Highly efficient numerical algorithm based on random trees for accelerating parallel Vlasov-Poisson simulations

Juan A. Acebrón [a]

[a] *Department of Information Science and Technology, ISCTE-Instituto Universitario de Lisboa, Av. Forças Armadas, 1649-026 Lisboa, Portugal*

and

Ángel Rodríguez-Rozas [b]

[b] *Laboratoire de Mathématiques et de leurs Applications de Pau (LMA-PAU) CNRS : UMR5142 Université de Pau et des Pays de l'Adour Center for Mathematics and its Applications, France*

Abstract

An efficient numerical method based on a probabilistic representation for the Vlasov-Poisson system of equations in the Fourier space has been derived. This has been done theoretically for arbitrary dimensional problems, and particularized to unidimensional problems for numerical purposes. Such a representation has been validated theoretically in the linear regime comparing the solution obtained with the classical results of the linear Landau damping.

The numerical strategy followed requires generating suitable random trees combined with a Padé approximant for approximating accurately a given divergent series. Such series are obtained by summing the partial contributions to the solution coming from trees with arbitrary number of branches. These contributions, coming in general from multi-dimensional definite integrals, are efficiently computed by a quasi-Monte Carlo method. It is shown how the accuracy of the method can be effectively increased by considering more terms of the series.

The new representation was used successfully to develop a probabilistic domain decomposition method suited for massively parallel computers, which improves the scalability found in classical methods. Finally, a few numerical examples based on classical phenomena such as the non-linear Landau damping, and the two streaming instability are given, illustrating the remarkable performance of the algorithm, when compared the results with those obtained using a classical method.

Key words: Monte Carlo and quasi Monte-Carlo methods, domain decomposition, Vlasov-Poisson system, parallel computing, scalable algorithms, random trees

1 Introduction

Traditionally, when developing numerical methods for solving numerically partial differential equations (PDEs), a major importance has been given to concepts such as convergence, consistency and stability, as well as more recently, to the conservation by the numerical scheme of certain invariants the model equations are known to preserve. In general a secondary importance is given to the computational cost of the associated algorithms, and often much less still to the advantageous features these algorithms may exhibit or not when they are run in parallel computers.

It is easily understood this lack of research, since historically and until relatively recently, the theoretical performance of computers, being purely sequential, was progressively increased at a pace closely governed by the well known Moore's Law. Therefore, it was no needed to adapt the existing numerical methods to the recent newborn technologies, since the performance in terms of computational time was automatically reduced every 18 months. However, with the advent of parallel architectures this does not occur anymore, due basically to the communication and synchronization overhead when parallelizing the sequential codes. Nowadays, the situation became even more dramatic. In fact, the most advanced computing systems in operation to date are composed of hundreds of thousands of processors, and soon millions of processors will come into operation, reaching the Exascale performance regime [19,31,39].

The consequences of an inefficient use of these computing resources affect not only the computational time of the algorithms, but also makes impossible to tackle computational problems whose complexity reaches levels requiring the full use of the available resources. This is because in extreme cases, the number of processors involved in the computation should be paradoxically reduced to avoid increasing the computational time of the algorithms. The major problem is the routine use of computational meshes when solving numerically a given problem. Since the mesh is a numerical tool connecting globally the discretized domain, this induces an unavoidable communication among the processors involved when the numerical method is parallelized. It is worth to point out that such a communication overhead acts always negatively degrading the performance of the algorithms [6,19]. This turns out to be particularly

Email addresses: juan.acebron@iscte.pt (Juan A. Acebrón), angel.rodriguez.rozas@gmail.com (Ángel Rodríguez-Rozas).

dramatic when using a large number of processors. That is why it is relevant to investigate on new numerical methods satisfying not only the usual requirements for convergence, consistency, and stability, but also to provide a more efficient use upon the resources available, thus adapting the methods to the current supercomputing architectures.

This is precisely the goal that guided our previous works [1,2,4,5]. There, the foundations of a new parallel algorithm for solving partial differential equations were laid down, which notably improved the scalability of classical algorithms. The ultimate goal was not to replace, but rather to complement, the existing algorithms by introducing new information about the solution obtained at single points within the computational domain. This alleviated significantly the intercommunication overhead of the algorithms, improving overall the scalability and efficiency when they run in parallel architectures. This timely information is obtained through the probabilistic representation of the solution. For linear equations this is based on the celebrated Feynman-Kac formula [25,27,36], while for semilinear equations requires a suitable generalization of it based on branching diffusion equations processes [32,4,5,15,37].

In practice, this representation allows to compute numerically the solution in a single point in space and time using Monte Carlo simulations. It is therefore a method that does not require any computational mesh, and since it is based on Monte Carlo simulation is specially suited to be parallelized. However, since Monte Carlo based methods suffers from slow convergence, and therefore exhibits a high computational cost, they are not recommended to be used in the full computational domain, but rather at specific points in it. These points are typically chosen as interpolation nodes at internal interfaces inside the domain. The interpolated solution along those interfaces allows to obtain the sought boundary conditions, splitting in such a way the given domain in independent subdomains. The problem can be solved then on each of the subdomains resorting to any classical numerical scheme for solving PDEs. This new algorithm has been called Probabilistic Domain Decomposition (PDD), because it combines a probabilistic method for computing the solution at specific points, and the classical domain decomposition [14,35]. In principle, it can be applied to any problem, provided a probabilistic representation of the solution can be found. In previous works it has been applied successfully to both elliptic [1,2] and parabolic equations, linear and semilinear [3–5], but never to transport equations of any type.

To illustrate the potentiality behind these probabilistic techniques for solving numerically even transport equations, this paper has been particularized to the Vlasov-Poisson system of equations. The main reason is twofold: First such an equation is of unquestionable interest in Plasma Physics, and second, it arises as an excellent benchmark for testing numerical methods. This is because the solution behaves in such a way that leads to important technical

difficulties for the numerical methods. More specifically, these are the problem of filamentation, which requires very fine grid to approximate accurately the solution for long times, and the typical high dimensionality of the discretized problem, since the solution of such system of equations depends on both, space and velocity as independent variables. The latter can be specially severe, when solving accurately realistic 3D problems, since the computational mesh needed by the classical numerical method, results in extremely large matrices, which in turn demands the use of large computational infrastructures equipped with large distributed memory. This leads to intercommunication overhead, being particularly important when a large number of processors are involved, and thus degrading the theoretical performance of the algorithms.

Over the last few years many numerical methods have been proposed for solving the Vlasov-Poisson system of equations. There are already excellent reviews in the literature describing the different numerical methods proposed so far (see [22].e.g.), therefore it is not intended to go into any details here, simply describing briefly some of them for those readers not familiar with such a topic. Roughly the numerical methods are divided into two types: those corresponding to classical particle methods, such as Particle-In-Cell (PIC) method [10], and those based on the continuum description such as semi-Lagrangian [41,16,17], finite volume [23], discontinuous Galerkin [38,26], etc. Among the last ones, there are also those based in solving the problem in the Fourier space [28,29].

Particle methods intend to describe the plasma by a finite number of macro-particles whose trajectories are the characteristic curves of the Vlasov equation, and the electric field is typically calculated by discretizing the physical space by means of a computational mesh. It consists therefore of an approximation of the distribution function, being the error of statistical nature. The other methods instead, they solve the system of equations discretizing conveniently and using a suitable computational mesh, in some cases for both equations, and in others only for the Poisson equation.

In this paper, we have considered merely periodic boundary conditions in space. This has been done for simplifying as much as possible the probabilistic representation of the solution, and thus to help the reader to understand easily such a representation. It is theoretically well-known [27,33], that dealing with general boundary conditions requires estimating various stochastic quantities, which in turn introduces new sources of numerical errors, that we tried to avoid at this stage. The ultimate goal of this paper is to show the feasibility of this method as a complementary method capable to speed up the Vlasov-Poisson simulations in a parallel environment, and this was done simplifying the nature of the boundary data as much as possible. The generalization of the method to situations where more complicated geometries and boundary conditions are imposed, is left for a future work.

Being the boundary conditions periodic in space, it becomes natural to solve numerically the problem in Fourier space for the spatial coordinates. Moreover, it turns out that for dealing accurately with the filamentation phenomenon, it is convenient to analyze also the problem in Fourier space for velocities [21]. That is why in practice the numerical method to be presented in this paper is fully analyzed in Fourier space. Apart from such a mathematical reason, in some practical experimental situations one could be interested not to know the distribution function, but rather the spectrum energy or any other related quantities, being therefore natural to investigate the problem in Fourier space. Furthermore, while the probabilistic representation introduced in this paper was obtained in Fourier space, there already exists representations in configuration space [42], which may potentially be used to generalize further what has been done in this paper.

It is worth to remark that such a probabilistic representation, on which the numerical algorithm presented in this paper is based, differs completely from the PIC method. For the PIC numerical method, the initial condition is discretized by introducing suitable macro-particles. The statistical nature of errors can be specially severe affecting largely the tail of the distribution, where typically there are few particles. Other than that, it is necessary to introduce a computational mesh to solve the Poisson equation. Our probabilistic representation is essentially a way of representing the analytical solution of the problem, and therefore it is free from any numerical errors. Errors appear so far when computing numerically the definite integrals, which are part of the numerical method. When such integrals are computed by Monte Carlo, the numerical error turns out to be also of statistical nature, but rather than in the PIC method, this can be individually reduced point by point where the solution is computed. Therefore, it is not a global approximation of the distribution function by particles evolving in time, but rather a local pointwise approximation of the distribution function.

The purpose of this paper is to show that we can accelerate notably Vlasov-Poisson simulations, improving the scalability of classical algorithms, when those are run in parallel. This is done resorting to a probabilistic representation of such equations based on generating random trees, which makes it possible to compute the solution at single points inside the computational domain. Furthermore, this allows to decouple the given problem into independent subproblems, significantly reducing the communication cost of the classical algorithms. It consists therefore of an application of the PDD method to the solution of the Vlasov-Poisson system. Finally, and in order to test the efficiency of our algorithm, several examples traditionally used for verifying the numerical methods developed for the Vlasov-Poisson system, were run for large-scale problems and large number of processors, comparing the efficiency obtained with a classical algorithm.

The paper is organized as follows. The probabilistic representation for a certain class of transport equations is presented in Sec. 2. Sec. 3 concerns the probabilistic representation for the Vlasov-Poisson system of equations. Here such a representation is derived in the Fourier space for arbitrary dimensions, and the validation of the representation is done analyzing the classical linear Landau damping. In Sec. 4, it is explained how the probabilistic representation can be used in practice, and which are the associated numerical errors. The PDD method is presented in Sec. 5. First the algorithm is described, and analyzed the numerical error, then the computational cost is estimated, and finally several numerical tests consisting in typical problems considered often in the literature are given, focusing in both, the accuracy and performance of the method. To conclude we summarize the main results and discuss potential directions for future research.

2 A probabilistic representation for a class of transport equations

The purpose of this section is to illustrate in simple unidimensional problems how the probabilistic representations can be obtained for semilinear transport equations. Recall that the Vlasov-Poisson system of equations is composed by the transport Vlasov equation coupled with the Poisson equation. Therefore, it is convenient to describe first how a probabilistic representation can be derived for transport equations.

Similarly to the probabilistic representation for semilinear parabolic PDEs, a probabilistic representation for semilinear transport equations does exist. In the following, and for the purpose of illustration, the procedure to obtain such a representation is carefully explained for a very simple case. The most important difference with the parabolic case rests in the absence of the elliptic operator governing the diffusion process. Consequently, it cannot be defined as the generator of a suitable stochastic process, solution of a given Ito stochastic differential equations, and consequently the celebrated Feynman-Kac formula for the probabilistic representation of the solution does not apply. However, two different probabilistic representations can be proposed in terms of the characteristic curves of the transport equation. In the following, by probabilistic representation of the solution we mean an explicit solution of the problem in terms of the given initial and/or boundary data, obtained through a Monte Carlo-type method. It is important to remark that for hyperbolic problems in general, the characteristic curves play a similar role in such a representation as the stochastic process does for the parabolic problems.

Let consider the following initial value problem for a semilinear transport equation,

$$\frac{\partial u}{\partial t} + v\,\frac{\partial u}{\partial x} = c(x,t)\,u^2, \quad u(x,0) = f(x) \tag{1}$$

The power of the nonlinearity has been fixed intentionally to two to mimic somehow the nonlinearity found in the Fourier transformed Vlasov-Poisson system, which in fact it is given by a convolution product of the density function with itself (see Section 3). Using the characteristic curves, the equation above can be transformed into the following semilinear integral equation:

$$u(x,t) = f(x - v\,t) + \int_0^t ds\, c(x - v\,s, t - s)\, u^2(x - v\,s, t - s), \tag{2}$$

which can be solved recursively, and yields

$$\begin{aligned}
&u(x,t) = f(x - v\,t) + f^2(x - v\,t) \int_0^t ds_0\, c(x - v\,s_0, t - s_0) \\
&+2f^3(x - v\,t) \int_0^t ds_0 \int_0^{t-s_0} ds_1\, c(x - v\,s_0, t - s_0) c(x - v\,(s_0 + s_1), t - (s_0 + s_1)) \\
&+f^4(x - v\,t) \left[\int_0^t ds_0, c(x - v\,s_0, t - s_0) \left(\int_0^{t-s_0} ds_1\, c(x - v\,(s_0 + s_1), t - (s_0 + s_1)) \right)^2 \right. \\
&+4 \int_0^t ds_0 \int_0^{t-s_0} ds_1 \int_0^{t-s_0-s_1} ds_2\, c(x - v\,s_0, t - s_0)\, c(x - v\,(s_0 + s_1), t - (s_0 + s_1)) \\
&\left. \times c(x - v\,(s_0 + s_1 + s_2), t - (s_0 + s_1 + s_2)) \right] + \cdots
\end{aligned} \tag{3}$$

It is worth to observe that such an expansion of the solution can be used for numerical purposes, computing numerically all the coefficients which appear multiplying the powers of the initial function $f(x - v\,t)$. In fact, defining

$$\alpha := \sup_{x \in (-\infty, \infty)} \sup_{\tau \in 0, t} |c(x, \tau)|, \tag{4}$$

and

$$\beta := \sup_{x \in (-\infty, \infty)} |\mathrm{f}(x)|, \tag{5}$$

it follows that the expansion in (6) is bounded as

$$|u(x,t)| \leq \beta + \beta^2 \alpha t + \beta^3 \alpha^2 t^2 + \beta^4 \alpha^3 t^3 + \cdots . \tag{6}$$

Note that in practice this ensures the convergence of the expansion provided that the time t is chosen sufficiently small. The same argument can be similarly applied for the case of the Vlasov-Poison system of equations, since the

expansion of the solution is equally bounded, as it will be shown later on. This fact justifies the use of the expansion as a numerical method, being the infinite series conveniently truncated. For this purpose, using a Padé approximant might be convenient either to accelerate the convergence of a convergent series or to assymptotically approximate the solution in case of dealing with a divergent series, when the time t or the initial condition is chosen sufficiently large, as it will be explored later on.

Since the coefficients are definite integrals, those can be even treated probabilistically simply by computing them using a Monte Carlo method. Indeed the definite integrals can be conveniently replaced by expected values of suitable random variables. Clearly the definite integrals can be computed numerically resorting to classical quadrature methods as well, however it is well known that for high-dimensional integrals any quadrature method is affected by the *curse of dimension*. Note that this is specially severe when computing the higher terms appearing in the series expansion (6). Therefore, a possible numerical algorithm may consist in computing all the coefficients by Monte Carlo, and summing conveniently the series expansion to obtain numerically an approximation to the solution. It is worth to remark that the method allows to compute the solution at a single point (x, t).

Since for numerical purposes the infinite series should be truncated, this will lead to a truncation error, which will depend merely on the properties of the neglected coefficients. From Eq. (6) it can be seen that such coefficients are obtained through definite integrals, where the time t appears as one of the integration limits. The dimension of the integrals increases for higher terms, and in turn the minimum power of t obtained when expanding such coefficients in powers series of t around $t = 0$. Therefore, it is expected that including higher terms will improve the approximation for long times, reducing the truncation error accordingly. However, this cannot be done freely, since the computational cost to compute numerically such coefficients by Monte Carlo, will grow dramatically for increasingly higher terms. This is due to the high dimensionality of the integrals involved, which in practice entails a large variance, and thus large statistical error, requiring to increase conveniently the chosen sample size.

It is worth to observe that since the series should be truncated for numerical purposes, this precludes the use in practice of such probabilistic representations globally in time, being necessary to adopt some restarting procedures from time to time, choosing the suitable time interval according to the specific problem. This will discussed further in Sec. 4 for the particular case of the Vlasov-Poisson system of equations.

Moreover, a major problem appears with this representation, when trying to enumerate the different integrals of the series expansion. Note that as we

consider the higher terms in the expansion, the different types of possible integrands grow very fast, therefore requiring to perform a prior analytical treatment to obtain all of them. This can be extremely costly, and unless one is interested to obtain a suitable approximation to the solution computing only few coefficients in the expansion, such a numerical algorithm could be quite inefficient in practice. In the following we will refer to this algorithm as an algorithm based on a *static* representation of the solution.

An alternative probabilistic representation of the solution does exist. Indeed, Eq. (2) can be rewritten probabilistically as follows,

$$\begin{aligned} u(x,t) &= E\left[\tilde{f}(x-v\,t)\delta(\zeta)\right] \\ +E\left[\eta(t)\tilde{c}(x-v\,(t-S),S)\,u^2(x-v\,(t-S),S)\,\delta(\zeta-1)\right]. \end{aligned} \tag{7}$$

Here E denotes the expected value taken with respect to the probability function $P(S) = 1/t$, being therefore S a random variable uniformly distributed between 0 and t, and the two-point discrete random variable ζ taking the values 0, and 1, with probability $P(0) = q$, $P(1) = 1 - q$. Here $\tilde{f}(x) = f(x)/q$, $\tilde{c}(x,t) = c(x,t)/(1-q)$, and $\eta(t) = t$. Not all values of q are allowed, since only a set of them gives rise to a valid probability function, see [3] e.g.

In practice, Equation (7) can be visualized as follows: Toss a coin governed by a probability function $P(0) = q$, $P(1) = 1 - q$, and according to the outcome (being 0 or 1 with probability q, and $1 - q$, respectively), evaluate the first term or second term of the right hand side, respectively. In case of the outcome being 1, a random time S uniformly distributed between 0 and t is generated. Since the second term is still unknown, because it depends on the solution of the equation, this representation is useless, unless a further expansion of the solution is done. Indeed, the equation above can be recursively expanded, and yields

$$\begin{aligned} &u(x,t) = E\left[\tilde{f}(x-v\,t)\,\delta(\zeta_0)\right] \\ &+E\left[\eta(t)\tilde{c}'(x-v\,S_0,t-S_0))\tilde{f}^2(x-v\,t)\,\delta(\zeta_0-1)\delta(\zeta_1)\delta(\zeta_2)\right] \\ &+E\left[\eta(t)\eta(t-S_0)\tilde{c}'(x-v\,S_0,t-S_0)\right. \\ &\times\tilde{c}'(x-v\,(S_0+S_1),t-S_0-S_1)\tilde{f}^3(x-v\,t) \\ &\left.\delta(\zeta_1-1)\delta(\zeta_0-1)\delta(\zeta_2)\delta(\zeta_3)\delta(\zeta_4)\right]+\cdots \end{aligned} \tag{8}$$

Defining a suitable random tree, the solution above can be compacted formally as follows,

$$u(x,t) = E\left[\prod_{i=1}^{Ne(\omega)} \eta(t-\bar{S}_i(\omega))\tilde{c}(x-v\,\bar{S}_i(\omega))\tilde{f}(x-v\,t)\right]. \tag{9}$$

where the expected value of the multiplicative functional is taken with respect to a suitable distribution function of random trees. Here $Ne(\omega)$ is the random number of splitting events obtained when generating the random tree. By $\bar{S}$ we denote the corresponding global random time obtained by summing conveniently the random times S_i according to the specific structure of the generated random tree. It is worth to observe that such trees are used as a tool to construct the particular structure representing a given partial contribution to the solution, allowing afterward to follow easily how the arguments of the functions $\tilde{c}$ and η are modified when solving recursively (8). Note that within this formalism, the ith coefficient in the series expansion (9) corresponds to the partial contribution to the solution coming from random trees composed by i splitting events.

For a graphical illustration of such a probabilistic representation, in Fig. 1 it is depicted the two different configuration diagrams of random trees corresponding to two splitting events. In this case, two different diagrams can be generated, despite the resulting integrals are eventually identical due to the underlying symmetries. It is only when three or more splitting events occurs, that different integrals will effectively appear coming from different configuration diagrams. In [3], the number of possible configurations N_D in terms of the number of splitting events Ne was already derived, and is given by

$$N_D(Ne) = \frac{1}{2Ne+1}\begin{pmatrix} 2Ne+1 \\ Ne \end{pmatrix}.$$

Note here that N_D grows really fast with the Ne, and so does the number of different integrals required for computing the solution probabilistically using the static representation in (6). Therefore, deriving at hand all the different integrals required to evaluate the corresponding coefficients of the series expansion, as proposed for the static probabilistic representation, is indeed a tour de force, and of limited use for developing an efficient computational algorithm.

The alternative method based on the probabilistic representation derived in (7), allows rather to develop an algorithm capable to generate automatically such integrals dynamically, simply letting to expand the recursive equation probabilistically governed by the random variables previously defined. This gives rise to a random tree, that can be labeled in the following way resorting to Fig. 1 for illustration. Let the point (x,t) be the root of the trees, denote by a star symbol in the Figure, and the nodes of the trees are labeled with a number 1 when a splitting event occurs (the corresponding ζ randomly

generated in (7) is 1), and 0 otherwise (ζ is now 0). For both diagrams shown in the Figure, initially a splitting event took place, thus leading to two new branches, and a single coupling term $\tilde{c}'$ whose arguments are given by the value $(x - vS_0, t - S_0)$, being S_0 the random time associated to this splitting event. Two different scenarios can occur now, and they are,

- For the diagram (a), the leftmost branch splits again, leading to a new coupling coefficient and two new branches with arguments given by $(x - v(S_0+S_1), t-(S_0+S_1))$, being S_1 a new time randomly chosen associated this time to the last splitting event. Finally these two branches do not split further and neither does the rightmost branch generated during the first splitting event. Such a tree can be uniquely characterized by tracking forward the sequence of splitting events following a pre-order traversal of the tree, commonly known as depth-first in graph theory. In fact, recursive programming is the more natural choice to implement in practice such an algorithm. Thus, this tree is given by the unique sequence 11000;
- For the diagram (b), this tree is given instead by the sequence 10100.

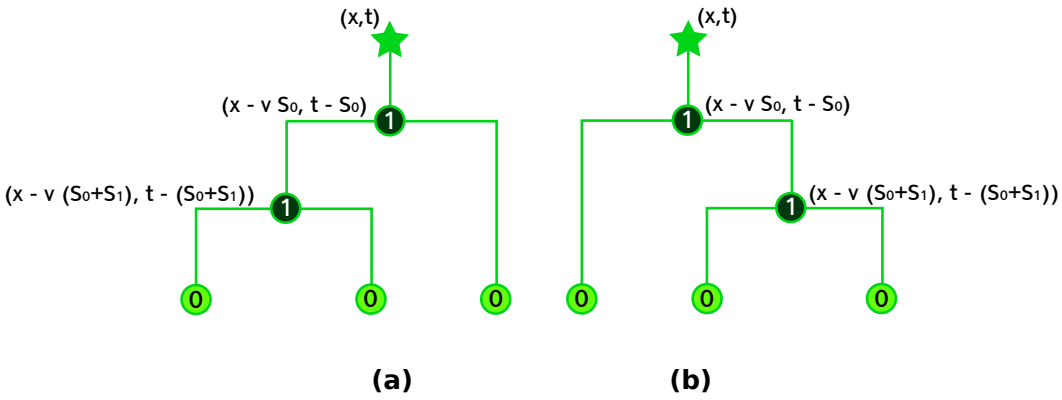


Fig. 1. Two different configuration diagrams for random trees with two splitting events, (a) and (b), uniquely characterized by the sequences 11000 and 10100, respectively.

The numerical algorithm based on this approach will be called in the following a *dynamic* probabilistic representation. As an important advantage compared with the static approach, the dynamic algorithm allows to compute the coefficients of the series expansion without the need of deriving previously analytically the integrands of the definite integrals. However, the price to pay is that we cannot sample independently (choosing a given sample size N) each of the integrals in the series expansion (8) as in the static approach, being rather determined by the probability of finding a random tree composed by a certain number of splitting events. Since the probability of obtaining such trees decreases with the number of splitting events, it turns out that the statistical error done when computing numerically the coefficients should be larger for the higher order terms of the series expansion.

Yet, it is possible to combine both approaches taking advantage of the best of both representations as follows: Instead of generating randomly the splitting events characterizing the tree, dynamically, without prior knowledge of the previous result, those can be statically pre-computed resorting to a simple algorithm based on the labeling of the trees explained above. For example, for $Ne = 1$ only one configuration diagram exists, given by 100; and for $Ne = 3$, the following five, 1110000, 1101000, 1100100, 1011000 and 1010100. In general given Ne_{max} splitting events, and according to the proposed depth-first order labeling strategy, it is possible to know in advance the number and type of the different configuration diagrams, as well as reconstructing the corresponding integrands for evaluating the integrals. This can be done without the need of generating them random and dynamically, or deriving them first analytically, as it happens in the static approach. Moreover, since all the different integrals necessary to be computed by Monte Carlo are known a priori, it is possible to select individually a particular sample size to compute each of them, allowing in such a way to uniformly distribute the statistical error among the different coefficients of the series expansion. The algorithm based on such idea will be called in the following the *hybrid* approach.

Although the series expansion in (8), and the related compact form in (9) can be used for computing probabilistically the solution to the PDE problem, a major drawback appears when dealing with arbitrary large initial data, coefficients, and time, since the series may in general be divergent and thus it cannot be summed simply by a sequence of partial sums. In [3], a Padé approximant has been proposed for approximating the asymptotic series obtained when solving probabilistically a semilinear parabolic problem. The Padé approximant often gives better approximation of the function than truncating the given series, and it may still work where the series does not converge. In fact, the Padé approximant is the best rational approximation of a power series, allowing to attain an efficient analytic continuation of the series beyond its circle of convergence. However, being a rational expression, has poles, which are not present in the original function, and therefore special care should be taken when used in practice [7,8].

3 Probabilistic representation for the Vlasov-Poisson equation

Briefly, Vlasov equation describes the temporal evolution of charged particles subject to the self-generated electric field created by the charged particles inside the plasma. It is actually a system of equations, consisting of a kinetic equation, the Vlasov equation which describes the transport of charged particles, along with the classical Poisson equation for electrostatic potential. The solution of the equation is the distribution function of particles in the phase space, where the independent variables are space, x, velocity, v, and time

t. Consider the two-species Vlasov-Poisson equation in d dimensions conveniently adimensionalized,

$$\begin{aligned}
\frac{\partial f^{(1)}}{\partial t} + \bar{v}\cdot\nabla_{\bar{x}} f^{(1)} - \nabla\phi\cdot\nabla_{\bar{v}} f^{(1)} &= 0,\\
\frac{\partial f^{(2)}}{\partial t} + \bar{v}\cdot\nabla_{\bar{x}} f^{(2)} + \frac{1}{m_2}\nabla\phi\cdot\nabla_{\bar{v}} f^{(2)} &= 0,\\
\triangle_{\bar{x}}\phi = -\left[\int f^{(1)} d\bar{v} - \int f^{(2)} d\bar{v}\right],&
\end{aligned} \tag{10}$$

along with a 2π-periodic boundary condition for the space variables, $f^{(i)}(\bar{x},\bar{v},t) = f^{(i)}(\bar{x}+2\pi,\bar{v},t)$, decay to zero as $|\bar{v}|\to\infty$ with sufficiently high rate for velocity variables, and suitable initial conditions for both, $f^{(i)}$, and the space average over a period of the electric field $\bar{E} = -\nabla\phi$. In [30] it has been proved that in order the Vlasov-Poisson equations provide a complete description of the plasma, such a quantity should be kept fixed to zero for all time. Finally, being $f^{(i)}$ a density probability, it holds that $\int_0^{2\pi}\int_{-\infty}^{\infty} d\bar{x}\, d\bar{v}\, f^{(i)} = 1$.

Since the prescribed boundary condition for space variables are periodic, it is more natural to analyze the problem in Fourier space. Moreover, it turns out to be more convenient to transform as well to the Fourier space for velocities, in order to mitigate the well known filamentation effect in velocity space observed in the solution for sufficiently long times [21]. Because of the periodicity in the space variables, the transformation in space is discrete, while for velocities is continuous. Then, transforming Eq. (10), yields

$$\begin{aligned}
\frac{\partial F^{(1)}_{\bar{k}}}{\partial t} - \bar{k}\cdot\nabla_{\bar{\xi}} F^{(1)}_{\bar{k}} - \bar{k}\cdot\bar{\xi}\,\hat{\phi}_{\bar{k}} * F^{(1)}_{\bar{k}} &= 0,\\
\frac{\partial F^{(2)}_{\bar{k}}}{\partial t} - \bar{k}\cdot\nabla_{\bar{\xi}} F^{(2)}_{\bar{k}} + \frac{1}{m_2}\bar{k}\cdot\bar{\xi}\hat{\phi}_{\bar{k}} \star F^{(2)}_{\bar{k}} &= 0,\\
-|\bar{k}|^2\hat{\phi} = -\left[F^{(1)}_{\bar{k}}(0,t) - F^{(2)}_{\bar{k}}(0,t)\right],&\quad |\bar{k}|\neq 0,
\end{aligned} \tag{11}$$

where

$$F^{(i)}_{\bar{k}}(\bar{\xi},t) = \int_{\mathbb{R}^d} d\bar{v}\int_0^{2\pi} d\bar{x}\, e^{-i\bar{\xi}\cdot\bar{v}} e^{-i\bar{k}\cdot\bar{x}} f^{(i)}(\bar{x},\bar{v},t),\quad i=1,2, \tag{12}$$

$$\hat{\phi}_{\bar{k}}(t) = \int_0^{2\pi} d\bar{x}\, e^{-i\bar{k}\cdot\bar{x}}\phi(\bar{x},t). \tag{13}$$

Here $\bar{k}$, and ξ corresponds to the conjugate variables of $\bar{x}$, and $\bar{v}$, respectively, being $\bar{k}$ a discrete variable, while ξ is continuous, and $*$ denotes the convolution operator for $\bar{k}$. Note that passing to the Fourier space becomes crucial to be

able to reduce the system of equations into a single one. Moreover, this is mandatory for the purpose of finding a probabilistic representation for the solution of Eqs. (10), applying directly the strategy described previously for the semilinear transport equation. As mentioned in the previous section, the first step towards the probabilistic representation requires rewriting the system of equations (11) in integral form, and is given by

$$F_{\bar{k}}^{(i)}(\bar{\xi},t) = F_{\bar{k}}^{(i)}(\bar{\xi}+t\bar{k},0) + \rho_i \int_0^t ds \sum_{\substack{\bar{k}'=-\infty \\ \bar{k}'\neq 0}}^{\infty} \frac{\bar{k}'\cdot(\bar{\xi}+s\bar{k})}{|\bar{k}'|^2}$$
$$\times[F_{\bar{k}'}^{(1)}(0,t-s) - F_{\bar{k}'}^{(2)}(0,t-s)]F_{\bar{k}-\bar{k}'}^{(i)}(\bar{\xi}+s\bar{k},t-s), \tag{14}$$

where $\rho_1 = 1$, and $\rho_2 = -1/m_2$. Both, the static and dynamic probabilistic representation can be derived similarly to the case of the semilinear transport equation. Here we describe the dynamic representation, since the static one is straightforward. Eq. (14) can be written probabilistically as follows,

$$F_{\bar{k}}^{(i)}(\bar{\xi},t) = E\left[\tilde{F}_{\bar{k}}^{(i)}(\bar{\xi}+t\bar{k},0)\delta(\zeta)\right] \tag{15}$$
$$+E\left[\eta(t)\, g^{(i)}(\bar{k},\bar{k}',\bar{\xi},S)\, F_{\bar{k}'}^{(1)}(0,t-S)\, F_{\bar{k}-\bar{k}'}^{(i)}(\bar{\xi}+S\bar{k},t-S)\delta(\rho-1)\delta(\zeta-1)\right]$$
$$+E\left[\eta(t) g^{(i)}(\bar{k},\bar{k}',\bar{\xi},S)\, F_{\bar{k}'}^{(2)}(0,t-S)\, F_{\bar{k}-\bar{k}'}^{(i)}(\bar{\xi}+S\bar{k},t-S)\delta(\rho-2)\delta(\zeta-1)\right],$$

where

$$g^{(1)}(\bar{k},\bar{k}',\bar{\xi},S) = 2\rho_1 \frac{\bar{k}'\cdot(\bar{\xi}+S\bar{k})}{(1-q)p(\bar{k})|\bar{k}'|^2},$$
$$g^{(2)}(\bar{k},\bar{k}',\bar{\xi},S) = -2\rho_1 \frac{\bar{k}'\cdot(\bar{\xi}+S\bar{k})}{(1-q)p(\bar{k})|\bar{k}'|^2}, \tag{16}$$

and $\tilde{F}_{\bar{k}}^{(i)} = F_{\bar{k}}^{(i)}/q$. Here four random variables have been introduced, those are: ρ is a two-point, $\rho = 1, 2$, discrete random variable equally distributed with probability $1/2$; S a continuous random variable uniformly distributed between 0 and t; $\bar{k}'$ is a discrete random variable with density probability $p(\bar{k}')$, and finally ζ, which takes the values 0, and 1, with probability $P(0) = q$, $P(1) = 1-q$, respectively. E denotes the expected value taken with respect to the density probabilities corresponding to all those four random variables. Note that the value chosen for q has a direct consequence on the numerical error done in computing probabilistically the different integrals in the series expansion, since this affects how the different random trees are randomly distributed among the different coefficients of the expansion. Despite the apparently freedom to choose its value within a certain range of admissible values [3], it does not provide the necessary flexibility for controlling the

numerical error made in computing independently each coefficient. This fact is indeed the major motivation for adopting the more flexible *hybrid* approach, as it was explained in the previous section.

In [24], it has been already proposed a probabilistic representation of the solution of the system equations (10). However, such a representation is rather stringent for practical purposes, since it requires to fulfill strong constraints in terms of the allowed initial data and time. Moreover, the density probability $p(\bar{k}')$ should be carefully chosen, hindering the task of finding a valid density for any dimension of the problem. This is related to the problem of finding admissible majorizing kernels, see [9]. Indeed, it can be readily proved that the majorizing kernel chosen in [24] is no longer valid in one dimensional problems. However, this does not mean that no solution can be found for the system of equations (10), but rather that the numerical method based on such a probabilistic representation gives rise to a divergent series, which requires further numerical treatment. In fact, in this paper we show that relaxing the requirements concerning the initial condition and time, the solution is still smooth, being now needed to resort to some sort of approximation of the divergent series, such as the Padé approximant. Since the density probability $p(\bar{k}')$ can now be chosen freely among a suitable class of functions, this can be used to reduce the statistical error done in computing numerically the solution. Typically, this corresponds to the well known variance reduction techniques often used in Monte Carlo simulations.

Note that Eq.(15) is indeed a probabilistic representation of the Vlasov-Poisson system of equations (in the sense defined previously for the transport equations), and therefore, it can be used for computing the solution at a single point $(\bar{k}, \bar{\xi}, t)$, without the need of a computational mesh. This can be done generating suitable random trees governed by the densities probabilities mentioned above, and taking the expected value of a corresponding multiplicative functional. For a numerical purpose, the associated numerical algorithm turns out to be specially costly for computing the solution in the whole computational domain, since a large sample size is typically required to avoid a large statistical error. However, an alternative does exist, and this will be addressed in Sec. 4.

3.1 Validation of the representation for the linear theory

Here, in order to validate analytically the probabilistic representation derived above, the classical linear Landau damping will be investigated. This will be done linearizing conveniently the system of equations (10) around the equilibrium solution. Let look for a density function according to the Ansatz,

$$f^{(i)}(\bar{x},\bar{v},t) = f_0^{(i)}(\bar{v}) + \varepsilon\, f_1^{(i)}(\bar{x},\bar{v},t) + O(\varepsilon^2),$$
$$\phi(\bar{x},t) = \phi_0 + \varepsilon\,\phi_1(\bar{x},t) + O(\varepsilon^2), \tag{17}$$

where $\varepsilon \ll 1$. Note that ϕ_0 is intentionally set to be constant to satisfy the constraint mentioned above concerning the space average over a period of the electric field. Inserting (17) into (10), the following system of equations are obtained to order ε:

$$\begin{aligned}
\frac{\partial f_1^{(1)}}{\partial t} + \bar{v}\cdot\nabla_{\bar{x}} f_1^{(1)} - \nabla\phi_1\cdot\nabla_{\bar{v}} f_0^{(1)} &= 0,\\
\frac{\partial f_1^{(2)}}{\partial t} + \bar{v}\cdot\nabla_{\bar{x}} f_1^{(2)} + \frac{1}{m_2}\nabla\phi_1\cdot\nabla_{\bar{v}} f_0^{(2)} &= 0,\\
\triangle_{\bar{x}}\phi_1 = -\left[\int f_1^{(1)} d\bar{v} - \int f_1^{(2)} d\bar{v}\right],&
\end{aligned} \tag{18}$$

with initial data $f_1^{(i)}(\bar{x},\bar{v},0) = A\, g_i(\bar{v})\,\cos(k_1 x), i = 1, 2$, 2π-periodic boundary conditions in $\bar{x}$, and $g_i(v)$ decaying to zero as $|\bar{v}| \to \infty$ with sufficiently high rate.

Applying identical mathematical treatment as done previously for the fully Vlasov-Poisson system of equations, the following integral equation for the Fourier transform $F_{\bar{k}}^{(i)}(\bar{\xi},t)$ of $f_1^{(i)}$ is obtained,

$$\begin{aligned}
F_{\bar{k}}^{(i)}(\bar{\xi},t) &= F_{\bar{k}}^{(i)}(\bar{\xi} - t\bar{k},0)\\
+\rho_i \int_0^t ds\, \frac{\bar{k}\cdot(\bar{\xi} - s\bar{k})}{|\bar{k}|^2}&[F_{\bar{k}}^{(1)}(0,t-s) - F_{\bar{k}}^{(2)}(0,t-s)]\hat{g}_i(\bar{\xi}_2 - s\bar{k}).
\end{aligned} \tag{19}$$

Here $\hat{g}_i$ is the corresponding Fourier transform of $g_i(\bar{v})$. In the following let consider $\bar{\xi} = 0$, and for simplicity we assume $\bar{k} = (k_1, 0, 0)$. The Fourier transform of the x-component of the electric field is given by, $\hat{E}_x(k_1,t) = -\frac{i}{k_1}[F_{k_1}^{(1)}(0,t) - F_{k_1}^{(2)}(0,t)]$. A recursive solution can be obtained for $\hat{E}_x(k_1,t)$ using Eq.(19), and yields,

$$\hat{E}_x(k_1,t) = -i\frac{1}{k_1}\Phi(k_1,t) + \sum_{j=1}^{\infty}(-1)^j\,\eta_j(\bar{k},t). \tag{20}$$

Here η_j is given by

$$\begin{aligned}
\eta_j(k_1,t) = \frac{(-i)^j}{k_1^j}\int_0^t ds_1\, s_1 \int_0^{t-s_1} ds_2\, s_2 \cdots \int_0^{t-\sum_{l=1}^{j} s_l} ds_j\, s_j&\\
\times\Phi(k_1, t - \sum_{l=1}^{j} s_l)\hat{h}(-s_1\, k_1)\hat{h}(-s_2\bar{k})\cdots\hat{h}(-s_j\bar{k}),&
\end{aligned} \tag{21}$$

where $\Phi(k_1,t) = F^{(1)}_{k_1}(-t\,k_1,0) - F^{(2)}_{k_1}(-t\,k_1,0)$, and $\hat{h}(-t\,k_1) = [\hat{g}_1(-t\,k_1) + \frac{1}{m_2}\hat{g}_2(-t\,k_1)]$. Note that η_j can be obtained recursively from η_{j-1} as follows

$$\eta_j(k_1,t) = \int_0^t ds\, s\, \eta_{j-1}(k_1,t-s)\hat{h}(-s\,k_1) \tag{22}$$

Multiplying Eq.(20) by $\hat{h}(-s\,k_1)$, and integrating with respect to s, it holds that

$$\int_0^t ds\, s\, \hat{E}_x(k_1,t-s)\hat{h}(-s\,k_1) = -\hat{E}_x(k_1,t) - i\frac{1}{k_1}\Phi(k_1,t), \tag{23}$$

The integral equation above turns out to be a Volterra equation of the second kind, whose solution can be obtained readily by means of the Laplace transform. In fact, Laplace transforming Eq. (23), we obtain

$$\tilde{\hat{E}}_x(k_1,p) = -i\frac{1}{k_1}\frac{\tilde{\Phi}(k_1,p)}{D(k_1,p)}, \tag{24}$$

where $D(k_1,p)$ is given by

$$D(k_1,p) = 1 - \frac{1}{k_1}\frac{d\tilde{\hat{h}}}{dp} \tag{25}$$

The solution can be obtained taking the inverse Laplace transform, and is given formally by

$$\hat{E}_x(k_1,t) = \frac{1}{2\pi i}\int_{\sigma-i\infty}^{\sigma+i\infty} \tilde{\hat{E}}_x(k_1,p)e^{p\,t}\,dp, \tag{26}$$

where integration is taken along a line parallel to the imaginary p-axis and to the right of all singularities of the integral. Then, it holds

$$\hat{E}_x(k_1,t) = \sum_j R_j e^{p_j\,t}, \tag{27}$$

where p_j are simple poles where the function $D(k_1,p)$ vanishes, and R_j is the residue of $\tilde{\hat{E}}_x$ at p_j. Since the poles are in general complex, Eq. (27) can be rewritten as

$$\hat{E}_x(k_1,t) = \sum_j R_j e^{\gamma_j\,t + i\omega_j t}, \tag{28}$$

being $p_j = \gamma_j + i\omega_j$. Note that if $\gamma_j < 0$, all terms are exponentially damped, and the electric field behaves as a damped oscillator, where γ_j, and ω_j denote the damping rate, and the frequency of the oscillation, respectively. In the following some specific examples are given:

(a) *Landau damping with homogeneous background*. Let consider only electron motion, assuming the ions form an static, homogeneous, neutralizing background, that is $m_2 = \infty$, and the initial condition $f_1^{(2)}$ does not depend on $\bar{x}$. Suppose that the initial condition for the electrons is the maxwellian distribution, $g_1(\bar{v}) = (\alpha/\pi)^{d/2} \exp(-\alpha \bar{v}^2)$, that is

$$f_1^{(1)}(\bar{x}, \bar{v}, 0) = A\, g_1(\bar{v})\, \cos(k_1 x) \tag{29}$$

Then, the function $D(k_1, p)$ takes the form

$$D(k_1, p) = 1 - \frac{\alpha}{k_1^2} Z'(\zeta), \tag{30}$$

where $\zeta = i\sqrt{\alpha} p/k_1$, and $Z(\zeta)$ the plasma dispersion function. Note that this reproduces exactly the classical result for the dispersion relation using the well-known linear Landau theory [18].

(b) *Landau damping with heterogeneous background*. Now we take into account also the ion dynamics. For simplicity, let assume that both initial conditions for ions, and electrons, are maxwellian distribution for velocities, being given by

$$f_1^{(i)}(\bar{x}, \bar{v}, 0) = A \left(\frac{\alpha_i}{\pi}\right)^{d/2} e^{-\alpha_1 \bar{v}^2} \cos(k_1 x), \quad i = 1, 2 \tag{31}$$

The function $D(k_1, p)$ reduces to

$$D(k_1, p) = 1 - \frac{\alpha_1}{k_1^2} Z'(\zeta_1) - \frac{1}{m_2} \frac{\alpha_2}{k_1^2} Z'(\zeta_2), \tag{32}$$

where $\zeta_i = i\sqrt{\alpha_i} p/k_1\, i = 1, 2$. Again, this coincides exactly with the results obtained using the classical linear Landau theory [18].

4 Evaluating numerically the probabilistic representation of Vlasov-Poisson

Here we explain in detail how the probabilistic representation (15) can be used in practice to compute numerically the solution of the Fourier-transformed Vlasov-Poisson system, and which are the numerical errors done. For simplicity, in the following only the 1-dimensional case and one specie of charged particles (electrons) moving in a neutralizing homogeneous background charge (ions), has been considered. Eq.(14) can then further simplified by setting $i = 1$ and $P(\rho = 1) = 1$. Note that similarly to the probabilistic representation obtained for the transport equation in (7), the probabilistic representation for

the Vlasov-Poisson system in (15) is not given in a closed form, being the right-hand side a function of the solution itself. Following the same strategy as explained previously for the case of the transport equations, such an implicit equation can be solved recursively resorting to the aforementioned hybrid probabilistic representation approach, which in practice requires generating prescribed random trees governed by two random variables S, and k'.

4.1 Numerical errors of the probabilistic part

When computing numerically the probabilistic representation of the solution, adopting the hybrid method earlier explained, three independent sources of numerical errors appear, which should be controlled to assure the convergence of the numerical method, and these are

- *Statistical error.* The coefficients of the series expansion are essentially multidimensional definite integrals. In general these integrals cannot be computed analytically and we have to resort to numerical techniques, thus introducing a source of errors in the numerical method. Among the possible numerical methods that can be used to compute numerically such integrals, a Monte Carlo method has been adopted, since the dimensionality of the definite integral for the higher terms in the expansion may be so large that it makes unaffordable the use of classical quadrature integration techniques. The integrals can be seen theoretically as expected values of suitable random variables, and for numerical purposes conveniently approximated, considering a finite sample size N, by the arithmetic mean. It is well known that this accounts for an error, which is of statistical nature, and of order of $O(\sigma/N^{-1/2})$ when N goes to infinity, being σ the variance of the given random variable. A quasi-Monte Carlo method can be used instead, since it offers a better convergence rate, speeding up notably the simulations [40]. The error is typically deterministic and of order of $O(log^{d^*-1}N/N)$, where d^* is the effective dimension [11–13,34]. This choice is an alternative to that of the more common Monte Carlo method, based as a rule on sequences of pseudorandom numbers, being now based rather on quasi-random (low discrepancy) sequences of numbers. It is worth to observe that for both methods increasing the dimensionality of the problem increases the statistical error. For the Monte Carlo method, this is because the variance typically depends on the dimension, while for quasi-Monte Carlo the dependence is made explicit through the effective dimension.

  In Fig. 2, it is shown the maximum absolute numerical error done, when computing numerically the integrals corresponding to one of the coefficients pertaining to the series expansion using the quasi-Monte Carlo

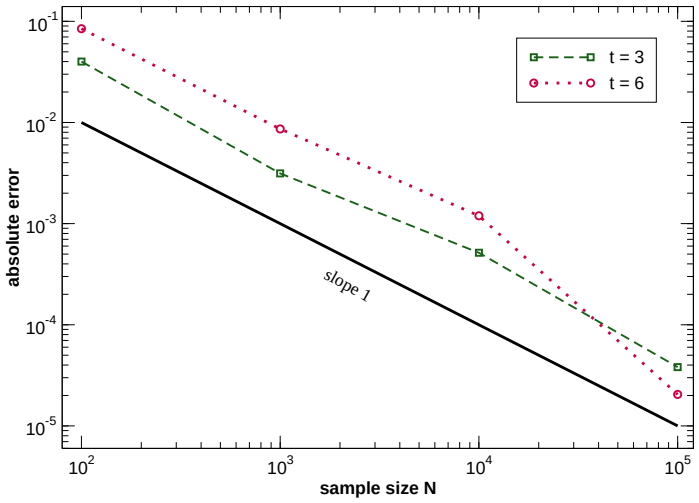
and Monte Carlo method. This is done at a single point (k, ξ), two different values of time, and for the initial condition in (35). The error has been computed comparing with the solution obtained using an extremely accurate numerical quadrature based on sparse grid techniques.

Note that the numerical curves (dotted and dashed) appear to match well the curves serving as reference, which are of slope 1 for the case of quasi-Monte Carlo, and 1/2 for the case of Monte Carlo, in accordance with the theory. Moreover, the error increases when the dimension of the definite integrals increases, as expected for the theoretical considerations above.

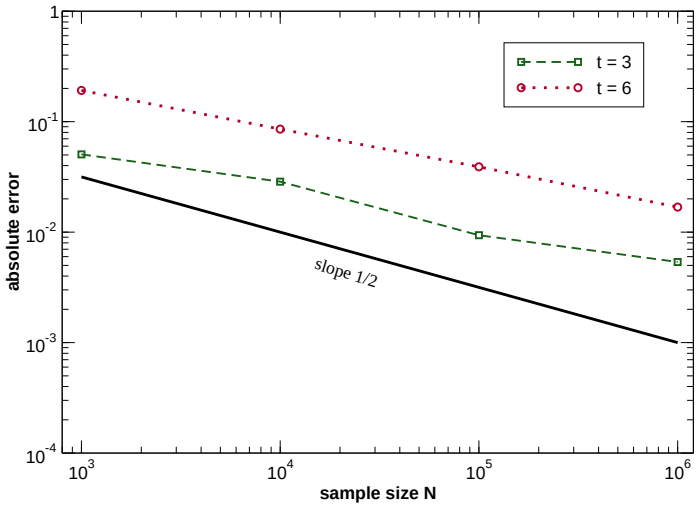
Finally, it is worth to point out that the statistical error done when evaluating the definite integrals can be reduced by choosing a particular density function $p(k')$, which minimizes the variance of the corresponding random variable. We cannot estimate a priori which function is the optimal one, and for simplicity in all calculations done so far a uniform function has been used. Clearly, it may be possible to develop an adaptive algorithm capable to modify $p(k')$ in run-time, but this goal is out of the scope of this paper and left as a possible future work.

- *Truncation error*. Clearly, the series expansion obtained when solving recursively (15) has to be truncated for numerical evaluation. Such a truncation induces a numerical error, which typically increases with time. Therefore, to obtain an accurate solution for arbitrary long times it is needed to consider more coefficients in the series expansion. However, since such terms are definite integrals of increasingly dimensionality, computing them accurately requires to increase the sample size to keep constant the statistical error. As a consequence the computational cost rises dramatically. Furthermore, as mentioned previously, the number of possible integrals determining each of the terms of the series expansion grows rapidly with the order of the given term, therefore calculating higher terms is extremely undesirable in view of the high computational cost.

  This is actually the practical reason why we cannot expect to compute numerically the solution probabilistically for arbitrarily long times. Consequently it becomes necessary to apply some kind of restarting procedure from time to time, reusing the solution obtained at a given time as the new numerical initial condition from which restarting the numerical method. Being now the new numerical initial condition given numerically, it becomes necessary to interpolate for computing the solution for subsequent times, and thus introducing a new source of error, which should be conveniently controlled. In the following, we will refer to this procedure as *time sliding window*. The particular time interval chosen obviously will depend on the desired accuracy, and the number of coefficients in the series expansion wanted to be computed. In fact, when the series expansion turns out to be convergent, it can be seen from Eq. (6) that the truncation error made truncating the series up to n splitting events is bounded by $O(t^{n-1})$. Therefore, the order of the method in time is proportional



(a) Quasi-Monte Carlo.



(b) Monte Carlo.

Fig. 2. Maximum absolute error as function of the sample size, when computing numerically the integrals corresponding to one of the coefficients of the series expansion. This has been done at a single point ($k = 1, \xi = 7.5$), and for $t = 3$ and $t = 6$. The solid lines serving as reference are of slope 1, and 1/2, for quasi-Monte Carlo in (a), and Monte Carlo in (b), respectively.

to the maximum number of splitting events considered in the expansion, being the size of the window, T_w, somehow equivalent to the typical discretization parameter Δt found in classical numerical schemes, such as finite difference or finite elements, for instance.

- *Padé approximant.* As mentioned earlier, an important issue comes from the fact that the series expansion turns out to be in general divergent. In such cases, a Padé-Approximant has been considered to asymptotically approximate the given divergence series, which gives rise to a new source of errors. Since finding theoretical estimates of such an error for any given problem may be a formidable task, our goal here consists merely to gain some insight of such an error, illustrating how well Padé approximation actually works by analyzing a few relevant test problems. In Fig. 3(a) we show in practice the convergence behavior of the Padé-Approximant by considering from 2 to 6 coefficients in the approximation. These coefficients were computed by the quasi-Monte Carlo method using Sobol sequences of numbers [11–13] with a sample size $N = 10^3$. On the other hand, in Fig. 3(b) we compute the same solution this time by Monte Carlo with 10^6 as sample size, that is three orders of magnitude larger. Only four coefficients were computed so far, since for higher coefficients the statistical error becomes already so large that makes completely useless the numerical solution. Notice the jiggling behavior of the solution, as expected due to the random nature of the Monte Carlo algorithms, in contrast with the typical smooth behavior of the deterministic solution of the quasi-Monte Carlo method.

  One could expect that the statistical error made in evaluating the coefficients by Monte Carlo might spoil the convergence of the Padé approximant. However, when the statistical error is sufficiently small, the coefficients of the Padé approximant can be obtained within a reasonable accuracy, being therefore the Padé approximant rather robust at least for the examples here considered, and converging reasonably fast to the solution of the problem. In Fig. 4 the error done, when computing the solution at a given point and for two different values of time, as function of the number of coefficients is shown. Note the fast convergence of the approximation, being the error smaller for smaller times, as expected.

  Recall that in general the convergence of the Padé approximant can be affected by artificial poles present in the denominator of the approximant, but not being own by the function to be approximated, see e.g. [7,8]. In fact, it can be seen in Fig. 3(a), and (b), the presence of an artificial pole that progressively disappears when considering more coefficients into the expansion. Therefore, to assess properly the validity of our findings, it becomes essential to compute the Padé approximant for different number of coefficients.

  Concerning the apparent robustness of the Padé approximant against the statistical error affecting the coefficients of the series expansion, a main reason could be that the solution of the test examples seems to

be apparently locally Lipschitz. Thus, the error made in computing the coefficients of the Padé approximant should be bounded. In fact, in [43] it has been proved the following related theorem

$$\| P_f - P_{f'} \| \leq K \| c - c' \|, \tag{33}$$

provided that $\| c - c' \| \leq d$. Here P_f, and $P_{f'}$ are the Padé approximants of order (m, n) in $[a, b]$ of a given power series f and f' with coefficients c_j, and c'_j respectively, being $\| c \| = max_{i \leq i \leq n+m} |c_i|$, f locally Lipschitz, and K and d constants depending only on c_i and $[a, b]$.

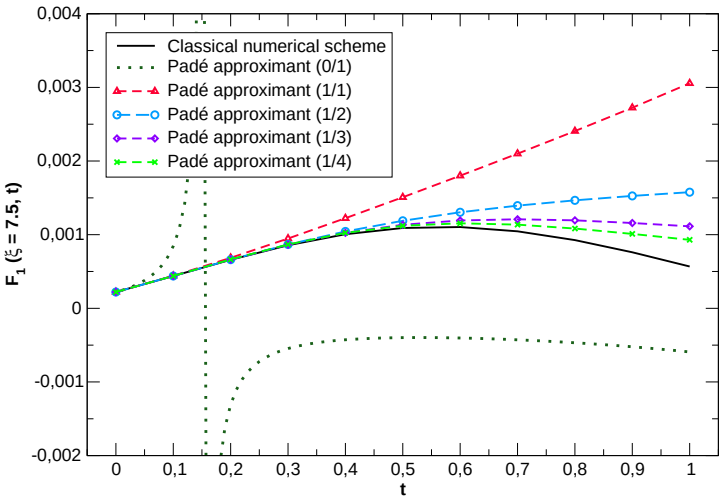
In closing, it is worth to observe that all errors described above may be alleviated in any case by increasing conveniently the sample size N, and considering more coefficients in the expansion in order to compute the Padé approximant.

5 The PDD algorithm for Vlasov-Poisson

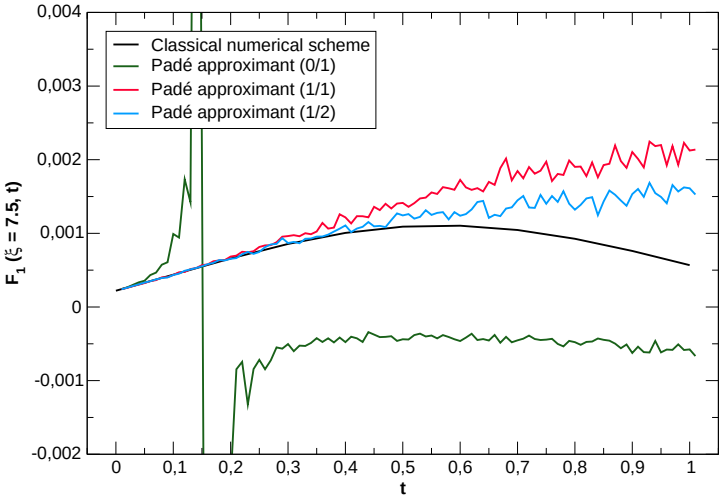
As earlier mentioned, the probabilistic method will be combined with a classical method by means of the PDD method to solve the Vlasov system for the full domain, rather than applying the probabilistic method alone point by point within a given computational mesh arbitrarily chosen. This is in view of the high computational cost of the probabilistic method. In the first subsection the PDD algorithm is described in detail, following a brief overview of the numerical errors associated. The next subsection analyzes the computational cost of the algorithm, and finally, the two last subsections are devoted to test both the accuracy and performance of the PDD algorithm by considering traditional problems extensively studied in the literature of numerical methods for solving the Vlasov-Poisson system.

5.1 Description

In this section we describe in detail the different parts compounding the PDD algorithm, and how the algorithm has been implemented in practice, resorting to Fig. 5 for the purpose of illustration. Recall that the domain for the particular problems covered by this paper, is unbounded in the Fourier space for both variables k, and ξ. Hence for numerical purposes it should be truncated conveniently as $\Omega = [0, \xi_{max}] \times [-k_{max}, k_{max}]$, with ξ_{max}, and k_{max} chosen large enough to impose properly zero Dirichlet boundary conditions at the extreme of the domain. Note that ξ is restricted to taken positive values, because being $F(k, \xi, t)$ the Fourier transform of the real function $f(x, v, t)$, it holds that $F(k, -\xi, t) = F(k, \xi, t)^*$, therefore only computing the solution at the positive semi-axis is needed to reconstruct the solution for the full domain.



(a) Quasi-Monte Carlo.



(b) Monte Carlo.

Fig. 3. Time evolution of the solution $F_1(\xi = 7.5, t)$ numerically obtained by using several Padé-Approximants of different order. Here (m/n) indicates a rational approximant of order m and n for the numerator and denominator, respectively. The corresponding terms in the series expansion are computed by quasi-Monte Carlo integration, using $N = 10^3$ as sample size in (a) and 10^6 in (b).

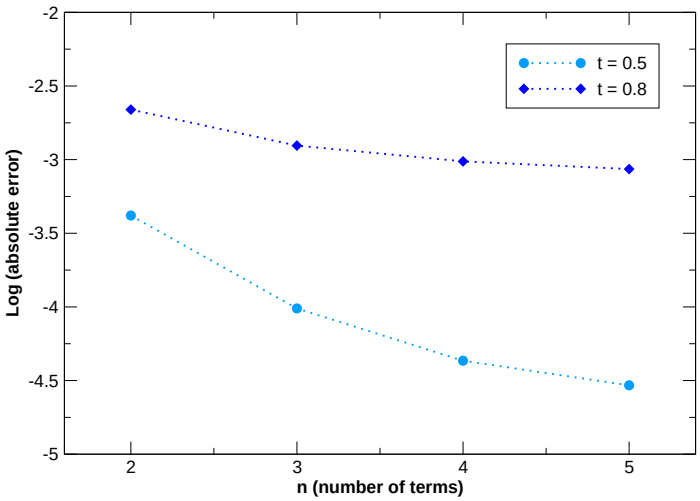


Fig. 4. Absolute error in log-scale done when computing numerically $F_1(\xi = 7.5, t)$ for different number of terms of the series expansion. The terms were computed by quasi-Monte Carlo for two different values of time, $t = 0.5$ and $t = 0.8$. Here N is kept fixed to 10^3.

- *Probabilistic part.* Following the same strategy as in [3–5], a domain decomposition is effectively accomplished by first, computing a numerical solution using the probabilistic representation given in (15) on suitable interfaces, and then splitting the computational domain into p independent subdomains, being p the number of processors involved in the computation. The ultimate idea is to be able to fully decouple the original problem into p independent subproblems, which they can be eventually solved resorting to classical numerical methods. The corresponding computational task can be assigned then to independent processors, which process the tasks simultaneously without any intercommunication.

  Many different ways of partitioning the computational domain can be considered so far, but for this our particular problem it turns out to be more convenient to split it selecting the interfaces as the lines of constant ξ. In fact, due to the convolution term in (11) the resulting discretized equations when solving locally the subproblems appears to be globally coupled in the k variable, while locally coupled in the ξ variable. Let define the p subdomains $\Omega_i, i = 0, ..., p-1$, as those given by $\Omega_i = [\xi_i, \xi_{i+1}] \times [-k_{max}, k_{max}]$, being $\xi = \xi_i$ the interfaces chosen, as shown in Fig. 5(a). Then, the solution is computed at few points, given by (ξ_i, k, t_j), $j = 0, .., w$, with $t_w = T$, for all values of k, and different values of time using the probabilistic representation. Recall that k is a discrete variable ranging between $-k_{max}$ and k_{max}.
- *Interpolation.* The points computed probabilistically will be used as nodal

points for interpolation, thus obtaining a continuous approximation of the solution along the given interfaces, and consequently the sought boundary data needed to fully decouple the problem into p independent subproblems defined on Ω_i. In this paper, the B-Spline basis functions are chosen to interpolate by building cubic B-Spline interpolants. This is done due to the advantages given by piecewise over polynomial interpolation (such as avoiding the Runge's phenomenon) and their particular flexibility to algorithmically deal with splines. Moreover, since it is an interpolation method of higher order, we can assure that the error done when interpolating will be well below the errors coming from the probabilistic part of the algorithm.

- *Local solver*. The third and final part consists of computing the solution inside each subdomain Ω_i, this task being assigned to different processors. This can be accomplished resorting to local solvers, which may consist in classical numerical schemes, such as finite differences for simple geometries or finite elements methods for more complex configurations. An implicit finite difference numerical scheme has been adopted so far to solve locally the system equations (11), discretizing conveniently the variable ξ, and the time t on a grid with step sizes $\Delta\xi$, and Δt respectively. An implicit method is chosen to avoid leading with the well-known CFL condition, which in practice restricts the range of allowed time step to ensure stability of the numerical scheme. The price to pay is that a linear algebra problem should be solved for each time step. Among the many available numerical packages, LAPACK has been chosen for this purpose. This is done in view of the well-established robustness and accuracy of their subroutines for dealing specifically with the banded matrices characterizing our problem. For the finite difference scheme, an upwind scheme has been adopted for the numerical simulations. This has been done for simplicity, and because as mentioned previously, the main goal of this paper was merely to show how a probabilistic representation of the solution can be used to complement any classical schemes, speeding up accordingly the parallel simulations. Obviously higher order schemes such as WENO e.g., can be considered as well, and this is left for a future work.

Apart from the steps already described, a new component should be included into the algorithm for dealing with long-time simulations. In fact, as mentioned in the previous section, the solution cannot be computed in practice for an arbitrarily large time due to the uncontrolled growth of the numerical errors, and therefore a time sliding window procedure should be adopted instead. Thus, the strategy consists of considering a fixed T_w smaller than the given final time T_f. The restarting procedure should be repeated further until reaching such a prescribed final time T_f. This is shown schematically in Fig. 5(a). Implementing such a restarting procedure is computationally costly, because it requires a further interpolation now in both variables k, and ξ since the initial condition is now given numerically, which in practice results in a unde-

sirable communication overhead. Therefore, to improve at best the efficiency of the algorithm reducing such communication overhead, it is important that the restarting time T_w be as large as possible.

However, it turns out that this restarting time should not be chosen rather large. This is to avoid the need of interpolating globally using the full domain. In fact, since every processor only holds the local data belonging to their assigned subdomain Ω_i, this eventually may induce a global communication operation among the processors involved in the computation, degrading notably the overall performance of the algorithm. The explanation rests in the following observation: Let (ξ_i, k, T_w) the point chosen to compute the solution probabilistically belonging to the subdomain Ω_i. Note that the probabilistic representation in Eq. (15) requires generating random trajectories starting at such a point, which may explore the whole domain during the time T_w, and thus they can reach parts of the domain not belonging to the initial subdomain where the trajectories started. Nevertheless, since the unbounded domain has to be truncated to be Ω for computational purposes, in practice such random trajectories will be never go out from the region $R_i = [\xi_i - k_{max}T_w, \xi_i + k_{max}T_w] \cup [0, k_{max}T_w]$, see Fig. 5(b) for illustration. In fact, this can seen from Eq. (15), noting that when solving recursively such equations, the solution modifies their arguments according to $(0, T_w - S)$ and $(\xi_i + Sk, T_w - S)$, respectively. This propagates in such a way that in the worst case, due to the random nature of k', and S, the first term of the right-hand side corresponding to the initial condition, or the numerical approximation to be interpolated, is never been evaluated for values of ξ outside of the region R_i.

Therefore, the intercommunication overhead can be diminished simply choosing conveniently the time T_w such that the region R_i involves the smallest possible number of subdomains. In the most favorable case, this can be reduced to only two, Ω_{i-1}, and Ω_0. This will force the processor assigned to the domain Ω_i to communicate data with the processors assigned to the subdomains Ω_{i-1}, and Ω_0, respectively. Concerning the subdomain Ω_0, and the local data that contains, note that they are needed for every processor involved, and thus they should be globally broadcast to all processors. The remaining non-local data needed to be transferred between processors assigned to subdomains Ω_i, and Ω_{i-1}, can be done through a nearest-neighbor communication operation, and allowing in consequence to implement a highly efficient parallel algorithm. Finally, note that this procedure should be repeated every certain time interval given by T_w until the final time T_f is reached, as illustrated in Fig. 5(c).

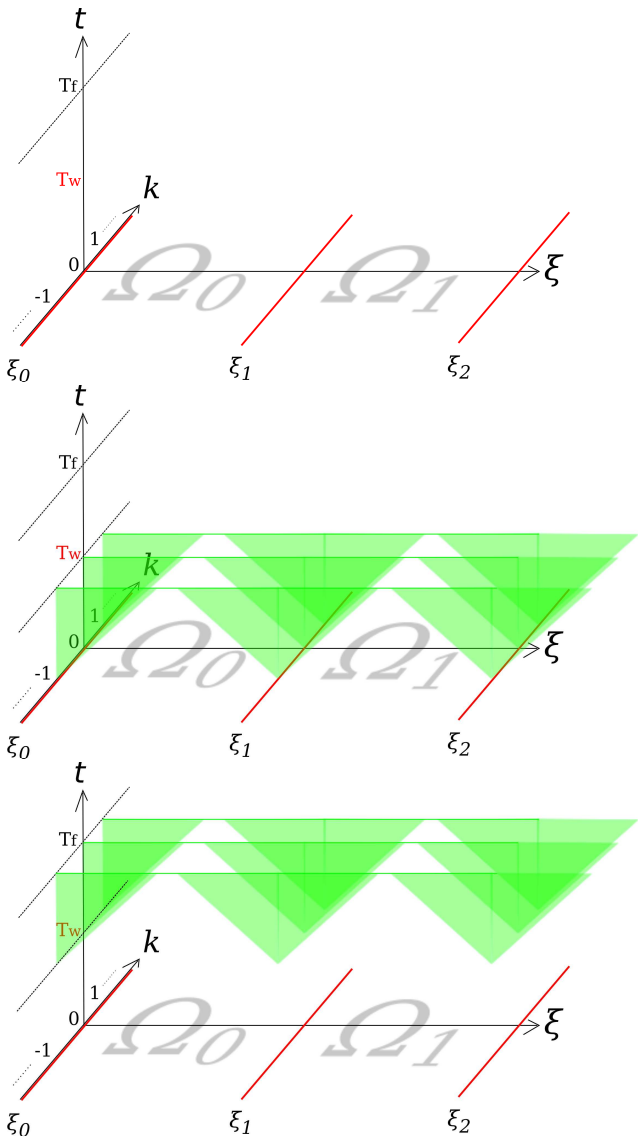


Fig. 5. Graphical description of the different steps of the algorithm organized in time: (a) The different subdomains, $\Omega_0, \Omega_1, ...$, dividing the domain are bounded by the pair of values $(\xi_0, \xi_1), (\xi_1, \xi_2), ...$, along the ξ-dimension; (b) A numerical solution using the probabilistic representation is computed on the interfaces, being the random trajectories confined inside the green region, which starts at time $t = 0$ and ends at time T_w; (c) The algorithm restarts many times as necessary from time T_w until reaching the final time T_f.

5.2 Global error of the method

Several numerical tests, extracted from the classical repertoire of initial data found in literature for the Vlasov-Poisson system, have been carried out. In this section we show the typical global error found in practice when solving such problems. For this purpose, since in general analytical solutions are not

available for comparison, the numerical solution obtained with an upwind finite difference scheme using a very fine mesh, was considered as the solution serving as reference.

Let consider the typical initial condition used for analyzing the strong Landau damping in (35). The reference solution was computed by choosing the following discretization parameters: $\Delta\xi = 10^{-6}$ and $\Delta t = 1/4\Delta\xi$. In Fig. 6 the absolute numerical error made with our algorithm for different values of the discretization parameter is plotted, showing the expected convergence for smaller values. Due to the oscillatory nature of the solution in time, in these curves for clarity only the maximum values taken are shown.

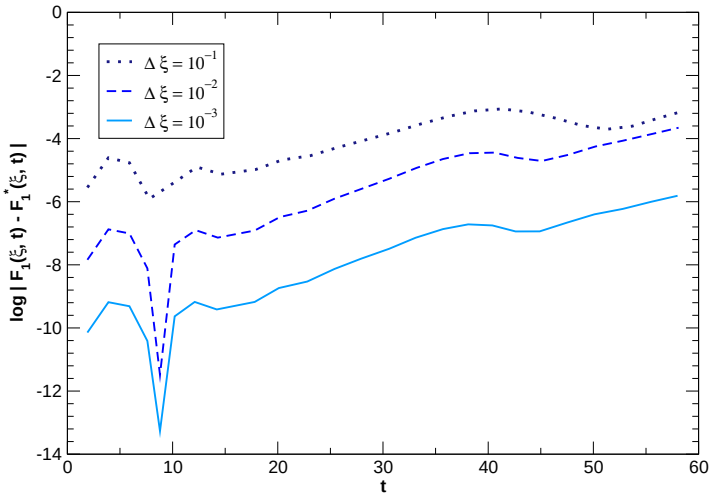


Fig. 6. Time evolution of the absolute numerical error of the PDD method, obtained for several values of the discretization parameters $\Delta\xi$ and Δt.

5.3 Computational cost

Once the algorithm was fully described, the computational cost could now be estimated. For simplification, and without loss of generality, here we focus merely on estimating the computational cost spent by the processor holding initially the subdomain Ω_1. Let first summarize the different discretization parameters involved for solving a given problem by the PDD algorithm. Recall that these concerns both, those affecting the probabilistic part of the algorithm, and the local solver. For the former, the numerical method based on Monte Carlo depends on the sample size M as discretization parameter, while for the latter, the implicit upwind numerical scheme specifically chosen for this paper, can be characterized in terms of discretization parameter $\Delta\xi$,

and Δt. Other relevant parameters are k_{max}, T_f, which determine the size of the computational domain in space and time, and $T_w, \Delta T_w$ the time sliding window parameters. For illustration, in Fig. 7 a graphical description of the parameters is shown. Based on the parameters above, and in order to derive the computational cost, the following auxiliary parameters are needed,

- $N_\xi = \xi_{max}/\Delta_\xi$ the number of points along the $\xi-$dimension;
- $N_k = 2k_{max} + 1$ the number of points along the $k-$dimension;
- $N_{T_w} = T_w/\Delta T_w$ the number of points used for interpolation in time on the interfaces ξ_1 and ξ_2 delimiting the subdomain Ω_1. T_w is the size of the time sliding window, while ΔT_w is the step size between the nodal points. Note that for each interface, and different value of k, ranging from $-k_{max}$ to k_{max}, N_k different interpolants are needed.
- $N_w = T_f/T_w$ the number of times the sliding window procedure should be applied.
- $N_t = T_w/\Delta_t$ the number of points corresponding to the time discretization of the finite difference scheme for every time sliding window.
- $N_{R_0} = k_{max}T_w/\Delta\xi$ the number of points chosen for interpolation on the region R_0. Recall that this is the common region whose data are needed for any of the processors involved;
- $N_{R_i} = 2N_{R_0} - 1$, $i \neq 0$, the number of points chosen for interpolation on the neighboring regions to Ω_i, which is twice as much the number of points of R_0.

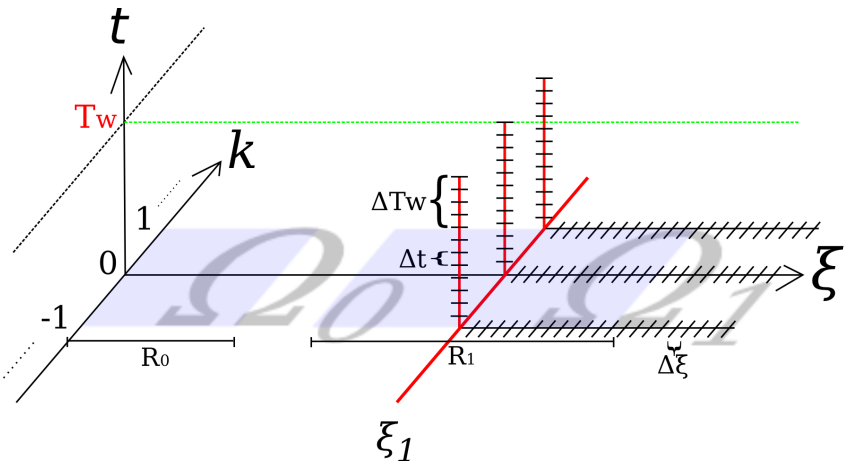


Fig. 7. The different discretized parameters of the algorithm required to estimate the computational cost in Sec. 5.3

In addition to all previous definitions, let introduce as well the following constant values, related with the cost of the communication operations through the network, and the interpolation costs. They are respectively: C_s the start-up cost when sending a message through the network, C_w the cost of sending a single word, and C_B the cost of constructing a B-Spline, C_I the interpolation cost. Finally, the total cost of the algorithm can be estimated as the result of

combining the following subsequent stages:

(1) *Interpolation over regions numerically given*. Constructing the corresponding set of 1-dimensional B-Splines for interpolation, necessary on the regions where the initial data is numerically given since the restarting procedure has been applied. The different phases of this stage, along with their respective computational cost, are:
   · Broadcasting the set of points belonging to the region R_0 to all processors: $(C_s + C_w N_{R_0} N_k) \log p$.
   · Constructing the corresponding set of B-Splines: $C_B N_k N_{R_0}$.
   · Point-to-point communication operation of the region $R1$: $4(C_s + C_w N_{R_1}/2N_k)$.
   · Constructing the corresponding set of B-Splines: $C_B N_k N_{R_1}$.

(2) *Monte Carlo evaluation of the solution along interfaces*: $N_k N_{T_w} MC_I$.

(3) *Time interpolation over the interfaces*. Constructing the set of 1-dimensional B-Splines needed for interpolating along the time-axis over the interfaces delimiting the subdomains. The different phases and cost are given by,
   · Broadcasting the different points located on $\xi = 0$: $(C_s + C_w N_{T_w} N_k) \log p$.
   · Constructing the corresponding set of B-Splines: $C_B N_k N_{T_w}$.
   · Point-to-point communication operation of the interfaces delimiting R_1: $2(C_s + C_w N_{T_w} N_k)$.
   · Constructing the corresponding set of B-Splines: $C_B N_k N_{T_w}$.

(4) *Local solver*. The cost can be estimated as follows:

$$
\begin{aligned}
3N_k N_t C_I + N_t \Bigg[& \underbrace{\frac{4(N_k+1)N_\xi}{p}}_{\text{(i)}} + \underbrace{\frac{N_k N_\xi}{p}\left(C_{mc} + N_k C_{sc}\right)}_{\text{(ii)}} + \underbrace{N_k \frac{N_\xi}{p} {N_k}^2}_{\text{(iii)}} + \underbrace{{N_k}^2 \frac{N_\xi}{p}}_{\text{(iv)}} \Bigg] \\
& = 3N_k N_t C_I + N_t \frac{N_\xi}{p} \left[4(N_k+1) + N_k \left(C_{mc} + N_k C_{sc}\right) + {N_k}^3 + {N_k}^2 \right],
\end{aligned}
$$

where C_{mc} corresponds to the cost of computing the matrix coefficients, being C_{sc} the cost of the convolution term. The first term appearing in the formula refers to the cost of interpolating on both interfaces delimiting Ω_1 and on the interface ξ_0. The second term corresponds to the cost of solving N_t times the ensuing linear algebra problem, and the different phases involved (labeled accordingly) are given by,

(i) the memory space required by LAPACK for introducing the matrix coefficients and to factorize, has to be set up to 0;
(ii) the matrix coefficients are computed, taking into account the convolution term;
(iii) the factorization of the matrix is carried out;
(iv) the solver provides the final solution.

Note that among the different costs described above, the corresponding to the local solver is the only one which depends explicitly on the size of the ξ-dimension measured by N_ξ. Typically N_ξ should be chosen large enough to accommodate accurately the truncated unbounded domain, and thus this stage becomes the most computationally demanding part of the algorithm. Furthermore, it is precisely this part which ensures the good scalability properties of the algorithm, since the cost decreases proportionally to the number of processors p used, reaching however a constant limit whenever the ratio N_ξ/p becomes sufficiently small.

5.4 Numerical test examples

Previously to analyzing the performance of our algorithm when run in parallel, in the following several numerical examples are given. Those were chosen from the classical repertoire of possible initial conditions traditionally used for testing numerical methods developed for Vlasov-Poisson system, and which describe certain phenomena well known in Plasma physics. The ultimate goal is to characterize the accuracy of our algorithm when dealing with realistic situations.

5.4.1 Landau damping

Let consider the following initial condition

$$f(x,v,0) = \left(\frac{\alpha}{\pi}\right)^{d/2} e^{-\alpha v^2} \left[1 + A\cos\left(k_1\, x\right)\right], \tag{34}$$

which in Fourier space reads

$$F_k(\xi,0) = e^{-\frac{\xi^2}{4\alpha}}[A/2\,\delta(|k| - k_1) + \delta(k)] \tag{35}$$

The first numerical test deals with the so-called weak Landau damping, being the perturbation parameter A chosen sufficiently small. Here the damping rate and the oscillation frequency obtained numerically with our algorithm has been compared with the results obtained by the linear theory theoretically derived in Sec. 3.1. In Fig. 8 and 9, the damping rate and the oscillation frequency are shown for different values of α, which is related so far to different values of the plasma temperature β, being $\beta = 1/4\alpha$. Here A has been kept fixed to 0.01. The remarkable agreement between the analytical linear theory and the numerical results allows us to safely analyze more complicated situations such as the strong Landau damping, where the aforementioned filamentation phenomenon is significantly more severe.

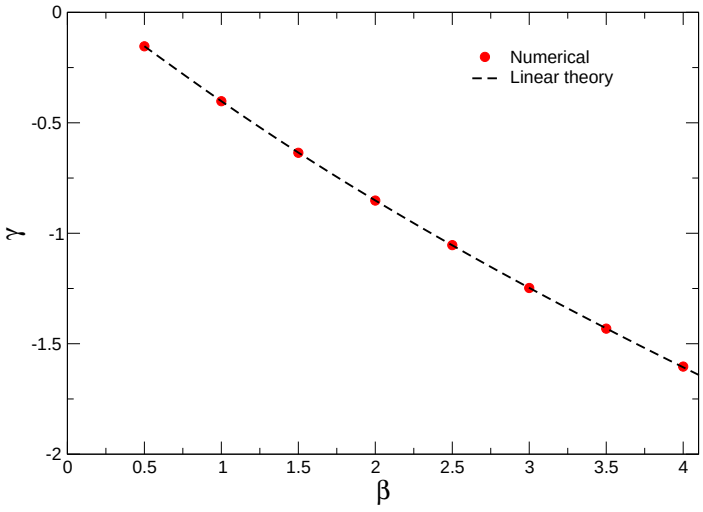


Fig. 8. Weak Landau damping: damping rate for different values of β. Parameters are: $A = 0.01$ and $M = 10^3$.

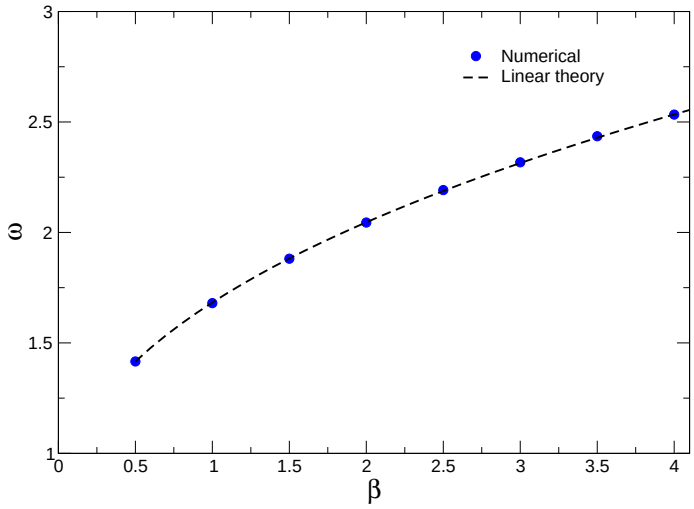


Fig. 9. Weak Landau damping: oscillation frequency for different values of β. Parameters are: $A = 0.01$ and $M = 10^3$.

Let consider the same initial condition, but now choosing larger values of A, that is $A = 0.5$ for exploring the strong Landau damping regime. The numerical solution obtained by the PDD method has been compared with the solution obtained when using an implicit upwind finite-difference scheme with a very fine mesh. The initial evolution of the mode $k = 1$ is shown in Fig. 10, showing the well-known filamentation phenomenon: an initial profile

smooth in velocities, and peaked around $\xi = 0$ in Fourier space, evolves in time along the corresponding characteristics at constant velocity given by k, that is $F_k(\xi, t) = F_k(\xi - kt, 0)$. Thus, the solution propagates toward higher values of $|\xi|$, proportionally fast to the value of $|k|$, therefore faster for shorter wavelengths. Eventually this give rise to the development of small structures in the velocity distribution. So it is observed that the filamentation and mixing of modes appears strongly for long times in the nonlinear regime, and in the Fourier space, specifically for large values of ξ. Therefore, for this case, it has been considered a computational domain large enough in the $\xi-$dimension, despite in the figure it is only shown a part of it. To see more clearly that the solution is closely in agreement with the results obtained using other classical methods, the time evolution of the first harmonic of the electric field obtained by the PDD method is shown in Fig. 11, being qualitatively similar to the typical plots found in the literature [28,17]. Here, $\Delta\xi$ has been kept fixed to 10^{-2} for the local solver. Note again the perfect agreement between the solution obtained by an upwind implicit scheme with a very fine mesh and our PDD method. The absolute numerical error has been numerically computed and it turns out to be or order of 10^{-2} in all simulations done.

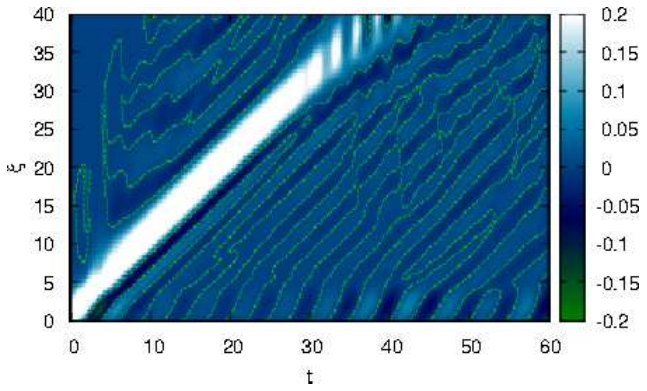


Fig. 10. Strong Landau damping: Time evolution of the numerical solution for $k = 1$. The contour lines correspond to $F_1(\xi, t) = 0$. Parameters are: $A = 0.5$, $\alpha = 2$ and $M = 10^3$.

5.4.2 Two streaming instability

Let consider the following initial condition, chosen for analyzing the two streaming instability phenomenon,

$$f(x, v, 0) = \left(\frac{\alpha}{\pi}\right)^{d/2} 2\alpha\ v^2 e^{-\alpha v^2} \left[1 + A\cos\left(k_1 \cdot x\right)\right], \tag{36}$$

which in Fourier space reads

$$F_k(\xi, 0) = (1 - \frac{\xi^2}{2\alpha}) e^{-\frac{\xi^2}{4\alpha}} [A/2\, \delta(|k| - k_1) + \delta(k)] \tag{37}$$

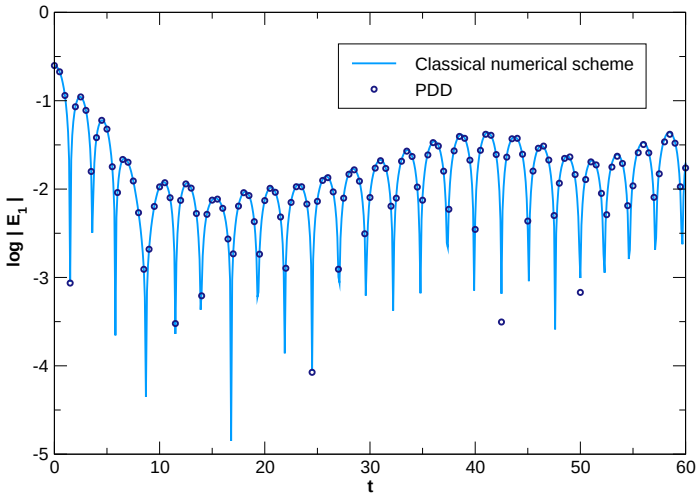
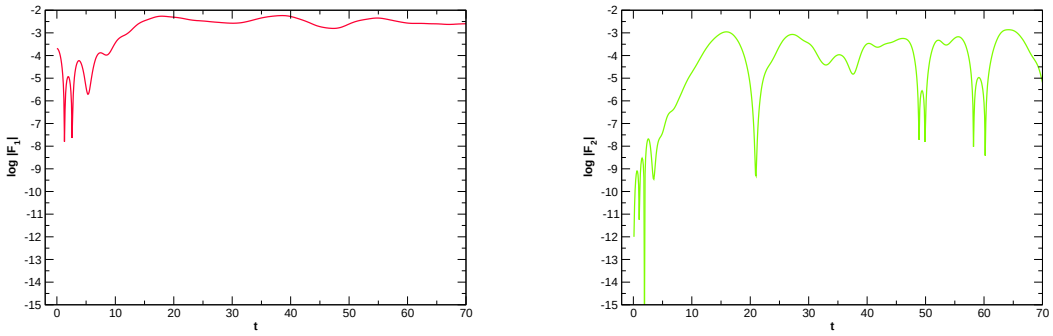


Fig. 11. Strong Landau damping: Logarithm of the absolute value of the first harmonic of the electric field. Parameters are: $A = 0.5$, and $\alpha = 2$. The solution obtained by the PDD method is compared with that obtained with an upwind implicit scheme with a very fine mesh.

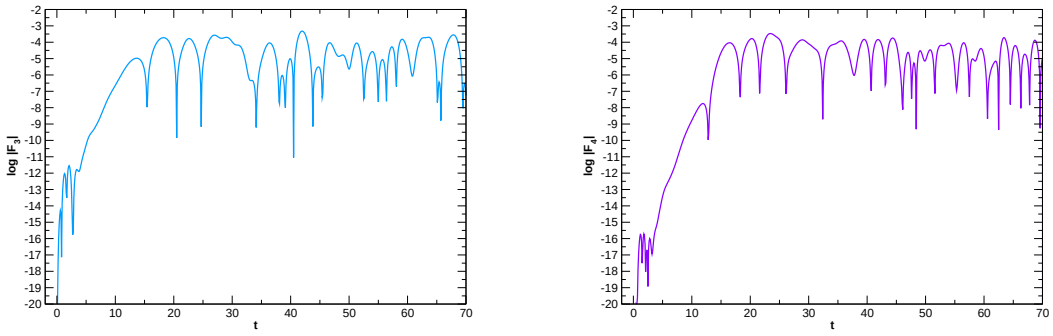
In Fig. 12 it is shown the time evolution of the predominant modes, where an initial, small perturbation leads to a final state characterized by a rapid modes grow and saturation, approximately at the time $t = 20$. As already reported in literature by other authors (e.g., see [28]), the mode-one is the dominant due to its initial excitation and reaches its maximum amplitude at $t = 18$. Once again, the numerical solution has been compared with the solution obtained with an implicit upwind scheme with a very fine mesh, and the numerical error computed as in the previous example, obtaining a similar result.

5.5 Evaluating the performance

In order to validate the computational cost of the algorithm previously derived, as well as its scalability and performance, several tests were run for large scale problems in parallel measuring the computational time required by our PDD algorithm to complete the corresponding simulations. To compare with the results obtained with a classical algorithm, an implicit upwind finite-difference scheme was implemented and conveniently parallelized. The parallelization has been done replacing the sequential subroutine based on LAPACK for solving the banded linear algebra system by the corresponding parallel version based on ScaLAPACK. This consists in a widely used and freely available numerical package, considered extremely efficient for the parallel solution of banded linear systems. To make a fair comparison with the PDD method, LAPACK was



(a) First and second mode amplitude.



(b) Third and fourth mode amplitude.

Fig. 12. Two streaming instability: Logarithm of the absolute values of the first four Fourier modes.

used as local solver for each of the independent subdomains. Both methods, PDD, and the parallel upwind scheme, were compared correspondingly to the same maximum error being of order of 10^{-3}.

All simulations were carried out on the Matrix supercomputer, belonging to the Inter-University Consortium for the Application of Super-Computing for Universities and Research (CASPUR) located in Rome (Italy), using up to 512 processors. This supercomputer consists of a Linux cluster based on multi-core Opteron processor nodes with Infiniband interconnection, and it was ranked in the Top500 list with a peak performance of 22 TFlops.

The particular example chosen for the scalability benchmarks was that corresponding to the strong Landau damping phenomenon. In order to analyze the scalability, two different problem sizes have been considered, one the double of the other, and using 128, 256 and 512 processors, respectively. The results, in

terms of computational time, are given in Table 1 and 2. From the results in Table 1, we can observe a reasonable good scalability and performance of our method, while ScaLAPACK appears to be far from being competitive, suffering in general of strong limitations in terms of memory requirements compared with the PDD algorithm. In fact, when using only 128 processors, the memory requirements of ScaLAPACK for solving the given problem already exceeds the available memory resources, while the PDD method handled successfully such an issue. This has been already reported previously in literature [4] for solving other type of partial differential equations, arising therefore as other important advantage of our algorithm when compared with parallel classical algorithms. The efficient use of the memory resources is an outstanding feature of the the PDD algorithm, allowing to exploit at best the computational resources at hand, and in consequence to be able to tackle large scale problems optimizing the memory consumption.

For a second-round of testing problems, in Table 2 the computational time measured when doubling the problem size has been shown. No results with ScaLAPACK has been reported, since ScaLAPACK was unable to complete the simulations due to the aforementioned memory issues. Note that the scalability results achieved by the PDD method are now significantly better compared with the previous example, being in general excellent in both cases. In view of the estimated computational cost derived previously, this was theoretically expected, since it is known that the computational cost approximately scales as $1/p$ with the number of processors, provided the problem size, measured by N_ξ, is sufficiently large.

Table 1
Example 1: T_{PDD}, $T_{ScaLAPACK}$ denotes the computational time spent in seconds by the PDD method, and ScaLAPACK, respectively. Parameters are: $\xi_{max} = 8 \times 10^6, \Delta\xi = 10^{-1}, T = 2$.

Procs.	$\mathbf{T_{PDD}}$	$\mathbf{T_{ScaLAPACK}}$
128	1109”	-
256	554”	2635”
512	376”	2224”

Table 2
Example 2: T_{PDD}, $T_{ScaLAPACK}$ denotes the computational time spent in seconds by the PDD method. Parameters are: $\xi_{max} = 16 \times 10^6, \Delta\xi = 10^{-1}, T = 2$.

Procs.	$\mathbf{T_{PDD}}$
128	2153”
256	1066”
512	691”

6 Summary

A new numerical method has been presented that, when combined with classical methods, is capable of accelerating the Vlasov-Poisson simulations, thus improving dramatically the overall scalability of classical algorithms. Such method is based on the probabilistic representation of the Vlasov-Poisson system of equations, obtained in Fourier space and generalized to deal with any realistic initial condition. The probabilistic representation allows to compute the solution at single points within the computational domain, and is obtained as the expected value of a multiplicative functional over suitable random trees. Such a feature can be exploited to decouple the original problem into independent subproblems, previously obtaining the required boundary conditions at given interfaces dividing the domain. The probabilistic method was used to compute the solution at a few points, which act as interpolation nodes to obtain the sought boundary conditions at the interfaces. It consists therefore of a straightforward application of the Probabilistic Domain Decomposition(PDD) method for the numerical solution of the Vlasov-Poisson system.

Moreover, the probabilistic representation was validated successfully in the linear regime comparing with the classical results of the linear Landau damping theory. Regarding the numerical implementation of such representation, this requires evaluating in practice a series with terms composed of definite integrals, corresponding to the partial contribution to the solution of random trees with a given number of branches. Typically, the higher terms are of high dimensionality, and were calculated by the quasi-Monte Carlo method. Rather than classical Monte Carlo method, the quasi-Monte Carlo offers a better convergence rate, of order of $O(1/N)$ compared with $O(N^{-1/2})$, speeding up notably the simulations. When dealing with arbitrary initial conditions, such a series might be divergent, and was approximated by the Padé approximant. To study the error made, several test problems were analyzed so far, and it was shown that considering a few coefficients of the series suffices to obtain a reasonable accuracy for sufficiently small times. Since the approximation degrades fast when the time grows, and to avoid including higher terms in the series expansion with the computational cost that this entails, a restarting procedure in time has been proposed. The solution is computed globally in the full domain from time to time, and it is reused as the new initial condition restarting the numerical procedure again. Being now the new initial condition numerically obtained, a suitable interpolation procedure was implemented, which in practice degrades the theoretical expected performance of the algorithm. However, some theoretical considerations were given and applied to the algorithm to improve its overall performance, reducing the computational cost associated to such a global interpolation.

To conclude, several examples were run in parallel and the results compared

with those obtained with classical algorithms. The examples were chosen from the typical repertoire of initial conditions traditionally used for testing numerical methods developed for solving the Vlasov-Poisson system of equations. The results shows the excellent scalability properties of the algorithm proposed when run in large-scale simulations.

It is worth to remark that the method can be further generalized to deal with the Vlasov-Poisson system in configuration space, since the needed probabilistic representation does already exist [42]. Moreover, the probabilistic representation can be combined with any classical existing numerical method according to the procedure described in this paper, improving notably the performance of the resulting algorithm when run in parallel supercomputers.

Acknowledgments

This work was supported by the Portuguese FCT under grants PTDC/EIA-CCO/098910/2008 and SFRH/BD/45362/2008. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Rome Supercomputing center (CASPUR).

References

[1] J. A. Acebrón, M. P. Busico, P. Lanucara, and R. Spigler. Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and Quasi-Monte Carlo methods, SIAM Journal on Scientific Computing. 27(2):440–457, 2005.

[2] J. A. Acebrón, M. P. Busico, P. Lanucara, and R. Spigler. Probabilistically induced domain decomposition methods for elliptic boundary-value problems. Journal of Computational Physics, 210(2):421–438, 2005.

[3] J. A. Acebrón and A. Rodríguez-Rozas. A new parallel solver suited for arbitrary semilinear parabolic partial differential equations based on generalized random trees. Journal of Computational Physics, 230(21):7891–7909, 2011.

[4] J. A. Acebrón, A. Rodríguez-Rozas, and R. Spigler. Domain decomposition solution of nonlinear two-dimensional parabolic problems by random trees. Journal of Computational Physics, 228(15):5574–5591, 2009.

[5] J. A. Acebrón, A. Rodríguez-Rozas, and R. Spigler. Efficient Parallel Solution of Nonlinear Parabolic Partial Differential Equations by a Probabilistic Domain Decomposition. Journal of Scientific Computing, 43(2):135–157, 2010.

[6] W. Petersen, and P. Arbenz, Introduction to parallel computing. A practical guide with examples in C. Oxford Univ. Press, (2004).

[7] G. A. Baker, and P. Graves-Morris, Padé Approximants. New York: Cambridge University Press, 1996.

[8] C. Bender, and S. A. Orszag, Advanced Mathematical Methods for Scientists and Engineers, McGraw Hill, New York (1978)

[9] R. Bhattacharya, L. Chen, S. Dobson, R. B. Guenther, and C. Orum, Majorizing Kernels and Stochastic Cascades with Applications to Incompressible Navier-Stokes Equations. M. Ossiander, E. Thomann, and E. C. Waymire. Reviewed work (s): Source : Transactions of the American Mathematical Society , Vol . 355 , No. Transactions of the American Mathematical Society, 355:5003–5040, 2003.

[10] C. K. Birdsall, A. B. Langdon, Plasma physics via computer simulation, McGraw-Hill, New York, 1985.

[11] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods, Acta Numerica, 7, 1-49, 1998.

[12] W. Morokoff, and R. E. Caflisch, Quasi-Monte Carlo integration, Journal of Computational Physics, 122, 218-231, 1995.

[13] B. Moskowitz, R. E. Caflisch, Smoothness and dimension reduction in Quasi-Monte Carlo methods, Mathematical and Computer Modelling, Volume 23, Issues 8-9, April-May 1996, Pages 37-54.

[14] T. F. Chan, and T. P.Mathew, Domain decomposition algorithms. Acta Numerica (1994), 61-143 [Cambridge University Press, Cambridge, 1994].

[15] A. Chorin, and O. H. Hald, Stochastic tools in Mathematics and Science, Springer (2009).

[16] N. Crouseilles, G. Latu, and E. Sonnendrücker. A parallel Vlasov solver based on local cubic spline interpolation on patches. Journal of Computational Physics, 228(5):1429–1446, 2009.

[17] N. Crouseilles, M. Mehrenberger, and E. Sonnendrücker. Conservative semi-Lagrangian schemes for Vlasov equations. Journal of Computational Physics, 229(6):1927–1953, 2010.

[18] R. C. Davidson, Kinetic Waves and Instabilities in Uniform Plasma, in Handbook of Plasma Physics by A. A. Galeev and R. N. Sudan, Volume 1, pp.521–585. North-Holland Publishing Company (1983).

[19] J. Dongarra, P. Beckman, et al, International Exascale Software Project Roadmap, UT-CS-10-652 (2010)

[20] J. Dongarra, G. Fox, K. Kennedy, L. Torczon, W. Gropp, Foster, I., and White, A.: The Sourcebook of Parallel Computing, Morgan Kaufmann (2002)

[21] B. Eliasson. Outflow Boundary Conditions for the Fourier Transformed One-Dimensional Vlasov Poisson System. Journal of Scientific Computing, 16(1):1–28, 2001.

[22] F. Filbet and E. Sonnendrücker. Comparison of Eulerian Vlasov solvers. Computer Physics Communications, 150:247–266, 2003.

[23] F. Filbet, R. Duclous, and B. Dubroca. Analysis of a high order finite volume scheme for the 1D Vlasov-Poisson system. Discrete and Continuous Dynamical Systems - Series S, 5(2):283–305, 2011.

[24] E. Floriani, R. Lima, and R. Vilela Mendes. Poisson-Vlasov: stochastic representation and numerical codes. The European Physical Journal D, 46(2):295–302, 2007.

[25] M. Freidlin, Functional Integration and Partial Differential Equations. Annals of Mathematics Studies no. 109, Princeton Univ. Press, Princeton (1985)

[26] R. E. Heath, I. M. Gamba, P. J. Morrison, and C. Michler. A discontinuous Galerkin method for the Vlasov-Poisson system. Journal of Computational Physics, 231:1140–1174, 2012.

[27] I. Karatzas, and S. E. Shreve, Brownian Motion and Stochastic Calculus. 2nd ed., Springer, Berlin (1991)

[28] A. J. Klimas. A numerical method based on the Fourier-Fourier transform approach for modeling 1-D electron plasma evolution. Journal of Computational Mathematics, 50:270–306, 1983.

[29] A. J. Klimas. A method for overcoming the velocity space filamentation problem in collisionless plasma model solutions. Journal of Computational Physics, 68:202–226, 1987.

[30] A. J. Klimas. VlasovMaxwell and VlasovPoisson equations as models of a one-dimensional electron plasma. Physics of Fluids, 26(2):478, 1983.

[31] P. M. Kogge, and et al, ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, DARPA Information Processing Techniques Office, Washington, DC, pp.278, September 28, 2008.

[32] H. P. McKean, Application of brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov., Communications on Pure and Applied Mathematics, 28 (1975), 323-331.

[33] G.N. Milstein, and M.V. Tretyakov, Stochastic Numerics for Mathematical Physics. Springer (2004)

[34] H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods, Society for Industrial and Applied Mathematics, Philadelphia, (1992).

[35] A. Quarteroni, and A. Valli, Domain Decomposition Methods for Partial Differential Equations. Oxford Science Publications, Clarendon Press, Oxford (1999)

[36] J.M. Ramirez, Multiplicative cascades applied to PDEs (two numerical examples), Journal of Computational Physics, 214 (2006), 122-136.

[37] H. Regnier, and D. Talay, Special Edition of the Proceedings of the Royal Society on Stochastic Analysis A(460), 199-220, 2004.

[38] J. A. Rossmanith and D. C. Seal. A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the VlasovPoisson equations, Journal of Computational Physics, 230: 6203–6232, 2011.

[39] V. Sarkar, W. Harrod, and A.E. Snavely, Software challenges in extreme scale systems. Journal of Physics: Conference Series pp. 012045 (2009)

[40] I. H. Sloan and H. Wózniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?, Journal of Complexity, 14:133, 1998. (refs: pp. 28, 29, 30)

[41] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. The Semi-Lagrangian Method for the Numerical Resolution of the Vlasov Equation. Journal of Computational Physics, 220:201–220, 1999.

[42] R. Vilela Mendes. Poisson-Vlasov in a strong magnetic field: A stochastic solution approach. Journal of Mathematical Physics, 51(4):043101, 2010.

[43] L. Wuytack, On the conditioning of the Pade approximant problem, Lect. Notes Math. 888 (1981), 78-89.