulation of the system which is well suited for the new numerical scheme for gyroaveraging we propose. We chose these gyrokinetic-Poisson equations for two reasons. First, their relative simplicity allows us to focus on the mathematical aspects of our numerical method for gyroaveraging, which is the main point of the article. Second, the equations correspond to a good model for the study of intense non-neutral beams in cyclotrons and vacuum tubes with high magnetic fields.

2.1. *Avoiding gyroaverages of the electrostatic potential*

We consider the simple situation of a two-dimensional single-species plasma in the xy-plane immersed in a constant and uniform magnetic field $\boldsymbol{B}=B_0\boldsymbol{e}_z$. We define the gyrofrequency $\varOmega = qB_0/m$, with q the charge of the particles in the plasma, and m their mass. The position of a given particle is $\boldsymbol{r}=\boldsymbol{R}+\boldsymbol{\rho}$, where $\boldsymbol{R}$ is the guiding centre (or gyrocentre) position, and $\boldsymbol{\rho}$ is the Larmor radius vector $\boldsymbol{\rho}=-\rho\sin\gamma\boldsymbol{e}_x+\rho\cos\gamma\boldsymbol{e}_y$, where γ is the gyroangle and $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$ is the standard Cartesian orthonormal basis of $\mathbb{R}^3$, with $\boldsymbol{e}_z$ aligned with the magnetic field. If velocities are normalised to the thermal speed v_{th} and spatial scales are normalised to the Larmor radius $\rho_L = v_{\mathrm{th}}/\varOmega$, the gyrokinetic-Poisson equations for the gyrocentre distribution function $f(\boldsymbol{R}, \rho, t)$ are given by (Hazeltine & Meiss 2003; Plunk *et al.* 2010)

$$\frac{\partial f}{\partial t}+\boldsymbol{e}_z\times\langle\nabla_{\boldsymbol{r}}\varPhi\rangle_{\boldsymbol{R}}\boldsymbol{\cdot}\nabla_{\boldsymbol{R}}f=0 \tag{2.1}$$

$$\nabla_{\boldsymbol{r}}^2\varPhi(\boldsymbol{r},t)=-\int_0^{+\infty}\int_0^{2\pi}f(x+\rho\sin\gamma, y-\rho\cos\gamma,\rho,t)\rho\,\mathrm{d}\rho\,\mathrm{d}\gamma\equiv-2\pi\tilde{f}(x,y,t), \tag{2.2}$$

where

$$\tilde{f}(\boldsymbol{r},t)=\frac{1}{2\pi}\int_0^{+\infty}\int_0^{2\pi}f(x+\rho\sin\gamma, y-\rho\cos\gamma,\rho,t)\rho\,\mathrm{d}\rho\,\mathrm{d}\gamma. \tag{2.3}$$

In (2.1), $\langle\cdot\rangle_{\boldsymbol{R}}$ represents the gyroaverage at fixed guiding centre position $\boldsymbol{R}=X\boldsymbol{e}_x+Y\boldsymbol{e}_y$:

$$\langle\varPhi\rangle_{\boldsymbol{R}}=\frac{1}{2\pi}\int_0^{2\pi}\varPhi(X-\rho\sin\gamma, Y+\rho\cos\gamma, t)\,\mathrm{d}\gamma, \tag{2.4}$$

where X and Y are held fixed. We chose the relatively simple gyrokinetic system of equations (2.1) and (2.2) to better focus on the central question of this paper, namely the fast and accurate evaluation of the averages over the gyroangle γ appearing in both equations. However, the method we present here is applicable to the more general gyrokinetic systems commonly used to study astrophysical and fusion plasmas. We should mention that (2.1) and (2.2) are a surprisingly accurate description of the dynamics of a beam of charged particles in the plane perpendicular to the magnetic field in high intensity cyclotrons (Cerfon *et al.* 2013; Guadagni 2015; Cerfon 2016). With this application in mind, we want to allow boundary conditions on $\varPhi$ that are not periodic, such as free space boundary conditions for instance.

If one has a numerical method to accurately evaluate $\langle\nabla_{\boldsymbol{r}}\varPhi\rangle_{\boldsymbol{R}}$ on the desired numerical grid, then a number of established numerical schemes are available to advance f in time according to (2.1) (Peterson & Hammett 2013; Guadagni 2015). Clearly, the challenge that is specific to gyrokinetics is the numerical evaluation of the gyroaverage for the charge density in (2.2) and of the gyroaveraged potential (2.4) when these quantities are not periodic. In the introduction, we have mentioned popular methods to accomplish this task. We propose a different approach, based on Fourier transforms, which leads to high-order accurate answers. Such an approach is not practical in the current formulation of the problem since Φ is not periodic and is unbounded for free space boundary conditions. Our first step therefore consists of casting equations (2.1) and (2.2) in a form which is compatible with a Fourier representation.

For our simple geometry, $\boldsymbol{\nabla}_{\boldsymbol{r}} = \boldsymbol{\nabla}_{\boldsymbol{R}}$, so it is straightforward to re-express (2.1) and (2.2) in terms of equations for quantities which only depend on the guiding centre position (X, Y):

$$\frac{\partial f}{\partial t} + \boldsymbol{e}_z \times \boldsymbol{\nabla}_{\boldsymbol{R}} \langle \Phi \rangle_{\boldsymbol{R}} \boldsymbol{\cdot} \boldsymbol{\nabla}_{\boldsymbol{R}} f = 0 \tag{2.5}$$

$$\nabla_{\boldsymbol{R}}^2 \Phi(\boldsymbol{R}, t) = -2\pi \tilde{f}(X, Y, t), \tag{2.6}$$

where we also made use of the fact that the gradient operator commutes with the gyroaveraging operator (Guadagni 2015). Since Φ is not periodic and unbounded, it does not have a Fourier transform. Computing $\langle \Phi \rangle_{\boldsymbol{R}}$ using standard Fourier techniques is therefore not an available option. The idea instead is to define a new potential-like function $\chi(\boldsymbol{R}, \rho, t) = \langle \Phi \rangle_{\boldsymbol{R}}(\boldsymbol{R}, \rho, t)$ which we call the gyropotential. The key then is to not evaluate χ as given by its definition, but instead to see that it is the solution of the Poisson equation

$$\nabla^2 \chi = \frac{1}{2\pi} \int_0^{2\pi} \tilde{f}(X + \rho \sin\gamma, Y - \rho\cos\gamma, t)\, \mathrm{d}\gamma \equiv -2\pi \langle \tilde{f} \rangle_{\boldsymbol{R}}, \tag{2.7}$$

which we obtained by gyroaveraging (2.6) holding the guiding centre position $\boldsymbol{R}$ fixed, and using once more the fact that the gradient operator commutes with the gyroaveraging operator. At this point, we have turned (2.1) and (2.2) into the following new system of equations:

$$\frac{\partial f}{\partial t} + \boldsymbol{e}_z \times \boldsymbol{\nabla} \chi \boldsymbol{\cdot} \boldsymbol{\nabla} f = 0 \tag{2.8}$$

$$\nabla^2 \chi = -2\pi \langle \tilde{f} \rangle_{\boldsymbol{R}}. \tag{2.9}$$

Unsurprisingly, this system shares many similarities with the two-dimensional inviscid Euler equations in vorticity–streamfunction form (Plunk *et al.* 2010; Cerfon 2016). The major difference with the Euler equations is that the term $\langle \tilde{f} \rangle_{\boldsymbol{R}}$ couples the dynamics at different values of ρ. In the context of the present article, the significant aspect of the system of equations above is that it has the desirable property of only involving gyroaverages of $\tilde{f}$, which, unlike Φ, can be approximated numerically by a compactly supported function. Once $\langle \tilde{f} \rangle_{\boldsymbol{R}}$ is known, Poisson's equation (2.9) can be solved with standard methods.

It may first seem as if our reformulation of the gyrokinetic Vlasov–Poisson system has a large computational cost because χ is a function of the variable ρ. That means that Poisson's equation (2.9) has to be solved as many times as there are discretisation points for the ρ variable. In practice, however, this is not a salient issue, for three reasons. First, there exists a wide choice of fast and high-order accurate Poisson solvers with nearly optimal computational complexity. For example, for the free space boundary conditions which are the relevant conditions for beam dynamics in cyclotrons, we may mention the solvers by Jiang *et al.* (2014) and by Vico *et al.* (2016), which solve Poisson's equation on a regular grid in $O(P \log P)$ time, where P is the number of space discretisation points. Second, it is quite common in gyrokinetic simulations to have a very small number ($\leqslant 20$) of grid points for the ρ variable (Wilkening *et al.* 2015; J. Candy, 2016, Private communication). Lastly, the Poisson solve is rarely the rate limiting step in gyrokinetic simulations. This is particularly true for particle-in-cell solvers, in which the particle operations typically dominate the computation and storage requirements (Ricketson & Cerfon 2017).

2.2. *Limitations of a Fourier series expansion*

Thus far, we have expressed the gyrokinetic-Poisson system of equations (2.1)–(2.2) in a form that only requires gyroaveraging a function which can be well approximated by a compactly supported function, given by the system of equations (2.8)–(2.9). This allows us to use a Fourier basis to represent that function and to calculate its gyroaverage in Fourier space. However, for problems which are not periodic, despite the compact support of $\tilde{f}$, a gyroaveraging scheme based on a Fourier series expansion for $\tilde{f}$ can only lead to highly accurate answers under specific conditions. To understand this point, consider the model charge density shown in figure 1. The computational domain is represented by the black box in the lower right corner, and the support of the charge density is shown in orange there. By expanding $\tilde{f}$ in a Fourier basis, one would make the problem periodic, which is represented by the three boxes which are contiguous to the computational domain in figure 1 and represented with dashed lines. Now, consider the four gyroaverages corresponding to different values of ρ, with common centre the black dot in the upper left corner of the computational domain, and shown in the figure as four dashed circles. The two inner circles, highlighted in blue, give the correct result for the gyroaverage: the average only involves contributions from the true charge density – and no contribution at all for the innermost circle, as it should. However, the two outer circles, highlighted in red, give a false result for the gyroaverage. Along the third circle, the contribution from the true charge density is summed, but so are the contributions from the 'ghosts', unphysical charge densities in the contiguous cells. For the outermost circle, corresponding to large ρ, the average should be zero, but it is not because the 'ghost' charge densities contribute to the average.

A possible way to address this issue is to increase the size of the computational domain by padding it with zeroes in such a way that even for the maximum ρ considered in the simulations, gyrocircles only average the true charge density. This idea is very similar to what is sometimes done to solve Poisson's equation with a plane wave representation (Genovese *et al.* 2006). It has the clear disadvantage of leading to large computational domains, and therefore significantly more expensive evaluations of the fast Fourier transform (FFT) typically used to calculate the coefficients in the Fourier series.

Now, while a Fourier series expansion leads to the complications discussed above, because it intrinsically implies periodicity for the problem, an expansion on a continuous Fourier basis, i.e. through the Fourier transform, has all the desirable properties for a high-order accurate numerical scheme for gyroaveraging. This is what we discuss in the next section.

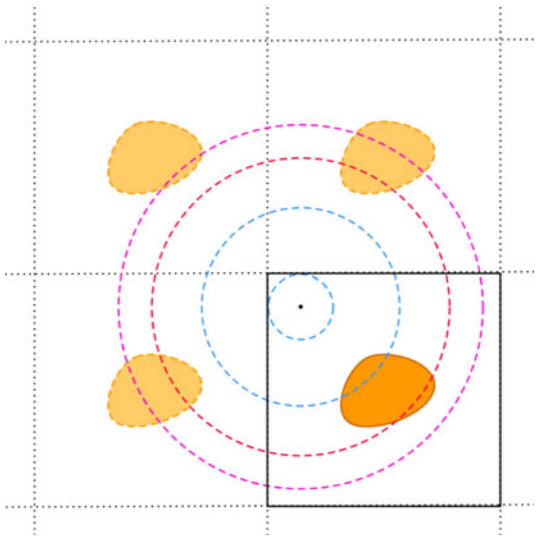

FIGURE 1. Illustration of the difficulties associated with using a Fourier series representation for computing the gyroaverage of a distribution function f is a non-periodic setting. The computational domain is the box bounded by the continuous lines. The boxes bounded by dashed lines represent some of the boxes induced by the periodicity of the Fourier series representation. The gyroaverage along the blue orbits yield correct results, while the gyroaverage along the red and purple orbits lead to fictitious contributions to the true gyroaverage.

2.3. *Fourier and Hankel transform representation*

Consider a function $u(\boldsymbol{x}, \rho)$ on $\mathbb{R}^2 \times \mathbb{R}_{\geqslant 0}$. The Fourier transform of u with respect to the first two inputs is defined by

$$\hat{u}(\boldsymbol{\xi}, \rho) \equiv (\mathcal{F}_{\boldsymbol{x}} u)(\boldsymbol{\xi}, \rho) = \int_{\mathbb{R}^2} u(\boldsymbol{x}, \rho) \mathrm{e}^{-\mathrm{i}\boldsymbol{\xi}\cdot\boldsymbol{x}} \, \mathrm{d}\boldsymbol{x}. \tag{2.10}$$

The inverse Fourier transform is defined by

$$u(\boldsymbol{x}, \rho) \equiv (\mathcal{F}_{\boldsymbol{\xi}}^{-1} \hat{u})(\boldsymbol{x}, \rho) = \frac{1}{4\pi^2} \int_{\mathbb{R}^2} \hat{u}(\boldsymbol{\xi}, \rho) \mathrm{e}^{\mathrm{i}\boldsymbol{\xi}\cdot\boldsymbol{x}} \, \mathrm{d}\boldsymbol{x}. \tag{2.11}$$

Finally, given a real-valued function $u(s)$ defined on $\mathbb{R}_{\geqslant 0}$, its Hankel transform of order zero is defined by

$$(\mathcal{H}_0 u)(s) = \int_0^\infty u(\rho) \mathrm{J}_0(\rho s) \rho \, \mathrm{d}\rho, \tag{2.12}$$

where J_0 is the Bessel function of the first kind and of order zero. Introducing the gyroaverage at fixed particle position $\boldsymbol{r} = x\boldsymbol{e}_x + y\boldsymbol{e}_y$,

$$\langle f \rangle (x, y, \rho, t) \equiv \frac{1}{2\pi} \int_0^{2\pi} f(x + \rho \sin\gamma, y - \rho\cos\gamma, \rho, t) \, \mathrm{d}\gamma, \tag{2.13}$$

it is well-known that $\mathcal{F}\langle f\rangle(\boldsymbol{\xi},\rho,t)=\mathrm{J}_0(\rho\xi)(\mathcal{F}f)(\boldsymbol{\xi},\rho,t)$. Therefore, we can write

$$(\mathcal{F}\tilde{f})(\boldsymbol{\xi},t)=\int_0^\infty(\mathcal{F}\langle f\rangle)(\boldsymbol{\xi},\rho,t)\rho\,\mathrm{d}\rho=\int_0^\infty \mathrm{J}_0(\rho\xi)(\mathcal{F}f)(\boldsymbol{\xi},\rho,t)\rho\,\mathrm{d}\rho=(\mathcal{H}_0\mathcal{F}f)(\boldsymbol{\xi},t). \tag{2.14}$$

And since we also have the identity

$$(\mathcal{F}\langle\tilde{f}\rangle_{\boldsymbol{R}})(\boldsymbol{\xi},\rho,t)=\mathrm{J}_0(\rho\xi)(\mathcal{F}\tilde{f})(\boldsymbol{\xi},t)=\mathrm{J}_0(\rho\xi)(\mathcal{H}_0\mathcal{F}f)(\boldsymbol{\xi},t), \tag{2.15}$$

we obtain the desired expression for the only term which needs to be gyroaveraged in the system of equations (2.8)–(2.9):

$$\langle\tilde{f}\rangle_{\boldsymbol{R}}(\boldsymbol{R},\rho,t)=\mathcal{F}^{-1}(\mathrm{J}_0(\rho\xi)\mathcal{H}_0\mathcal{F}f)(\boldsymbol{R},\rho,t)\equiv\mathcal{G}f(\boldsymbol{R},\rho,t). \tag{2.16}$$

Note that there is a slight abuse of notation in (2.14)–(2.16) above: in (2.12) we have defined the Hankel transform for functions of a single variable but $(\mathcal{F}f)$ clearly is not.

Thus, to summarise § 2, we have shown that the gyrokinetic-Poisson system can be rewritten as

$$\frac{\partial f}{\partial t}+\boldsymbol{e}_z\times\nabla\chi\boldsymbol{\cdot}\nabla f=0 \tag{2.17}$$

$$\nabla^2\chi=-2\pi\mathcal{G}f(\boldsymbol{R},\rho,t). \tag{2.18}$$

All the gyroaveraging operations are contained in the operator $\mathcal{G}$ that operates on the function f, which is bounded and has compact support. Furthermore, if $f(X,Y,\rho,t)$ and $\chi(X,Y,\rho,t)$ are known at a given time t, the physical potential $\varPhi(X,Y,t)$ and the number density $n(X,Y,t)$ are given by $\varPhi(X,Y,t)=\chi(X,Y,0,t)$ and $n(X,Y,t)=2\pi\mathcal{G}f(X,Y,0,t)$. In other words, they do not require additional computation, and one just needs to read the values of $\chi(X,Y,\rho,t)$ and $2\pi\mathcal{G}f(X,Y,\rho,t)$ corresponding to $\rho=0$.

It remains to explain how we discretise f and the operators on the right-hand side of (2.16) in order to evaluate $\mathcal{G}f(\boldsymbol{R},\rho,t)$ numerically to high accuracy. This is the purpose of the next section.

3. Numerical scheme

In this section, we present the algorithmic details of our numerical scheme for computing $\mathcal{G}f(\boldsymbol{R},\rho,t)$, which is designed to lead to high-order accuracy and near optimal run-time complexity. We give a brief justification for our choice of grids in § 3.1, after which the presentation follows the natural decomposition of $\mathcal{G}$ into its elementary operators: we describe our scheme for calculating the Fourier transform $\mathcal{F}$ in § 3.2, present our numerical method for computing the Hankel transform $\mathcal{H}_0$ in § 3.3 and describe our scheme for the inverse Fourier transform $\mathcal{F}^{-1}$ in § 3.4. We conclude § 3 by giving recommendations for the choice of grid sizes and resolutions in § 3.5.

3.1. *Grid choices*

For the design of our numerical scheme, we chose to operate under the constraint that the spatial grid be uniformly spaced. The reason for this is that many of the popular and advanced schemes for both the Vlasov equation and Poisson's equation are either very difficult to implement on non-uniform grids, or simply do not work on such grids (Shu 1999; Peterson & Hammett 2013; Jiang *et al.* 2014; Vico *et al.* 2016). As we will see in §§ 3.2 and 3.4, this constraint leads to computational costs which could have been avoided if we had given ourselves the freedom to use a non-equispaced grid for the spatial variables. Still, even with this constraint and its associated computational cost, the scheme we propose here can be categorised as a fast solver, in the sense that its run-time complexity is $O(K \log K)$, where K is the number of degrees of freedom in the problem.

In contrast to the spatial grid, we never solve any partial differential equation with respect to either the Fourier space coordinates (ξ_x, ξ_y) or the ρ-coordinate. We take advantage of this fact by using Chebyshev grids for each, which give us access to highly efficient and accurate numerical quadrature schemes.

3.2. *The Fourier transform $\mathcal{F}$*

We start by presenting our method for evaluating $\mathcal{F}f$. For the simplicity of the presentation, we focus on the case in which f depends only on one spatial variable x. The method generalises directly to the two-dimensional tensor grid which we use to compute $\mathcal{G}f$. Consider the function u on $\mathbb{R}$. Its Fourier transform is

$$\hat{u}(\xi) = \int_{-\infty}^{\infty} u(x) \mathrm{e}^{-\mathrm{i}\xi x}\, \mathrm{d}x. \tag{3.1}$$

We consider the case in which u is compactly supported on the domain $I = [-a/2, a/2]$, which is relevant to us, and write u as the *exact* series

$$u(x) = \left(\sum_{k=-\infty}^{\infty} c_k \mathrm{e}^{2\pi\mathrm{i}kx/a} \right) \mathbf{1}_I(x) \tag{3.2}$$

$$c_k = \frac{1}{a} \int_{-a/2}^{a/2} u(x) \mathrm{e}^{-2\pi\mathrm{i}kx/a}\, \mathrm{d}x. \tag{3.3}$$

In (3.2), $\mathbf{1}_I$ is the indicator function for the interval I, defined by $\mathbf{1}_I(x) = 1$ if $x \in I$, and $\mathbf{1}_I(x) = 0$ otherwise. The Fourier transform of u is then given by

$$\hat{u}(\xi) = a \sum_{k=-\infty}^{\infty} c_k \operatorname{sinc}\left(k - \frac{a\xi}{2\pi} \right), \tag{3.4}$$

where $\operatorname{sinc}(z) \equiv \sin(\pi z)/(\pi z)$. We proceed as follows to compute the sum (3.4) in optimal time, with high accuracy, and for values ξ on an arbitrary grid in the Fourier domain. The Fourier coefficients c_k are calculated with a straight forward call to the FFT, after discretising u on a uniform grid. The complexity of this operation is $O(N \log N)$, where N is the number of discretisation points for the x-grid. Once the values of c_k are known, we would like to evaluate the sum (3.4) on a ξ-grid which is optimal for the operations which will follow the calculation of $\mathcal{F}$ in the evaluation of the full operator $\mathcal{G}$. As mentioned in the previous section, that grid is a non-equispaced Chebyshev grid, which may in addition have different bounds than those naturally induced by the bounds of I in the Fourier transform. To evaluate (3.4), we thus rely on the fast sinc transform (FST) (Greengard *et al.* 2006), which is itself based on the non-uniform fast Fourier transform (NUFFT) (Dutt & Rokhlin 1993), and which is freely available in a version described in (Greengard & Lee 2004; Lee & Greengard 2005). The computational complexity of the FST is $O((N+\hat{N})\log(N+\hat{N}))$, where $\hat{N}$ is the number of discretisation points for the ξ-grid. For the gyrokinetic-Poisson system (2.17)–(2.18) we are interested in, for which f depends on two spatial variables, this becomes $O((P+\hat{P})\log(P+\hat{P}))$, where $P\sim N^2$ is the total number of spatial grid points and $\hat{P}\sim\hat{N}^2$ is the total number of grid points in Fourier space. Since we need to repeat this operation for each value of ρ, the overall complexity of this step is $O(N_\rho(P+\hat{P})\log(P+\hat{P}))$, where N_ρ is the number of grid points in velocity space.

Before closing this section, we mention an alternative way of evaluating (3.1) for a compactly supported function u. Since u is compactly supported, it can be viewed as periodic on the domain of integration, so the trapezoidal rule provides a spectrally accurate scheme for the numerical evaluation of the integral (Trefethen & Weideman 2014). The discrete sum resulting from the application of the trapezoidal rule can then be computed with a direct application of the FFT (Pataki & Greengard 2011). The issue with this approach is that the number of grid points in real space determines the number of grid points in Fourier space. For most functions of physical interest, the representation in Fourier space is significantly more oscillatory than the representation in real space, so proper resolution in Fourier space requires significant oversampling in real space (Pataki & Greengard 2011), which is computationally costly. In contrast, the FST gives us the choice to have a larger number of grid points in Fourier space than the number of grid points we use in real space. Now, since the FST is more expensive than the regular FFT by a constant factor, it can be shown that the run-time complexity of the two approaches is comparable. In our implementations of the scheme we present here, we favour the FST approach because we find the framework in which real space and Fourier space are decoupled elegant, convenient and efficient.

3.3. *The Hankel transform $\mathcal{H}_0$*

The next step in the evaluation of $\mathcal{G}f$ corresponds to the calculation of the Hankel-like integral

$$g(\boldsymbol{\xi},t)=\int_0^\infty \hat{f}(\boldsymbol{\xi},\rho,t)\mathrm{J}_0(\rho\xi)\rho\,\mathrm{d}\rho, \tag{3.5}$$

with $\xi=||\boldsymbol{\xi}||$. Since $\boldsymbol{\xi}$ and t are fixed parameters in this integral, we simplify the notation and consider the computation of the integral

$$H(\xi)=\int_0^\infty h(\rho)\mathrm{J}_0(\rho\xi)\rho\,\mathrm{d}\rho. \tag{3.6}$$

We consider in this work that to the desired numerical accuracy, $\hat{f}$ has compact support in the ρ-variable. In the context of (3.6), this means that there exists a maximum $\overline{\rho}$ such that $h=0$ outside the interval $I_\rho=[0,\overline{\rho}]$. To compute (3.6) we discretise I_ρ with a Chebyshev grid in ρ and use Clenshaw–Curtis quadrature, which has geometric convergence for the class of functions we consider here (Trefethen 2008):

$$H(\xi)\approx\sum_{k=1}^{N_\rho} w_k h(\rho_k)\mathrm{J}_0(\rho_k\xi)\rho_k, \tag{3.7}$$

where N_ρ is the number of Chebyshev grid points on the interval I_ρ, the values of ρ_k are the Chebyshev abscissae, and the values of w_k are the Clenshaw–Curtis quadrature weights. Now, observe that for fixed computational ρ- and ξ-grids, the values of $w_k \mathrm{J}_0(\rho_k \xi)\rho_k$ can be precomputed for all k and for all possible values of ξ. Hence, in practice, each individual Hankel integral can be computed as the inner product of one data vector which changes with each time step and one vector of fixed kernel weights. To be more precise, observe that in our case, h is the function $\mathcal{F}f$, which does not only depend on ρ, but also on $\boldsymbol{\xi}$ and t, as can be seen in (2.14). Letting $h(\boldsymbol{\xi}, \rho, t) = \mathcal{F}f(\boldsymbol{\xi}, \rho, t)$ to simplify the notation in the matrices below, all Hankel integrals are computed at once by considering the $N_\rho \times \hat{P}$ matrices $\boldsymbol{h}$ and $\boldsymbol{W}$ defined by

$$\boldsymbol{h} = \begin{pmatrix} h(\boldsymbol{\xi}_1, \rho_1, t) & h(\boldsymbol{\xi}_2, \rho_1, t) & \dots & h(\boldsymbol{\xi}_{\hat{P}}, \rho_1, t) \\ h(\boldsymbol{\xi}_1, \rho_2, t) & h(\boldsymbol{\xi}_2, \rho_2, t) & \dots & h(\boldsymbol{\xi}_{\hat{P}}, \rho_2, t) \\ \vdots & \vdots & \dots & \vdots \\ h(\boldsymbol{\xi}_1, \rho_{N_\rho}, t) & h(\boldsymbol{\xi}_2, \rho_{N_\rho}, t) & \dots & h(\boldsymbol{\xi}_{\hat{P}}, \rho_{N_\rho}, t) \end{pmatrix} \tag{3.8}$$

and

$$\boldsymbol{W} = \begin{pmatrix} w_1 \mathrm{J}_0(\rho_1 \xi_1)\rho_1 & w_1 \mathrm{J}_0(\rho_1 \xi_2)\rho_1 & \dots & w_1 \mathrm{J}_0(\rho_1 \xi_{\hat{P}})\rho_1 \\ w_1 \mathrm{J}_0(\rho_2 \xi_1)\rho_2 & w_1 \mathrm{J}_0(\rho_2 \xi_2)\rho_2 & \dots & w_1 \mathrm{J}_0(\rho_2 \xi_{\hat{P}})\rho_2 \\ \vdots & \vdots & \dots & \vdots \\ w_1 \mathrm{J}_0(\rho_{N_\rho} \xi_1)\rho_{N_\rho} & w_1 \mathrm{J}_0(\rho_{N_\rho} \xi_2)\rho_{N_\rho} & \dots & w_1 \mathrm{J}_0(\rho_{N_\rho} \xi_{\hat{P}})\rho_{N_\rho} \end{pmatrix}, \tag{3.9}$$

where $\boldsymbol{\xi}_j$ is now viewed as the vector whose components are the coordinates of the jth pair of the $\hat{P}$ Fourier coordinate pairs. Note that $\boldsymbol{W}$ is fixed for all time and can be stored after its initial computation. The right-hand side of (2.14), $\mathcal{H}_0\mathcal{F}f(\boldsymbol{\xi}, t)$, is then calculated by computing the entrywise product of $\boldsymbol{h}$ and $\boldsymbol{W}$ and summing the columns of that matrix. In mathematical notation, we may write

$$\mathcal{H}_0\mathcal{F}f(\boldsymbol{\xi}, t) \approx \left(\sum_{i=1}^{N_\rho} (\boldsymbol{h} \odot \boldsymbol{W})_{i1}, \dots, \sum_{i=1}^{N_\rho} (\boldsymbol{h} \odot \boldsymbol{W})_{i\hat{P}} \right), \tag{3.10}$$

where $\odot$ represents the Schur product (Davis 1962), i.e. entrywise product. The run-time complexity of this operation is $O(N_\rho \hat{P})$. As a result, when evaluating $\mathcal{G}f$ numerically with the method we present here, the total time spent computing Hankel transforms is negligible compared to the total time spent computing forward Fourier transforms.

3.4. *The inverse Fourier transform* $\mathcal{F}^{-1}$

After a straightforward multiplication by $\mathrm{J}_0(\rho\xi)$, all that is left to compute $\mathcal{G}f$ is the computation of the inverse Fourier transform $\mathcal{F}^{-1}$. As before, we focus here on the case of the one-dimensional inverse Fourier transform defined by

$$u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}(\xi) \mathrm{e}^{\mathrm{i}\xi x} \, \mathrm{d}\xi. \tag{3.11}$$

In principle, this computation can be done with the same numerical tools presented in § 3.2. There are however two key points which we need to revisit. The first point to consider is that we relied on the fact that u was compactly supported (to within the desired numerical accuracy) to reduce the Fourier transform integral to the finite interval I. From the Fourier uncertainty principle (Hardy 1933; Hogan & Lackey 2005), it is not clear that the corresponding Fourier transform has numerical compact support on a finite interval of reasonable size. It may first seem as if this observation strongly limits the class of functions for which $\mathcal{G}f$ can be accurately computed with our numerical scheme. The functions for which both the Fourier transform and the inverse Fourier transform can be computed without significant loss of accuracy by restricting the quadratures to finite intervals with reasonable sizes are functions which can be represented as a Gaussian distribution function plus a small deviation from the Gaussian behaviour (Hardy 1933; Stein & Shakarchi 2003). However, this turns out to be too pessimistic an estimate in practice. This is because our scheme does not require the computation of the inverse Fourier transform of $\mathcal{F}f$, but instead of $\mathrm{J}_0(\rho\xi)\mathcal{H}_0\mathcal{F}f$. For finite ρ, $\mathrm{J}_0(\rho\xi)$ is a decaying function of ξ, so the (numerical) compact support of $\mathrm{J}_0(\rho\xi)\mathcal{H}_0\mathcal{F}f$ is contained within, and typically much smaller than, that of $\mathcal{F}f$. In § 3.5 we provide guidelines for choosing the size $\hat{a}$ of the Fourier domain and the number $\hat{N}$ of the Chebyshev points which we have empirically found to provide accurate results in a robust manner for a wide class of distribution functions, including sub-Gaussian and super-Gaussian distribution functions.

The second point to revisit concerns the numerical method to be used to evaluate (3.11). As we just mentioned, we could rely on the symmetry of the Fourier transform to reuse the methods employed in § 3.2 for the computation of (3.11). This is perfectly acceptable, but since one application of the FST requires the use of approximately four NUFFTs, for a given target accuracy this is not as efficient (Guadagni 2015) as computing (3.11) directly with the NUFFT through the expression

$$u(x_j) \approx \frac{1}{2\pi}\sum_{k=1}^{\hat{N}} w_k \hat{u}(\xi_k)\mathrm{e}^{\mathrm{i}\xi_k x_j}, \tag{3.12}$$

where the w_k are Clenshaw–Curtis weights associated with the Chebyshev grid ξ_k we use in Fourier space, the x_j are the nodes of the real space equispaced grid. The run-time complexity of the inverse Fourier transform computed in this manner is $O((N+\hat{N})\log(N+\hat{N}))$. So for a distribution functions which depend on two spatial variables, it is $O((P+\hat{P})\log(P+\hat{P}))$, with $P\sim N^2$ and $\hat{P}\sim\hat{N}^2$, and since this operation has to be done for every possible value of N_ρ, the overall run time complexity of this step is $O(N_\rho(P+\hat{P})\log(P+\hat{P}))$.

To complete the description of our algorithm, we need to specify how to choose $\hat{a}$ and $\hat{N}$, since the NUFFT gives us the freedom to choose them independently of a and N. This is the purpose of the next section.

3.5. *Size of the computational domain and grid resolution*

In what follows, we assume that $\overline{\rho}$ is the maximum value of ρ considered in the simulation. In other words, $f(x, y, \rho)$ has (numerical) compact support on $[-a/2, a/2]^2 \times [0, \overline{\rho}]$.

3.5.1. *Size and resolution for the spatial domain*

Let $d_{\tilde{f}}$ denote the diameter of the smallest ball in $\mathbb{R}^2$ centred at the origin that contains the compact support of $\tilde{f}$. It can be shown that for fixed ρ, the support of

$\mathcal{G}f(x, y, \rho)$ lies within the annular region $\sqrt{x^2+y^2} \in [\rho - d_{\tilde{f}}/2, \rho + d_{\tilde{f}}/2]$ (Guadagni 2015). We thus take a at least as large as $d_{\tilde{f}} + \overline{\rho}$. The quantity $d_{\tilde{f}}$ depends on time, so it must in principle be recomputed at every time step. However, in many situations of physical interest, $d_{\tilde{f}}$ as determined by the initial data $f(x, y, \rho, 0)$ is a good enough guess for the entire simulation, and does not have to be adjusted at later times. This was for example the case for our gyrokinetic simulations of beam dynamics in cyclotrons, which are dominated by $\boldsymbol{E} \times \boldsymbol{B}$ physics, which led to differential rotation about the guiding magnetic field $\boldsymbol{B}$ but limited radial expansion of the beam distribution (Guadagni 2015).

The number N of grid points in the x- and y-directions may be chosen arbitrarily, provided it is large enough to resolve the spatial variations of the distribution function one is interested in.

3.5.2. *Size and resolution for the Fourier domain*

We now provide simple guidelines for choosing the size $\hat{a}$ of each dimension of the Fourier domain, and the number of grid points in each dimension $\hat{N}$. These guidelines are based on elementary reasoning, which we have nonetheless found to be remarkably reliable for a wide range of distribution functions. Our heuristic argument is as follows. Consider the Gaussian distribution $u(x) = A\mathrm{e}^{-bx^2}$, where A and b are constants. It has numerical support on the interval $I' = [-(a'/2), a'/2]$, where

$$a' = 2\sqrt{-\frac{1}{b}\log\left(\frac{\epsilon^*}{A}\right)}. \tag{3.13}$$

The number ϵ^* is the absolute threshold that defines numerical compact support, and $\epsilon \equiv \epsilon^*/A$ is the relative threshold for compact support. For computations relying on double-precision floating-point format, one may for instance choose $\epsilon = 10^{-16}$. The Fourier transform of u is $\hat{u}(\xi) = A\sqrt{(\pi/b)}\mathrm{e}^{-\xi^2/4b}$, which has numerical compact support on the interval $\hat{I} = [-(\hat{a}/2), \hat{a}/2]$ with $\hat{a}$ given by

$$\hat{a} = 2\sqrt{-4b\log\left(\epsilon\sqrt{\frac{b}{\pi}}\right)}. \tag{3.14}$$

Using (3.13) to express b in terms of a', we obtain the desired formula for the length of the compact support of $\hat{u}$ in terms of the length of the compact support for u:

$$\hat{a}(\epsilon, a') = \frac{8}{a'}\sqrt{\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{a'}{2\epsilon}\sqrt{\frac{\pi}{\log\left(\frac{1}{\epsilon}\right)}}\right)}, \tag{3.15}$$

from which we can see that $\hat{a} \sim (\sqrt{\log a'})/a'$. Although this formula holds exactly only for Gaussian distribution functions, we have empirically found it to be a reliable estimate for a wide range of functions. It is important here to note that in general $a' \neq a$, i.e. the length a' for the compact support of f may be different from the size of the spatial domain a. In principle a' has to be computed at each time step, but in many situations of physical interest a' may not vary much over the course of a simulation.

As an illustration, as was the case for $d_{\tilde{f}}$, we found in our beam dynamics simulations that we were able to keep a' fixed at its initial value determined by $f(x, y, \rho, 0)$ and obtain accurate results. We observe finally that instead of using the formula given by (3.15) to estimate $\hat{a}$, a relatively straightforward (albeit possibly computationally costly) strategy, would be to calculate $\hat{a}$ numerically at each time step, or at regularly spaced time steps. We have not explored such a strategy in our simulations.

For the choice of the grid resolution in Fourier space, specified by $\hat{N}$, we also follow a simple reasoning. The general rule for the proper resolution of waves on a Chebyshev grid is to have at least π Chebyshev nodes per wavelength on average (Trefethen 2000; Marburg 2008; Trefethen 2013). If k is the largest wavenumber associated with the oscillatory signal $\hat{u}$, then we take $\hat{N} = k\hat{a}/2$. Now, it is hard to predict how oscillatory $\hat{u}$ can be in the most general case. We consider the unfavourable scenario in which u is the indicator function $u(x) = \mathbf{1}_{[-(a'/2),a'/2]}(x)$, with Fourier transform $\hat{u}(\xi) = a\,\mathrm{sinc}\,(a\xi/2\pi)$. This function is not smooth enough to be part of the class of functions for which our overall scheme for computing $\mathcal{G}f$ is reliable, but can serve as a good, often conservative estimate of how oscillatory $\hat{u}$ can be. For this function, we find $k \sim a'/2$. Now, accounting for the fact that in our simulations u may not be symmetric with respect to the origin, and accounting for the oscillatory nature of the inverse Fourier kernel, we obtain the estimate $k \sim (3/2)a'$ for the maximum k we may encounter (Guadagni 2015), so that $\hat{N} \sim (3/4)a'\hat{a}$.

3.5.3. *Resolution in ρ-space*

We have already explained that the domain for the ρ-variable is $[0, \overline{\rho}]$, where $\overline{\rho}$ is the smallest value such that $f(x, y, \rho)$ has numerical compact support for all values of x and y in $[-a/2, a/2]^2$. The choice for the number of grid points N_ρ to discretise the interval $[0, \overline{\rho}]$ depends on various considerations, some of which are not related to the computation of $\mathcal{G}f$. It is for example clear that N_ρ should be large enough to properly resolve the details of the ρ-dependence of f. Even if so, we suggest here an initial guess for N_ρ motivated by the need to accurately calculate $\mathcal{G}f$. Our reasoning is again quite elementary, since it does not take into account the oscillatory nature of $\tilde{f}(\xi_x, \xi_y, \rho, t)$ but the outcome has turned out to be a surprisingly robust guideline for choosing N_ρ. The Hankel kernel $\mathrm{J}_0(\rho\xi)\rho$ is an oscillatory function with an approximate wavenumber of $k = \xi$, so the interval $[0, \overline{\rho}]$ has approximately $\overline{\rho}\xi/(2\pi)$ wavelengths. For $(\xi_x, \xi_y) \in [-\hat{a}/2, \hat{a}/2]$, this quantity is maximised when $\xi = \hat{a}/\sqrt{2}$, which means that we need $N_\rho \sim \rho\hat{a}/(2\sqrt{2})$, since we use a Chebyshev grid for the ρ-variable.

3.6. *Summary*

Our numerical scheme for computing $\mathcal{G}f$ is summarised in figure 2. In this figure, we omitted for simplicity the time dependence of all the quantities, since the computation of $\mathcal{G}f$ is done at fixed t. The parallel planes represent the spatial (in blue) or Fourier (in yellow) grids for different values of ρ. The Hankel step $\mathcal{H}$ collapses the N_ρ planes into a single plane, which contains the data corresponding to the Fourier transform of $\tilde{f}$. Multiplying $\mathcal{F}\tilde{f}$ by $\mathrm{J}_0(\rho\xi)$, we recreate N_ρ panels on which we apply the inverse Fourier transform $\mathcal{F}^{-1}$ to obtain the desired $\mathcal{G}f$.

4. Numerical examples

In this section we examine the accuracy and run time of our method by investigating a specific function $f(x, y, \rho)$ for which $\mathcal{G}f(x, y, \rho)$ can be computed analytically. Here,

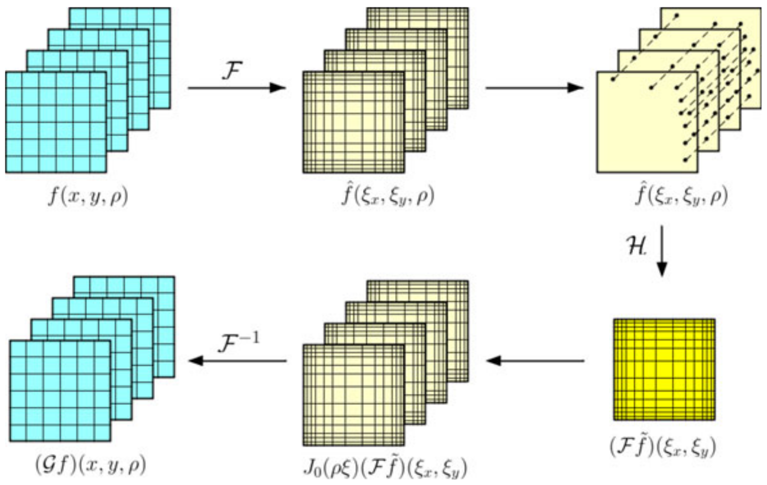
$f(x, y, \rho)$
$\mathcal{F}$
$\hat{f}(\xi_x, \xi_y, \rho)$
$\hat{f}(\xi_x, \xi_y, \rho)$
$\mathcal{H}$
$(\mathcal{F}\tilde{f})(\xi_x, \xi_y)$
$J_0(\rho\xi)(\mathcal{F}\tilde{f})(\xi_x, \xi_y)$
$\mathcal{F}^{-1}$
$(\mathcal{G}f)(x, y, \rho)$

FIGURE 2. Graphical representation of the successive steps involved in the numerical evaluation of $\mathcal{G}f$. The lines in each coloured plane correspond to lines of constant x and constant y and we note the equispaced grid for the spatial variables and the Chebyshev grid in Fourier space. The successive planes correspond to different values of the variable ρ. Since the computation of $\mathcal{G}f$ is done at a fixed time t, we omitted the time dependence in all the formulae in this figure.

we have suppressed the dependence on t for simplicity since the gyroaverages are computed once at each fixed time step. We consider the function

$$f(x, y, \rho) = \mathrm{e}^{-A(x^2+y^2)}\mathrm{e}^{-B\rho^2}, \tag{4.1}$$

where A and B are positive constants. It can be shown that $\mathcal{G}f$ has the following closed-form formula:

$$\mathcal{G}f(x, y, \rho) = \frac{1}{2(A+B)}\mathrm{e}^{-\alpha(x^2+y^2+\rho^2)}\mathrm{I}_0(2\alpha\rho\sqrt{x^2+y^2}), \tag{4.2}$$

where $1/\alpha = 1/A + 1/B$ and $\mathrm{I}_0(z)$ is the modified Bessel function of the first kind of order zero. In § 4.1, we focus on the convergence properties of our scheme, and in § 4.2 we present an analysis of the run times of our code for three levels of resolution: low, medium and high.

4.1. *Accuracy of our scheme*

For the sake of definiteness, in our experiments we put $A = B = 15$. Consistent with what we have already explained, we choose the following fixed interval sizes for the various grids involved in our numerical scheme.

$$\text{real space: } (x, y) \in [-3, 3] \tag{4.3}$$

$$\text{Fourier space: } (\xi_x, \xi_y) \in [-66, 66] \tag{4.4}$$

$$\text{gyroradius: } \rho \in [0, 1.55]. \tag{4.5}$$

We define the sampling rate of a particular grid to be the number of grid points per unit length. For a particular experiment, the sampling rates for the real space grid,

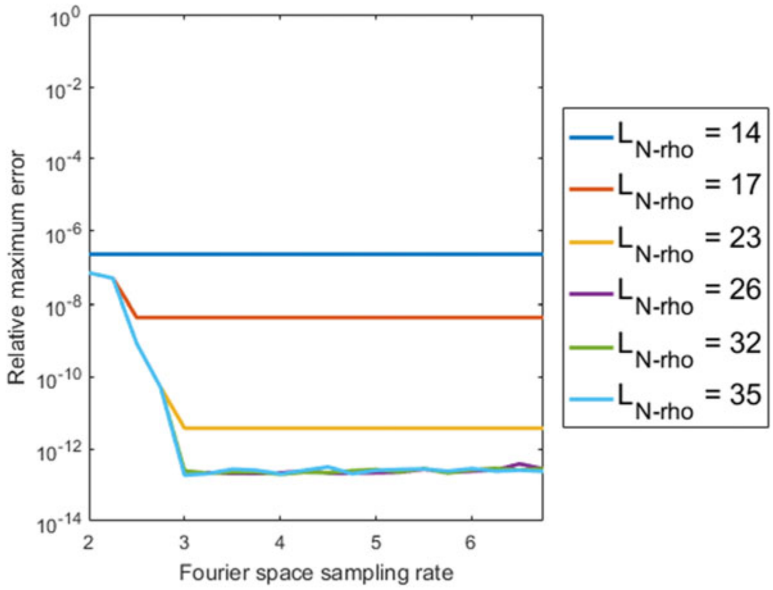

FIGURE 3. Plot of ϵ_{abs} as a function of $L_{\hat{N}}$ and selected values of L_{N_ρ}, for the fixed real sampling rate $L_N = 12$.

Fourier space grid and gyroradius grid are denoted, respectively, L_N, $L_{\hat{N}}$ and L_{N_ρ}. In the language of the previous sections, it follows that $L_N = N/a$, $L_{\hat{N}} = \hat{N}/\hat{a}$ and $L_{N_\rho} = N_\rho/\overline{\rho}$.

Let $\mathcal{G}f^{ex}(x, y, \rho)$ denote the *exact* function in (4.2) and let $\mathcal{G}f^{num}(x, y, \rho)$ denote the numerical approximation to $\mathcal{G}f^{ex}$ obtained by applying our numerical scheme for gyroaveraging to the function $\tilde{f}$ in (4.1). For our purposes, we are concerned with the relative maximum error ϵ_{abs}, defined as

$$\epsilon_{abs} = \frac{\max_{i,j,k} |\mathcal{G}f^{ex}(x_i, y_j, \rho_k) - \mathcal{G}f^{num}(x_i, y_j, \rho_k)|}{\max_{i,j,k} |\mathcal{G}f^{ex}(x_i, y_j, \rho_k)|}, \tag{4.6}$$

where (x_i, y_j, ρ_k) is a grid point in $[-3, 3]^2 \times [0, 1.55]$.

We have calculated and plotted (on a log scale) ϵ_{abs} for a wide range of sampling rates on each grid. In figures 3–8, each pair of side-by-side figures shows ϵ_{abs} for a *fixed* sampling rate of a particular grid. In figures 3–4 we see ϵ_{abs} for a fixed sampling rate on the real space grid, in figures 5–6 we see ϵ_{abs} for a fixed sampling rate on the Fourier space grid and in figures 7–8 we see ϵ_{abs} for a fixed sampling rate on the gyroradius grid. The fixed sampling rates in each case are those explained and suggested previously. The figures clearly show that with the combined suggested choices $L_N = 12$, $L_{\hat{N}} = 4.5$ and $L_{N_\rho} = 35$, the error ϵ_{abs} is of the order of approximately 10^{-13}. Generally, the error decreases exponentially as we increase all three sampling rates uniformly, which is a hallmark of numerical schemes based on spectral methods.

4.2. *Run-time results*

We are also interested in the CPU run time of our code, in particular, the relative run times of the various elements involved (i.e. forward Fourier transform, Hankel

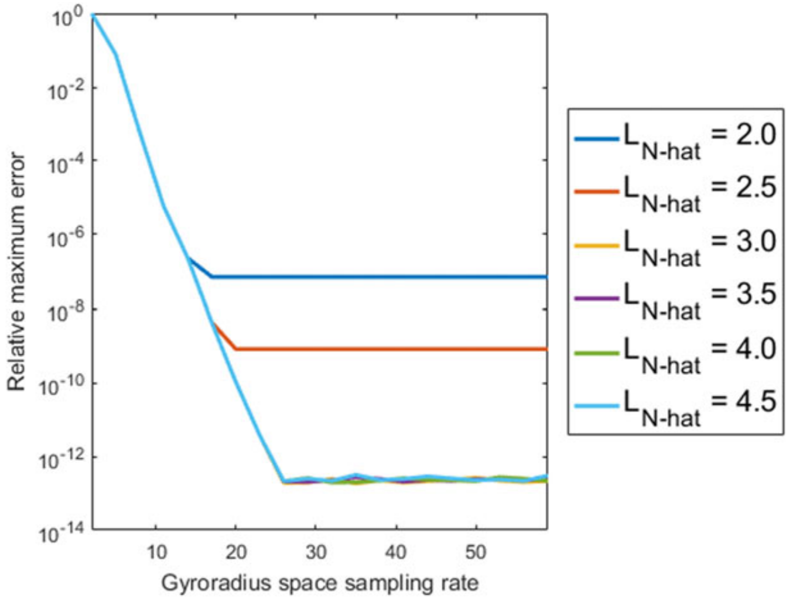

FIGURE 4. Plot of ϵ_{abs} as a function of L_{N_ρ} and selected values of $L_{\hat{N}}$, for the fixed real sampling rate $L_N = 12$.

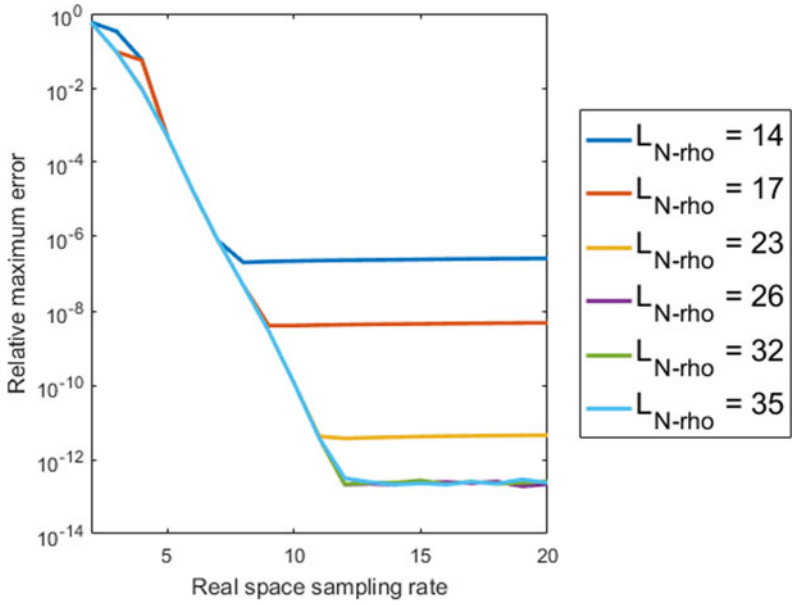

FIGURE 5. Plot of ϵ_{abs} as a function of L_N and selected values of L_{N_ρ}, for the fixed Fourier sampling rate $L_{\hat{N}} = 4.5$.

transform, Bessel function multiplier and inverse Fourier transform). For this purpose, we have run multiple simulations for selected sampling rates for computing the gyroaverage of the function in (4.1), and we have collected our results in table 1.

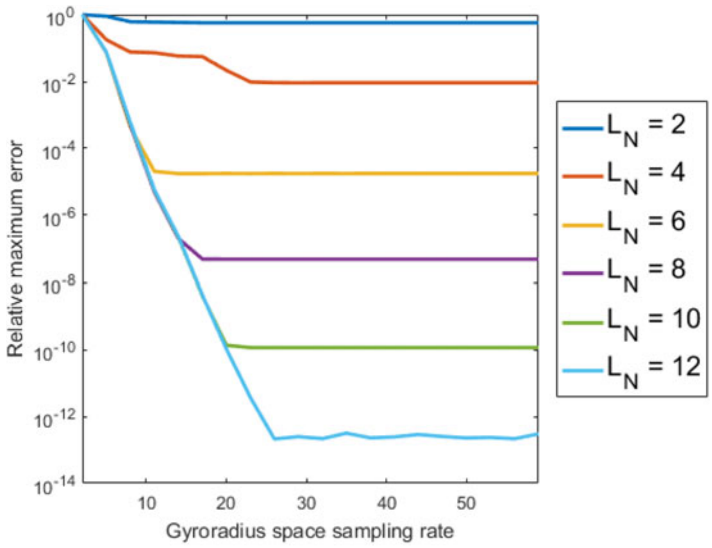

FIGURE 6. Plot of ϵ_{abs} as a function of L_{N_ρ} and selected values of L_N, for the fixed Fourier sampling rate $L_{\hat{N}} = 4.5$.

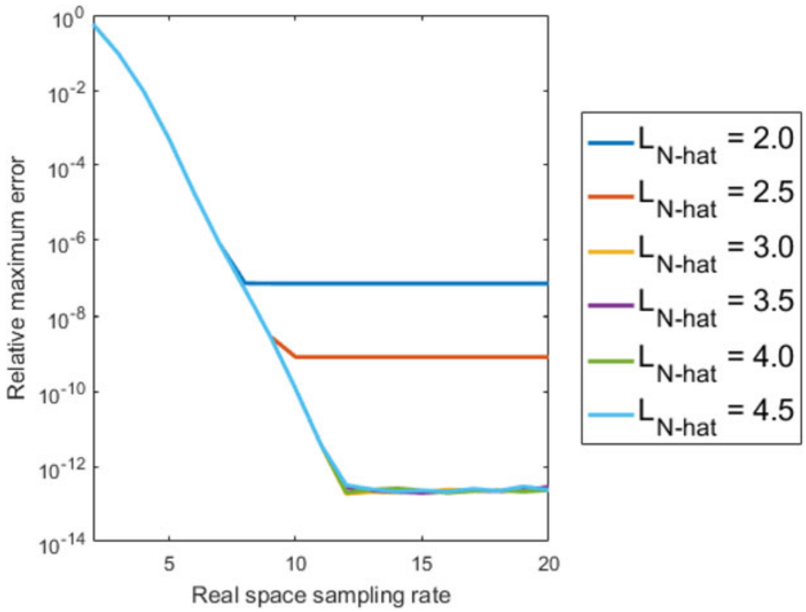

FIGURE 7. Plot of ϵ_{abs} as a function of L_N and selected values of $L_{\hat{N}}$, for the fixed gyroradius sampling rate $L_{N_\rho} = 35$.

For each resolution, the left-hand column corresponds to the time elapsed in seconds and the right-hand column to the time elapsed as a percentage of the total time.

We have separated the run-time data for the specific code elements from the run-time data for computing the various libraries in our code (e.g. Hankel transform weights, Chebyshev grid in Fourier space, etc.). The intended purpose of our

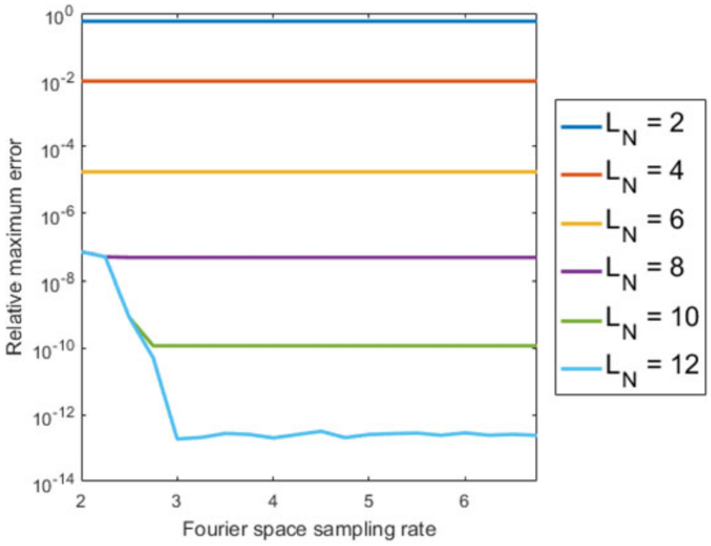

FIGURE 8. Plot of ϵ_{abs} as a function of $L_{\hat{N}}$ and selected values of L_N, for the fixed gyroradius sampling rate $L_{N_\rho} = 35$.

gyroaveraging code is its use in solving the gyrokinetic-Poisson equations. Thus the gyroaveraging code is run at each time step, and so the pre-computed libraries need be computed only once and not at each time step. Thus when analysing the relative time of the code elements, it is not appropriate to include the time taken to build those libraries. Hence the percentage times reported in table 1 are percentages relative to the total time elapsed excluding the time taken to build the libraries.

Although we show the absolute run-time data, given in seconds, for completeness, their importance and relevance are limited. Indeed, we have implemented our scheme in MATLAB, which is an interpreted language and therefore slower than compiled languages such as Fortran and C for the algorithms we rely on, often by an order of magnitude. Furthermore, our results were obtained by running our code on a single core, and we have not yet explored methods for accelerating our schemes through parallelisation.

In contrast, the conclusions we can make from the relative elapsed times of each code element are instructive and as follows. First, we see that the relative elapsed times are approximately the same for all resolutions. Second, the Hankel transform and Bessel function multiplication together take less than 1 % of the total time. This observation is consistent with the fact observed in § 3 that the complexity of the Hankel transform (i.e. $O(N_\rho \hat{P})$) is less than the complexity of the Fourier transforms (i.e. $O(N_\rho(P + \hat{P}) \log(P + \hat{P}))$). The elapsed time is dominated by the time taken by the Fourier transform and the inverse Fourier transform, with the former taking about twice as long as the latter. This difference in time is due to the use of the FST in the forward transform but not in the inverse transform. Finally, we simply note that the computation of the libraries takes about 16 % as long as one iteration of the gyroaveraging code.

Code element	Low resolution		Medium resolution		High resolution	
Pre-computed libraries	0.215	16.4 %	0.792	15.8 %	2.000	15.2 %
Fourier transform	0.835	63.8 %	3.223	64.6 %	8.789	66.7 %
Hankel transform	0.007	0.51 %	0.027	0.53 %	0.070	0.53 %
Bessel multiplier	0.004	0.34 %	0.020	0.41 %	0.070	0.40 %
Inverse Fourier transform	0.463	35.4 %	1.723	34.5 %	4.266	32.4 %
Total time (excl. libraries)	1.309	100 %	4.995	100 %	13.177	100 %
Accuracy	7.3×10^{-2}		4.6×10^{-8}		1.2×10^{-13}	

TABLE 1. For each set of resolutions and for each code element, the CPU run time is reported in two ways: the time elapsed in seconds and the time elapsed as a percentage of the total time (excluding the time required to pre-compute the necessary libraries). The specific sampling rates used for each resolution were $(L_N, L_{\hat{N}}, L_{N_\rho}) = (4, 2.5, 11)$ for 'low resolution', $(L_N, L_{\hat{N}}, L_{N_\rho}) = (8, 3.5, 23)$ for 'medium resolution' and $(L_N, L_{\hat{N}}, L_{N_\rho}) = (12, 4.5, 35)$ for 'high resolution'. The 'high resolution' sampling rates are the sampling rates we have suggested in § 3. These data were obtained by averaging the results of 1000 separate simulations of our code in MATLAB on a home desktop computer operating on an Intel i7-6700k processor at 4.0 GHz.

5. Summary and discussion

Focusing on a simple two-dimensional geometry with a uniform background magnetic field, we have proposed a new numerical scheme for the fast and high-order accurate computation of the gyroaveraged electrostatic potential for the challenging situation in which periodic boundary conditions may not be assumed. Our numerical scheme is based on a reformulation of the gyrokinetic Vlasov–Poisson system in which the electrostatic potential Φ is replaced with a new potential-like function χ which also solves a Poisson equation but depends on the gyroradius variable ρ. The key advantage of this approach is that the function to gyroaverage has compact support. The disadvantage of our approach is that Poisson's equation has to be solved as many times as the number of discretisation points used for the ρ-grid. Since fast Poisson solvers are readily available, and the field solve part of gyrokinetic codes is rarely the main driver of the overall computational cost, this is a reasonable price to pay.

Since the function to gyroaverage has compact support, we can express this gyroaverage as the composition of a Fourier transform, a Hankel transform, a multiplication by the Bessel function J_0 and an inverse Fourier transform, and each operation is numerically well defined. We perform each of these steps using spectrally accurate numerical methods with near optimal run-time complexity. The total run time of our scheme is dominated by the forward and inverse Fourier transforms, so the run-time complexity of our algorithm can be said to be $O(N_\rho(P + \hat{P}) \log(P + \hat{P}))$, where P is the number of spatial grid points, $\hat{P}$ the number of grid points in Fourier space and N_ρ the number of grid points in velocity space. By focusing on examples for which the gyroaverage can be evaluated analytically, we demonstrate that our scheme leads to geometric convergence of the numerical error for gyroaveraging, as expected. For reasonable sampling rates in real space, Fourier space, and in velocity space, we obtain an accuracy of the order of 10^{-13}, close to round off error.

We have successfully applied our scheme to gyrokinetic-Poisson simulations of the dynamics of a non-neutral beam of particles in the median plane of a cyclotron, and verified its high accuracy in this setting (Guadagni 2015). We nevertheless see several directions for further improvement. The scheme may for example be sped up further through parallelisation, and by better optimising the size of the Fourier domain and the grid resolutions $L_{\hat{N}}$ and L_{N_ρ}. The latter could be done either through analytic formulae giving tighter estimates or through automated computations at regularly spaced time steps. One may also wonder if there is an equivalent formulation of our scheme for electromagnetic equations, such as the gyrokinetic Vlasov–Maxwell equations for example. This is a central question for the applicability of our scheme to gyrokinetic simulations for tokamaks, and to beam dynamics studies in which relativistic effects play a significant role. These improvements are the subject of ongoing work, with progress to be reported at a later date.

Acknowledgements

The authors would like to thank Felix Parra for helpful discussions and the referees for valuable suggestions for improving the original manuscript.

REFERENCES

ANTONSEN, T. M. & LANE, B. 1980 Kinetic equations for low frequency instabilities in inhomogeneous plasmas. *Phys. Fluids* **23**, 1205–1214.

BRIZARD, A. J. & HAHM, T. S. 2007 Foundations of nonlinear gyrokinetic theory. *Rev. Mod. Phys.* **79**, 421–468.

CANDY, J. & WALTZ, R. E. 2003 An Eulerian gyrokinetic-Maxwell solver. *J. Comput. Phys.* **186**, 545–581.

CATTO, P. J. 1978 Linearized gyro-kinetics. *Plasma Phys.* **20**, 719.

CATTO, P. J. & TSANG, K. T. 1977 Linearized gyrokinetic equation with collisions. *Phys. Fluids* **20**, 396–401.

CERFON, A. J. 2016 Vortex dynamics and shear layer instability in high intensity cyclotrons. *Phys. Rev. Lett.* **116**, 174801.

CERFON, A. J., FREIDBERG, J. P., PARRA, F. I. & ANTAYA, T. A. 2013 Analytic fluid theory of beam spiraling in high-intensity cyclotrons. *Phys. Rev. Accel. Beams* **16**, 024202.

CROUSEILLES, N., MEHRENBERGER, M. & SELLAMA, H. 2010 Numerical solution of the gyroaverage operator for the finite gyroradius guiding-center model. *Commun. Comput. Phys.* **8**, 484–510.

DAVIS, C. 1962 The norm of the Schur product operation. *Numer. Math.* **4**, 343.

DUTT, A. & ROKHLIN, V. 1993 Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.* **1**, 121.

FRIEMAN, E. A. & CHEN, L. 1982 Nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria. *Phys. Fluids* **25**, 502–508.

GARBET, X., IDOMURA, Y., VILLARD, L. & WATANABE, T. H. 2010 TOPICAL REVIEW: Gyrokinetic simulations of turbulent transport. *Nucl. Fusion* **50**, 043002.

GENOVESE, L., DEUTSCH, T., NEELOV, A., GOEDECKER, S. & BEYLKIN, G. 2006 Efficient solution of Poissons equation with free boundary conditions. *J. Chem. Phys.* **125**, 074105.

GÖRLER, T., LAPILLONNE, X., BRUNNER, S., DANNERT, T., JENKO, F., MERZ, F. & TOLD, D. 2011 The global version of the gyrokinetic turbulence code GENE. *J. Comput. Phys.* **230**, 7053–7071.

GREENGARD, L. & LEE, J.-Y. 2004 Accelerating the nonuniform fast Fourier transform. *SIAM Rev.* **46**, 443.

GREENGARD, L., LEE, J.-Y. & INATI, S. 2006 The fast sinc transform and image reconstruction from nonuniform samples in k-space. *CAMCoS* **1**, 121.

GUADAGNI, J. 2015 Numerical solver for the two-dimensional Vlasov–Poisson equations in gyrokinetic variables. PhD thesis, Courant Institute of Mathematical Sciences, New York University.

HARDY, G. H. 1933 A theorem concerning Fourier transforms. *J. London Math. Soc.* **8**, 227–231.

HAZELTINE, R. D. & MEISS, J. D. 2003 *Plasma Confinement*. Courier Dover Publications.

HOGAN, J. A. & LAKEY, J. D. 2005 *Time Frequency and Time-Scale Methods. Adaptive Decompositions, Uncertainty Principles, and Sampling*. Birkhauser.

JIANG, S., GREENGARD, L. & BAO, W. 2014 Fast and accurate evaluation of nonlocal coulomb and dipole–dipole interactions via the nonuniform FFT. *SIAM J. Sci. Comput.* **36**, B777–B794.

JOLLIET, S., BOTTINO, A., ANGELINO, P., HATZKY, R., TRAN, T. M., MCMILLAN, B. F., SAUTER, O., APPERT, K., IDOMURA, Y. & VILLARD, L. 2007 A global collisionless PIC code in magnetic coordinates. *J. Comput. Phys.* **177**, 409–425.

LEE, J.-Y. & GREENGARD, L. 2005 The type 3 nonuniform FFT and its applications. *J. Comput. Phys.* **206**, 1.

MARBURG, S. 2008 Discretisation requirements: How many elements per wavelength are necessary? In *Computational Acoustics of Noise Propagation in Fluids – Finite and Boundary Element Methods* (ed. S. Marburg & B. Nolte), pp. 309–332. Springer.

NUMATA, R., HOWES, G. G, TATSUNO, T., BARNES, M. & DORLAND, W. 2010 AstroGK: Astrophysical gyrokinetics code. *J. Comput. Phys.* **229**, 9347–9372.

PATAKI, A. & GREENGARD, L. 2011 Fast elliptic solvers in cylindrical coordinates and the Coulomb collision operator. *J. Comput. Phys.* **230**, 7840–7852.

PETERSON, J. L. & HAMMETT, G. W. 2013 Positivity preservation and advection algorithms with applications to edge plasma turbulence. *SIAM J. Sci. Comput.* **35**, B576–B605.

PLUNK, G. G., COWLEY, S. C., SCHEKOCHIHIN, A. A. & TATSUNO, T. 2010 Two-dimensional gyrokinetic turbulence. *J. Fluid Mech.* **664**, 407–435.

RICKETSON, L. F. & CERFON, A. J. 2017 Sparse grid techniques for particle-in-cell schemes. *Plasma Phys. Control. Fusion* **59**, 024002.

RUTHERFORD, P. H. & FRIEMAN, E. A. 1968 Drift instabilities in general magnetic field configurations. *Phys. Fluids* **11**, 569–585.

SARAZIN, Y., GRANDGIRARD, V., FLEURENCE, E., GARBET, X., GHENDRIH, P., BERTRAND, P. & DEPRET, G. 2005 Kinetic features of interchange turbulence. *Plasma Phys. Control. Fusion* **47**, 1817–1839.

SHU, C.-W. 1999 High order ENO and WENO schemes for computational fluid dynamics. In *High-Order Methods for Computational Physics* (ed. T. J. Barth & H. Deconinck). Springer.

STEIN, E. M. & SHAKARCHI, R. 2003 *Complex Analysis*, Princeton Lectures in Analysis, vol. II.. Princeton University Press.

STEINER, C., MEHRENBERGER, M., CROUSEILLES, N., GRANDGIRARD, V., LATU, G. & ROZAR, F. 2015 Gyroaverage operator for a polar mesh. *Eur. Phys. J.* D **69**, 18.

TAYLOR, J. B. & HASTIE, R. J. 1968 Stability of general plasma equilibria. Part I. Formal theory. *Plasma Phys.* **10**, 479–494.

TREFETHEN, L. N. 2000 *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics.

TREFETHEN, L. N. 2008 Is Gauss quadrature better than Clenshaw–Curtis? *SIAM Rev.* **50**, 67.

TREFETHEN, L. N. 2013 *Approximation Theory and Approximation Practice*. Society for Industrial and Applied Mathematics.

TREFETHEN, L. N. & WEIDEMAN, J. A. C. 2014 The exponentially convergent trapezoidal rule. *SIAM Rev.* **56**, 385.

VICO, F., GREENGARD, L. & FERRANDO, M. 2016 Fast convolution with free-space Green's functions. *J. Comput. Phys.* **323**, 191–203.

WILKENING, J., CERFON, A. J. & LANDREMAN, M. 2015 Projected dynamics of kinetic equations with energy diffusion in spaces of orthogonal polynomials. *J. Comput. Phys.* **294**, 58–77.