University of Innsbruck
Department of Mathematics
Numerical Analysis Group

PhD Thesis

Splitting methods for Vlasov–Poisson and Vlasov–Maxwell equations

Lukas Einkemmer

Advisor: Dr. Alexander Ostermann

Submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD).

January 17, 2014

Contents

1 Introduction

A plasma is a highly ionized gas. That is, it describes a state of matter where the electrons dissociate from the much heavier ions. Usually the ionization is the result of a gas that has been heated to a sufficiently high temperature. This is the case for fusion plasmas, including those that exist in the center of the sun, as well as for artificially created fusion plasmas which are, for example, used in magnetic confinement fusion devices (Tokamaks, for example) or in inertial confinement fusion. However, astrophysical plasmas, for example, can exist at low temperatures, since in such configurations the plasma density is low. Therefore, re-absorption of electrons by ionized atoms is relatively unlikely. Also in certain quantum systems (such as in models of metal conductors) the electrons can be modeled as being dissociated from the corresponding ions. From the preceding discussion it is evident that an accurate description of plasma phenomena is of importance to understand an enormous range of physical phenomena.

Due to the nature of a plasma, as an ensemble of charged particles, electromagnetic effects in many instances dominate the plasma dynamics. Therefore, it is vital to include an appropriate description, not only of external fields, but also of the fields that are self-consistently generated by the plasma particles under consideration. The most fundamental classical model for a plasma (to a good approximation) is therefore the Vlasov equation; it describes the evolution of a particle density function subject to a given force. If coupled to an appropriate model of the electromagnetic field (such as the the Poisson equation or Maxwell's equations), the so called Vlasov–Poisson and Vlasov–Maxwell equations result. In some physical situations, simplified models can be derived from the Vlasov–Poisson or the Vlasov–Maxwell equations.

A large body of research on the properties of different plasma systems has been accumulated, since its experimental discovery by William Crookes in 1879. In recent years, supplementing both theoretical analysis as well as experimental results, numerical simulations have become increasingly important in investigating plasma phenomena. Such simulations provide insight into the non-linear character of plasma instabilities, for example, which are difficult to analyze theoretically; furthermore, they provide better access to the entire phase space than is usually possible in experiments. However, due to the difficulty associated with simulating the Vlasov equation, even on modern day computer hardware, many investigations have relied on simplified models or were restricted to a small portion of the physical domain. Nevertheless, the simulations conducted have greatly contributed to the un-

derstanding of the behavior of physical plasmas.

In addition to the computer science improvements that are necessary to simulate larger plasma systems in more detail, it is also vital to develop more efficient numerical algorithms. In order to fulfill that goal, an understanding of the currently used algorithms as well as their limitations is essential. Therefore, it is the goal of this thesis to provide an analysis of the well known Strang splitting scheme for the Vlasov equation. Furthermore, we suggest further lines of investigation that, at the time of writing, are used to design more efficient algorithms for a wider variety of partial differential equations (see section 5).

To conclude this discussion let us note that both the Vlasov–Poisson and the Vlasov–Maxwell equations exhibit a range of interesting and numerically challenging time-dependent phenomena. To just give two examples let us mention filamentation, i.e. the appearance of smaller and smaller scales in phase space, as well a wide variety of physical instabilities, where even a small perturbation of an equilibrium density can result in an interesting evolution. An example of the latter is the so called Landau damping that results in an exponential decay of the electric energy (without an obvious dampening mechanism), if a small perturbation is applied (see e.g. [38] and [27]). However, this is not as clear if larger perturbations are considered (the nonlinear case) as relatively long time numerical simulations do seem to show the appearance of a solution approximately periodic in time (see e.g. [9]).

As part of the research focal point *scientific computing* at the University of Innsbruck, a book on the various research projects in high performance computing (HPC) has been published. The chapter [16] includes a short and less technical introduction to the challenges faced by numerically solving the Vlasov equation. A copy of the before mentioned book chapter is attached to this thesis.

2 The Vlasov equation

The most fundamental theoretical description of a (collisionless) plasma is derived from the kinetic properties of the constituent particles. The result is the so called Vlasov equation as given by (see e.g. [1])

$$\partial_t f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f(t, \boldsymbol{x}, \boldsymbol{v}) = 0,$$

where $\boldsymbol{x}$ denotes the position and $\boldsymbol{v}$ the velocity. The function f describes a particle-probability distribution in the $3 + 3$ dimensional phase space. Since

a plasma interacts with a magnetic field in a non-trivial manner, the Vlasov equation needs to be coupled to the electromagnetic field. Depending on the application either the full Vlasov–Maxwell equations (see e.g. [25] or [2])

$$\begin{aligned}
&\partial_t f(t,\boldsymbol{x},\boldsymbol{v}) + \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f(t,\boldsymbol{x},\boldsymbol{v}) + \frac{e}{m}\left(\boldsymbol{E}+\boldsymbol{v}\times\boldsymbol{B}\right)\cdot\nabla_{\boldsymbol{v}} f(t,\boldsymbol{x},\boldsymbol{v}) = 0\\
&\partial_t \boldsymbol{B} = -\nabla\times\boldsymbol{E}, \qquad \epsilon_0\mu_0\partial_t\boldsymbol{E} = \nabla\times\boldsymbol{B} - \mu_0 eZ\int \boldsymbol{v} f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}\\
&\nabla\cdot\boldsymbol{E} = \frac{eZ}{\epsilon_0}\left(n_0 - \int f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}\right), \qquad \nabla\cdot\boldsymbol{B} = 0
\end{aligned} \tag{2.1}$$

or the Vlasov–Poisson equations (see e.g. [10], [22], or [21])

$$\begin{aligned}
&\partial_t f(t,\boldsymbol{x},\boldsymbol{v}) + \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f(t,\boldsymbol{x},\boldsymbol{v}) + \frac{e}{m}\boldsymbol{E}\cdot\nabla_{\boldsymbol{v}} f(t,\boldsymbol{x},\boldsymbol{v}) = 0\\
&\boldsymbol{E} = -\nabla\Phi, \qquad -\Delta\Phi = \frac{eZ}{\epsilon_0}\left(n_0 - \int f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}\right)
\end{aligned} \tag{2.2}$$

are appropriate. By writing down this model we have made two major assumptions, namely that the plasma is collisionless and that relativistic effects are negligible.

Let us first consider the assumption that the plasma is considered to be collisionless. If that is not the case, a collision term has to be added to the Vlasov equation. The result is usually called the Boltzmann equation. In many applications, however, this is not a serious restriction even if collisional effects are present. Many aspects of relatively dense plasmas, such as those present in magnetic confinement fusion devices, can be analyzed using the Vlasov equation. However, there are certain plasma phenomena that can only be explained if a collision term is added to the model described here.

The second assumption is that relativistic effects are negligible. This basically restrict the electron velocities (the ions usually move much slower due to their large mass) to speeds significantly below the speed of light. Relativistic extensions to the Vlasov–Maxwell equations are necessary, for example, to investigate certain aspects of inertial confinement fusion (see, for example, [33]).

Furthermore, it should be noted that the equations as stated here do represent the evolution for a single species of particles only. One often considers two (or more) distinct species of particles. However, since the interaction between different species of particles is exclusively due to the electromagnetic field, this extension only requires an obvious modification of the source and charge term in the Maxwell and Poisson equations, respectively. Such an extension can be found, e.g., in [2] or [25].

The difficulties in obtaining a numerical solution of the before mentioned equations are summarized in the following three statements:

- Due to the six dimensional phase space the amount of memory required to store the interpolation data is proportional to n^6, where n is the number of grid points per dimension.
- The Vlasov equation is stiff (i.e. the time step size for explicit schemes is limited by the CFL condition); This is explained in some detail in [1], for example.
- The coupling of the Vlasov equation to the Maxwell/Poisson equations makes the system highly non-linear (see e.g. [21]).

From the standpoint of performance the solution of the Maxwell and Poisson equations, respectively, is not a major concern. For given charge and currents this is a problem in a three dimensional space only. A number of efficient algorithms are available to solve such systems.

2.1 The Vlasov equation in relation to simplified models

Due to the difficulties involved in integrating the full Vlasov–Maxwell or Vlasov–Poisson system, a number of simplified equations have been proposed. We will, in what follows, discuss the two most common.

The first approach assumes an equilibrium distribution in velocity space. This leads to a fluid model, called magnetohydrodynamics (MHD), that is described by a system of equations in three dimensional phase space. Such methods are, for example, well suited to address the large scale stability issues inside a Tokamak (see e.g. [35]). This method has also been successfully applied to model astrophysical plasma phenomena such as coronal mass ejections (see e.g. [36]).

A less severe simplification consists in using the gyrokinetic equations which reduce the phase space of the Vlasov equation to $3+2$ dimensions (i.e. three dimension in space and two in velocity) by averaging over the gyro motion. The assumption made in the gyrokinetic equations is that of low frequency as compared to the cyclotron frequency (see e.g. [19]). It should also be noted that the structure of the gyrokinetic equation is quite similar to the full Vlasov equation. Gyrokinetic models have been employed routinely to study plasma instabilities inside Tokamaks (see e.g. [17] or [35]).

Nevertheless, many phenomena in plasma physics (see e.g. [31]) require the solution of the full Vlasov–Maxwell or Vlasov–Poisson equations. Another example is given in [2], where the full Maxwell equations are applied to model the interaction of a laser pulse with a plasma. Also the Weibel instability is investigated in [24] for the 2+1 dimensional Vlasov–Maxwell equations. The direct simulation (that is the use of the six dimensional Vlasov equation or the five dimensional gyrokinetic equations) in a Tokamak geometry has been attempted as well (see e.g. [34] or [8]); however, additional simplifications have to be made in this case. In many instances, the Vlasov equation can be considered in a lower dimensional setting (1+1 dimension for the Vlasov–Poisson equation and 1+2 dimensions for the Vlasov–Maxwell equations, for example). This is also of interest from a purely mathematical standpoint as somewhat less tedious proofs can often be provided in that case, while still capturing the essential properties of the system.

2.2 Dimensionless form

The Vlasov–Poisson equations (2.2) and the Vlasov–Maxwell equations (2.1) are usually rendered into a dimensionless form before numerical simulations are conducted (see e.g. [25]).

For the Vlasov–Poisson equation, using n_0 as a characteristic density, we introduce as characteristic frequency (or timescale) the so called plasma frequency

$$\omega_{pe}^2 = \frac{n_0 \left(eZ\right)^2}{\epsilon_0 m},$$

where eZ is the charge and m the mass of the particles under consideration. In addition, a characteristic length scale has to be chosen. The Debye length is a convenient choice and is given by

$$\lambda_D^2 = \frac{\epsilon_0 m v_{th}^2}{n_0 (eZ)^2},$$

where v_{th} is the thermal speed of the particles under consideration. The thermal speed is a function of the temperature only and thus all plasma parameters can be computed from measurements of density and temperature. As the electric field has units of $\frac{\mathrm{kg\ m}}{\mathrm{s^2 C}}$, the characteristic strength of the electric field is then given by

$$E_0 = \frac{m \lambda_D \omega_{pe}^2}{eZ}.$$

By measuring the particle density function f in units of n_0, time in units of ω_{pe}^{-1}, length in units of λ_D, speed in units of v_{th}, and the electric field in units of E_0, we finally arrive at the dimensionless Vlasov–Poisson equations

$$\begin{aligned} &\partial_t f(t,\boldsymbol{x},\boldsymbol{v}) + \boldsymbol{v}\cdot\nabla f(t,\boldsymbol{x},\boldsymbol{v}) + \boldsymbol{E}\cdot\nabla_{\boldsymbol{v}} f(t,\boldsymbol{x},\boldsymbol{v}) = 0 \\ &\boldsymbol{E} = -\nabla\Phi, \qquad -\Delta\Phi = 1 - \int f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}. \end{aligned} \tag{2.3}$$

Note that this implies that for the Vlasov–Poisson equation no dimensionless parameter remains.

For the Vlasov–Maxwell equation a characteristic strength of the magnetic field is needed. Since the unit of B is $\frac{\mathrm{kg}}{\mathrm{sC}}$ we define

$$B_0 = \frac{m\omega_{pe}}{eZ}.$$

The dimensionless Vlasov–Maxwell equation is then given by

$$\begin{aligned} &\partial_t f(t,\boldsymbol{x},\boldsymbol{v}) + \frac{v_0 t_0}{x_0}\boldsymbol{v}\cdot\nabla f(t,\boldsymbol{x},\boldsymbol{v}) + (\boldsymbol{E}+\boldsymbol{v}\times\boldsymbol{B})\cdot\nabla_{\boldsymbol{v}} f(t,\boldsymbol{x},\boldsymbol{v}) = 0 \\ &\partial_t\boldsymbol{B} = -\nabla\times\boldsymbol{E}, \qquad \epsilon_0\mu_0\partial_t\boldsymbol{E} = \frac{c^2}{v_{th}^2}\left(\nabla\times\boldsymbol{B} - \int \boldsymbol{v} f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}\right) \\ &\nabla\cdot\boldsymbol{E} = 1 - \int f(t,\boldsymbol{x},\boldsymbol{v})\,\mathrm{d}\boldsymbol{v}, \qquad \nabla\cdot\boldsymbol{B} = 0 \end{aligned} \tag{2.4}$$

and it includes a single dimensionless parameter v_{th}/c, i.e. the thermal velocity expressed in units of the speed of light.

3 State of the art

The numerical integration of the Vlasov–Poisson and Vlasov–Maxwell equations has received considerable attention from both physicists and mathematicians. It is the purpose of this section to describe some of the literature relevant to our work.

Methods that are used for the integration of the Vlasov equation can be roughly divided into two classes – particle methods and grid based methods. The class of particle methods as well as some drawbacks are discussed briefly in section 3.1. This section also serves as a motivation for the introduction of grid based methods, which are discussed in section 3.2. Finally, in section 3.3 we describe the (Strang) splitting approach for integrating the Vlasov equation in time.

3.1 Particle approach

The most common approach regarding the integration of the Vlasov as well as the gyrokinetic equations is to employ a so called particle method. In this class of methods, the phase space is left to be continuous and a (large) number of particles with various starting points are advanced in time. This is possible due to structure of the equations, which implies that a single particle evolves along a trajectory given by an ordinary differential equation. This is not surprising as the Vlasov equation is the continuum model that results from considering a very large number of particles that follow the trajectories mandated by Newton's second law of motion. Nevertheless, the Maxwell equations (which are formulated as fields) are solved by standard methods (which involves the discretization of the electric and magnetic field on some spatial grid).

A number of such methods have been developed, among which the probably most widely used is the so called particle-in-cell (PIC) method. Such particle methods have been extensively used for various applications (see e.g. [17]). The PIC scheme gives reasonable results, even for a relatively small number of particles, in cases where the tail of the particle-distribution is not of interest (for the physical phenomenon under consideration). If this is not the case, the method suffers from numerical noise that only decreases as $1/\sqrt{n}$, where n denotes the number of particles (see e.g. [22] or [18]). This is essentially a sampling error, i.e. a consequence of the fact that as the particles are evolved in time, they tend to cluster in parts of the computational domain where the density is high.

3.2 Grid based discretization in phase space

Motivated by the considerations in the previous section, a number of schemes employing a grid based discretization in phase space have been proposed. These methods are completely free from numerical noise (see e.g. [18]). However, due to the finite grid size these methods do not faithfully resolve filamentation (i.e. the appearance of smaller and smaller structures in phase space). However, they usually give more accurate results at the expense of increased computational cost. For an overview of the relative merits of PIC and grid based methods and the dependence of the respective computational cost on the dimension of the problem considered see [3].

The method most closely resembling the particle methods introduced above is the so called semi-Lagrangian approach. Here, one computes the function f at every grid point by following the characteristic curves (i.e. particle

trajectories) backward in time. Obviously, for some given time step the beginning of the characteristic curve does not exactly match a grid point. Thus, a high order interpolation method is necessary to compute the value needed. Semi-Lagrangian methods are, for example, described in [18].

Another class are finite volume methods (sometimes also called flux balance methods). In this case distribution averages on some volume elements are advanced in time by considering the fluxes that enter or leave such a volume. An obvious advantage of such methods is that they are perfectly conservative.

In [18] a number of semi-Lagrangian and finite volume methods are benchmarked for execution speed. It is also found that the PFC (positive and flux conservative) method performs, with respect to accuracy, comparable to semi-Lagrangian schemes, while ENO (essentially nonoscillatory) schemes are too dissipative to be considered a viable alternative.

More recently a number of discontinuous Galerkin methods have been applied to numerically solve the Vlasov–Poisson system of equations. Such schemes employ an approximation that is discontinuous across the cell boundaries. Discontinuous Galerkin methods have been considered for some time for advection dominated problems in computational fluid dynamics (see e.g. [11]).

In [21] the discontinuous flow upwind Galerkin (DFUG) method as well as the discontinuous flow upwind Galerkin nonsymmetric interior penalty Galerkin (DFUG-NIPG) method are introduced. This work is continued in [22], where the upwind penalty Galerkin method (UPG) is proposed as well as benchmarked against a number of analytical results. It is found that discontinuous Galerkin methods give good agreement with the analytic solution even for most non-linear phenomena.

To the best of our knowledge, in almost all of the literature that considers discontinuous Galerkin methods to discretize the Vlasov equation a Runge–Kutta scheme is used to advance the discretized system in time. This, however, is unfortunate as the step size is usually severely limited by the CFL condition.

Methods that combine the discontinuous Galerkin scheme in space with an alternative time integration algorithm have also been proposed. For example, in [25], a second order splitting method is suggested and implemented for a number of interpolation methods. It was found in the before mentioned paper that the scheme constructed using a discontinuous Galerkin approximation is extremely competitive against spline as well as Hermite interpolation.

It should be duly noted that the discontinuous Galerkin method is unique among the finite element methods in that it is able to handle complicated

geometries with relative ease as there is no requirement of continuity at the boundaries. In addition, the interpolation is local so that only data points from a single cell are employed (see e.g. [25]). This usually results in good performance, if such methods are implemented on parallel computing architectures.

3.3 Splitting methods

A splitting scheme for the Vlasov–Poisson equations was first proposed by [10] in 1976. In [25] the method was extended to the Vlasov–Maxwell equations. In both cases the Strang splitting algorithm (see e.g. [23]) is used to advance the solution of the Vlasov equation in time.

In the above mentioned methods the electromagnetic field is kept constant while computing a time step of the Vlasov equation. This is vital since assuming time independence in the force $\boldsymbol{F}$, the Vlasov equation can be simply split in the following two parts

$$\begin{aligned} \partial_t f &= -\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f, \\ \partial_t g &= -\boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} g. \end{aligned} \tag{3.1}$$

These two equations (3.1) can be solved analytically. Their solutions are given by

$$f(t, \boldsymbol{x}, \boldsymbol{v}) = f(0, \boldsymbol{x} - \boldsymbol{v}t, \boldsymbol{v}), \tag{3.2a}$$

$$g(t, \boldsymbol{x}, \boldsymbol{v}) = g(0, \boldsymbol{x}, \boldsymbol{v} - \boldsymbol{F}t). \tag{3.2b}$$

Note, however, for a given discretization in phase space a direct application of (3.2) is not feasible. The discrete data that define the solution are then interpolated, translated according to (3.2), and finally projected back to the discrete space. It should be noted that this method can be interpreted as a special case of the semi-Lagrangian type of methods (as described and analyzed e.g. in [4]) in that the trajectories, say starting from Gauss–Legendre quadrature points, can be solved exactly. However, compared to a finite difference approximation, in the case of discontinuous Galerkin methods this is usually interpreted as a projection, which can be calculated analytically. That is, the solution is translated according to equations (3.2) and then projected back to the approximation space in order to obtain the coefficients of the discontinuous Galerkin approximation.

The situation is more involved in the fully magnetized case (where $\boldsymbol{F}$ depends on $\boldsymbol{v}$). In [25] a further splitting of equation (3.2b) is proposed that takes

advantage of the cross product structure of the Lorentz force term. This is possible as

$$\boldsymbol{v} \times \boldsymbol{B} = \begin{bmatrix} v_2 B_3 - v_3 B_2 \\ v_3 B_1 - v_1 B_3 \\ v_1 B_2 - v_2 B_1 \end{bmatrix}$$

and thus

$$(\boldsymbol{v} \times \boldsymbol{B}) \cdot \nabla_{\boldsymbol{v}} = (v_2 B_3 - v_3 B_2)\, \partial_{v_1} + (v_3 B_1 - v_1 B_3)\, \partial_{v_2} + (v_1 B_2 - v_2 B_1)\, \partial_{v_3}.$$

Since the coefficient of ∂_{v_i} does not depend on v_i itself, it can be easily deduced that the above operators can be split into three advections terms (where, as before, we assume that $\boldsymbol{B}$ is held constant in time).

Using the idea that for a second order scheme the electric and magnetic fields should be computed up to order one at the half step, a similar scheme has been proposed in [32]. However, compared to [25] care is taken such that the global mass is conserved and numerical simulations show that the error in energy is tolerable in some situations.

In general, it is difficult to tell if the methods proposed for the Vlasov–Maxwell equations are indeed of second order. In [25] and [32] neither a detailed mathematical analysis is conducted nor are (numerically computed) order plots available.

The scheme proposed in [32] is implemented using a finite difference approximation in space. A number of different interpolation schemes are compared in [25]. It turns out that interpolation based on the discontinuous Galerkin method is competitive compared to spline and Hermite interpolation. Even though the discontinuous Galerkin scheme employed in [25] is quite different to the one introduced in [21], most of the desirable features remain. Most notably, to interpolate a value of f only the data inside at most two cells are needed.

However, even though the splitting methods described in the literature are routinely employed to study plasma phenomena (see e.g. [24]), a convergence analysis is only available in special cases. The magnetostatic case is considered in [6]. To the best knowledge of the author no convergence analysis is available that includes the fully electromagnetic Vlasov–Maxwell equations. However, a number of geometrical properties (such as the numerical damping rate) are investigated in [25] for the fully magnetized case.

Quite a few convergence results are available for semi-Lagrangian methods that employ Strang splitting in time and a finite difference approximation in space. For example, in [4], [5] and [29] convergence is shown in the case of the

1+1 dimensional Vlasov–Poisson equations. Furthermore, the convergence of a special case of the one-dimensional Vlasov–Maxwell equation in the laser-plasma interaction context is investigated in [7]. Note, however, that the analysis found in these papers is not applicable to the methods described in [25] and [32].

Recently, fourth order splitting methods based on the scheme proposed in [10] (i.e. in the context of the Vlasov–Poisson equation) have also been investigated (see [12] and [30]). Such methods are drawn from the theory of symplectic integrators (see e.g. [37] and [28]). This approach, however, can not be generalized to the Vlasov–Maxwell equations.

In many situations a quite general method is available to construct higher order methods from the second order Strang splitting scheme (see [20, Chap. II.3]). This so called composition methods do, however, require, that the constructed Strang splitting method is of second order and symmetric. To conclude this section, let us note that the schemes proposed for the Vlasov–Maxwell equations (as described in [25] and [32]) do not satisfy the latter condition. However, symmetry does hold for the canonical Strang splitting scheme in case of the Vlasov–Poisson equations.

3.4 Full discretization

Even though semi-Lagrangian methods do inherently mix the errors that occur in the space and time discretization of the problem under consideration, in many instances an analysis of a semi-discretized problem (usually assuming a discretization in time but not in space) can give insight into the behavior of the particular numerical scheme that is studied. In most of the mathematical literature the fully discretized problem is analyzed (see e.g. [4], [5], and [29]). However, the semi-discretized problem (discretized in time) usually has to be employed as a stepping stone in the corresponding proofs. This argument is then supplemented by error estimates that result from the spatial discretization. Since the splitting steps can be represented as translations, this procedure can often be applied and results in error bounds of the form

$$\mathcal{O}\left(\tau^2 + h^{p+1} + \frac{h^{p+1}}{\tau}\right).$$

The first term gives the second order time error of the Strang splitting algorithm, the second term consists of the space discretization error (if an approximation of degree p is used). Most interesting, however, is the third term which gives an error proportional to the number of steps taken. This

is reasonable, in a worst case estimate, as the projections that have to be performed in each time step commit an error that is proportional to the error of the space discretization.

4 Results & Publications

For the numerical analysis conducted as well as for the design of new numerical algorithms we found it beneficial to consider an abstraction that includes, as a special case, both the Vlasov–Poisson as well as the Vlasov–Maxwell equations. To that end we use the following initial valuable problem (in the following discussion any equation that can be written in that form will be refereed to as a Vlasov-type equation)

$$\begin{cases} f'(t) = (A + B)f(t) \\ f(0) = f_0, \end{cases}$$

where A is an (unbounded) linear operator and B is a nonlinear operator that can be written as $Bf = B(f)f$. That is, the nonlinear operator reduces to a linear operator, if we consider $B(f_0)f$ for some fixed f_0. In the Vlasov equation these two operators correspond to the linear advection

$$\partial_t f = -\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = Af,$$

and the non-linear acceleration

$$\partial_t f = -\boldsymbol{F}(f) \cdot \nabla_{\boldsymbol{v}} f = Bf,$$

respectively. In this case we use

$$B(f_0)f = -\boldsymbol{F}(f_0) \cdot \nabla_{\boldsymbol{v}} f.$$

It is always assumed tacitly that there is some algorithm available to efficiently compute a solution to the differential equations corresponding to the operators A and B, respectively. This is obviously satisfied for the Vlasov–Poisson and Vlasov–Maxwell equations as described in the previous section.

Let us note, however, that the class of problems that can be considered in this framework is much larger than just the kinetic equations considered in this thesis. For example, splitting methods for the KdV equation

$$\partial_t u + \partial_x^3 u + u\partial_x u = 0$$

as well as for the Brusselator system

$$\partial_t u = \alpha \Delta u + (uv - \beta)\, u + \delta,$$
$$\partial_t v = \alpha \Delta v - u^2 v + \gamma u,$$

can be described in the abstract framework of a Vlasov-type equation. The computation of an exact solution to the separate split-steps in such equations is, however, often quite different from the procedure that is applied for the Vlasov–Poisson and Vlasov–Maxwell equations (that has been described in the previous section).

This abstraction is advantageous as methods designed for the generic Vlasov-type equations can be applied to a wider class of problems and new numerical schemes can be tested on this simpler examples before implementing them for the full Vlasov–Maxwell equations. It should, however, be noted here that a number of simplifying assumptions can be made for the Vlasov–Poisson equations. Therefore, many methods from the literature tailored to the Vlasov–Poisson equations can not easily be generalized to a generic Vlasov-type equation or even the Vlasov–Maxwell equations.

In the first paper we consider a Vlasov-type equation in the purely abstract framework, i.e. where only the time variable is discretized and the space and velocity degrees of freedom are left continuous. In this case conditions can be derived under which the Strang splitting algorithm for a Vlasov-type equation yields an approximation of second order. Although the detailed statement of these conditions is somewhat technical, most of them can be summarized as an assumption on the regularity of the solution. In addition, the conditions, as an application, are verified to hold for the Vlasov–Poisson equation and numerical experiments have been conducted to confirm the theoretical behavior. This paper [14] has been accepted by the SIAM Journal on Numerical Analysis (SINUM) and a copy is attached to this thesis.

As discussed in the previous section, to combine the Strang splitting algorithm with a discontinuous Galerkin approximation potentially yields a efficient numerical scheme. This is the content of our second paper. In this case the discontinuous Galerkin/Strang splitting approximation is analyzed and in combination with the results from the first paper we were able to establish similar convergence results as are available, for example, for finite difference approximations. In addition, a number of numerical experiments are used to investigate the behavior of that scheme for a number of well known test problems. It is found that the theoretically predicted order in time as well as in space can be observed. Furthermore, the Molenkamp–Crowley test problem has been employed to confirm that the scheme does not suffer from the

instabilities described in [26]. Also the (purely numerical) recurrence phenomena has been investigated in case of higher order discontinuous Galerkin approximations in space (both for the Vlasov–Poisson equations as well as for a simple advection equation). This paper [13] has been submitted to the SIAM Journal on Numerical analysis (SINUM) and a copy is attached to this thesis.

In order to conduct the simulations in the two papers described a numerical program has been written in `C++`. It employs modern programming techniques, such as templates, to obtain a code base that is compact, easily extensible, and can still be considered to provide an efficient implementation. A parallelization using `OpenMP` is available that scales well at least up to a workstation with 16 cores (i.e. a modern dual socket workstation).

The third paper considers the implementation of exponential integrators on graphic processing units (a so called massively parallel architecture). There it is shown that for certain problems, stencil methods (i.e. matrix-free matrix-vector products) can be parallelized to up to 4 GPUs. This is vital in the context of Vlasov solvers as it is clear that because of the memory constraints, both due to the high dimensional phase space as well as due to the limited amount of memory available on GPUs, the storage of any additional data (besides the input and result of the computation) will most likely lead to a numerical method that is infeasible. The procedure developed in the paper yields a significant speedup as compared to a dual socket CPU workstation. This paper [15] has been accepted as a full paper for the Proceedings of the 2013 International Conference on High Performance Computing & Simulation (HPCS 2013).

5 Future research

The Strang splitting scheme has been the main focus in both analyzing convergence as well as in actual implementations. For the Vlasov–Poisson equations high order methods have been constructed (see e.g [12]). However, these methods do rely on assumptions that do not hold true for the Vlasov–Maxwell equations and more generic Vlasov-type equations. As a consequence, such methods can not be generalized to higher order. Most of the literature on the Vlasov–Maxwell equations consists of second order splitting methods that can not be extended to higher order by composition. This is due to the fact that the Strang splitting scheme, contrary to the case of the Vlasov–Poisson equations, is no longer symmetric and thus the triple jump scheme is only of order three.

To remedy this situation, at least two approaches seem promising. First, staying in the abstract framework of Vlasov-type equations, a symmetric scheme can be constructed by composing a Lie step with its adjoint method. This scheme, unfortunately, is implicit and therefore a nonlinear equation has to be solved in each step. Computationally this can be implemented, for example, as a fixed point iteration yielding a scheme that is almost symmetric and therefore amendable to composition. The second option is to consider splitting the Vlasov–Maxwell equations into three parts. Such an approach can, for example, be motivated by the Hamiltonian structure of the Vlasov–Maxwell equations. Higher order methods are then constructed by composition.

Furthermore, it is of interest to provide an efficient parallel implementation of such a scheme as many problems of interest in physics take place in high dimension and/or require high resolution and thus can only be solved on modern supercomputers. In this case it is conjectured that the local nature of the discontinuous Galerkin approximation provides a competitive advantage as compared to finite difference schemes, for example.

In addition, many numerical analysis questions remain to be answered. For example, is it possible to incorporate conservation of energy (in addition to conservation of mass) in such schemes and is it possible to show results for the long time behavior of the Strang splitting scheme. This is a question that, in practical applications, has to addressed by an appropriate combination of a splitting method with some suitable space discretization scheme.

References

[1] E.A. Belli. *Studies of numerical algorithms for gyrokinetics and the effects of shaping on plasma turbulence*. PhD thesis, Princeton University, 2006.

[2] C. Benedetti, P. Londrillo, L. Rossi, and G. Turchetti. Numerical investigation of Maxwell-Vlasov equations – Part I: Basic physics and algorithms. *Commun. Nonlinear Sci. Numer. Simul.*, 13(1):204–208, 2008.

[3] P. Bertrand. Vlasov code applications. In *Proceedings of ISSS*, volume 7, pages 26–31, 2005.

[4] N. Besse. Convergence of a semi-Lagrangian scheme for the one-dimensional Vlasov-Poisson system. *SIAM J. Numer. Anal.*, 42(1):350–382, 2005.

[5] N. Besse. Convergence of a high-order semi-Lagrangian scheme with propagation of gradients for the one-dimensional Vlasov-Poisson system. *SIAM J. Numer. Anal.*, 46(2):639–670, 2008.

[6] N. Besse and M. Mehrenberger. Convergence of classes of high-order semi-Lagrangian schemes for the Vlasov-Poisson system. *Math. Comp.*, 77:93–123, 2008.

[7] M. Bostan and N. Crouseilles. Convergence of a semi-Lagrangian scheme for the reduced Vlasov-Maxwell system for laser-plasma interaction. *Numer. Math.*, 112:169–195, 2009.

[8] A. Bottino, B. Scott, S. Brunner, B. McMillan, T.M. Tran, T. Vernay, L. Villard, S. Jolliet, R. Hatzky, and A. Peeters. Global nonlinear electromagnetic simulations of Tokamak turbulence. *IEEE Transactions on Plasma Science*, 38:2129–2135, 2010.

[9] M. Brunetti, F. Califano, and F. Pegoraro. Asymptotic evolution of nonlinear Landau damping. *Phys. Rev. E*, 62(3):4109, 2000.

[10] C.Z. Cheng and G. Knorr. The integration of the Vlasov equation in configuration space. *J. Comput. Phys.*, 22(3):330–351, 1976.

[11] B. Cockburn and C. W. Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3):173–261, 2001.

[12] N. Crouseilles, E. Faou, and M. Mehrenberger. High order Runge-Kutta-Nyström splitting methods for the Vlasov-Poisson equation. http://hal.inria.fr/inria-00633934/PDF/cfm.pdf.

[13] L. Einkemmer and A. Ostermann. Convergence analysis of a discontinuous Galerkin/Strang splitting approximation for the Vlasov–Poisson equations. *arXiv preprint arXiv:1211.2353*, 2012.

[14] L. Einkemmer and A. Ostermann. Convergence analysis of Strang splitting for Vlasov-type equations. *To appear in SIAM J. Numer. Anal.*, 2013.

[15] L. Einkemmer and A. Ostermann. Exponential Integrators on Graphic Processing units. *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 490–496, 2013.

[16] L. Einkemmer and A. Ostermann. Splitting methods for the Vlasov–Poisson & Vlasov–Maxwell equations. In M. Barden and A. Ostermann, editors, *Scientific Computing @ uibk*. Innsbruck University Press, Innsbruck, 2013.

[17] M.R. Fahey and J. Candy. GYRO: A 5-d gyrokinetic-Maxwell solver. In *Proceedings of the ACM/IEEE SC2004 Conference*, page 26. IEEE, 2008.

[18] F. Filbet and E. Sonnendrücker. Comparison of Eulerian Vlasov solvers. *Computer Physics Communications*, 150(3):247–266, 2003.

[19] E.A. Frieman and L. Chen. Nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria. *Physics of Fluids*, 25:502–508, 1982.

[20] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer-Verlag, Berlin Heidelberg, 2006.

[21] R.E. Heath. *Analysis of the discontinuous Galerkin Method Applied to Collisionless Plasma Physics*. PhD thesis, The University of Texas at Austin, 2007.

[22] R.E. Heath, I.M. Gamba, P.J. Morrison, and C. Michler. A discontinuous Galerkin method for the Vlasov-Poisson system. *J. Comput. Phys.*, 2011.

[23] T. Jahnke and C. Lubich. Error bounds for exponential operator splittings. *BIT Numer. Math.*, 40:735–744, 2000.

[24] F. Pegoraro L. Palodhi, F. Califano. Nonlinear kinetic development of the Weibel instability and the generation of electrostatic coherent structures. *Plasma Phys. Contr. F.*, 51(12):125006, 2009.

[25] A. Mangeney, F. Califano, C. Cavazzoni, and P. Travnicek. A numerical scheme for the integration of the Vlasov-Maxwell system of equations. *J. Comput. Phys.*, 179:495–538, 2002.

[26] K.W. Morton, A. Priestley, and E. Süli. Stability of the Lagrange-Galerkin method with non-exact integration. *Modél. Math. Anal. Numér.*, 22(4):625–653, 1988.

[27] C. Mouhot and C. Villani. On Landau damping. *Acta math.*, 207(1):29–201, 2011.

[28] E. Pohn, M. Shoucri, and G. Kamelander. Eulerian Vlasov codes. *Comput. Phys. Commun.*, 166(2):81–93, 2005.

[29] T. Respaud and E. Sonnendrücker. Analysis of a new class of forward semi-Lagrangian schemes for the 1D Vlasov Poisson equations. *Numer. Math.*, 118:329–366, 2011.

[30] J.A. Rossmanith and D.C. Seal. A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov-Poisson equations. *J. Comput. Phys.*, 230(16):6203–6232, 2011.

[31] H. Schamel. Electron holes, ion holes and double layers: electrostatic phase space structures in theory and experiment. *Phys. Rep.*, 140(3):161–191, 1986.

[32] N. J. Sircombe and T.D. Arber. VALIS: A split-conservative scheme for the relativistic 2D Vlasov-Maxwell system. *J. Comput. Phys.*, 228:4773–4788, 2009.

[33] A. Suzuki and T. Shigeyama. A conservative scheme for the relativistic Vlasov-Maxwell system. *J. Comput. Phys.*, 229:1643–1660, 2010.

[34] V.A. Svidzinski and D.G. Swanson. Possibility of a direct solution of Vlasov–Maxwell equations in Tokamaks in the ion cyclotron range of frequencies. *Phys. Plasmas*, 8(2), 2000.

[35] W.M. Tang and V.S. Chan. Advances and challenges in computational plasma science. *Plasma Phys. Control. Fusion*, 47:R1–R34, 2005.

[36] M. Tokman and P.M. Bellan. Three-dimensional model of the structure and evolution of coronal mass ejections. *Astrophysical J.*, 567(2):1202, 2002.

[37] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, 150(5-7):262–268, 1990.

[38] T. Zhou, Y. Guo, and C.W. Shu. Numerical study on Landau damping. *Phys. D.*, 157(4):322–333, 2001.

A Convergence analysis of Strang splitting for Vlasov-type equations

Journal	SIAM Journal on Numerical Analysis
Authors	Lukas Einkemmer, Alexander Ostermann
submitted	10.07.2012
accepted	11.10.2013

CONVERGENCE ANALYSIS OF STRANG SPLITTING FOR VLASOV-TYPE EQUATIONS

LUKAS EINKEMMER* AND ALEXANDER OSTERMANN*

Abstract. A rigorous convergence analysis of the Strang splitting algorithm for Vlasov-type equations in the setting of abstract evolution equations is provided. It is shown that, under suitable assumptions, the convergence is of second order in the time step τ. As an example, it is verified that the Vlasov–Poisson equations in 1+1 dimensions fit into the framework of this analysis. Further, numerical experiments for the latter case are presented.

Key words. Strang splitting, abstract evolution equations, convergence analysis, Vlasov–Poisson equations, Vlasov-type equations

AMS subject classifications. 65M12, 82D10, 65L05

1. Introduction.

The most fundamental theoretical description of a (collisionless) plasma comes from the kinetic equation. This so called Vlasov equation is given by (see e.g. [1])

$$\partial_t f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f(t, \boldsymbol{x}, \boldsymbol{v}) = 0,$$

where $\boldsymbol{x}$ denotes the position and $\boldsymbol{v}$ the velocity. The function f describes a particle-probability distribution in the $3+3$ dimensional phase space. Since a plasma interacts with the electromagnetic field in a non-trivial manner, the Vlasov equation needs to be coupled to the electromagnetic field through the force term $\boldsymbol{F}$. A one-dimensional example is given in section 4 below.

Depending on the application, either the full Vlasov–Maxwell equations or a simplified model is appropriate. Such models include, for example, the Vlasov–Poisson and the gyrokinetic equations.

Due to the high dimensionality of the equations the most common numerical approach are so called particle methods. In this class of methods, the phase space is left to be continuous and a (large) number of particles with various starting points are advanced in time. This is possible due to the structure of the equations, which implies that a single particle evolves along a trajectory given by an ordinary differential equation. A number of such methods have been developed, most notably the particle-in-cell (PIC) method. Such methods have been extensively used for various applications (see e.g. [7]). The PIC scheme gives reasonable results in case where the tail of the distribution is negligible. If this is not the case the method suffers from numerical noise that only decreases as $1/\sqrt{n}$, where n denotes the number of particles (see e.g. [12] or [8]). Motivated by these considerations, a number of schemes employing discretization in phase space have been proposed. A comparison of various such methods can be found in [8].

Using a time splitting scheme for the Vlasov–Poisson equations was first proposed by [5] in 1976. In [16] the method was extended to the Vlasov–Maxwell equations. In both cases, second-order Strang splitting (see e.g. [14]) is used to advance the solution of the Vlasov equation in time.

*Department of Mathematics, University of Innsbruck, Technikerstraße 13, Innsbruck, Austria (lukas.einkemmer@uibk.ac.at, alexander.ostermann@uibk.ac.at). The first author was supported by a scholarship of the Vizerektorat für Forschung, University of Innsbruck, and by the Austrian Science Fund (FWF), project id: P25346.

Quite a few convergence results are available for semi-Lagrangian methods that employ Strang splitting. For example, in [2], [3] and [17] convergence is shown in the case of the 1+1 dimensional Vlasov–Poisson equations. Both [2] and [3] assume the same analytical framework, regarding the regularity of the solution, that we employ in section 4. However, the convergence proofs presented in these papers are based on the method of characteristics and are valid only if certain assumptions are made, which hold for the Vlasov–Poisson equations in combination with the specific scheme under consideration in those papers. This is in contrast to our analysis, as we, for example, do not limit ourselves to a specific form of the auxiliary method (the technical details of this will be apparent in section 2.1). The resulting convergence results for the Vlasov–Poisson equations, however, are similar to what we derive in section 4. Furthermore, the convergence of a special case of the one-dimensional Vlasov–Maxwell equation in the laser-plasma interaction context is investigated in [4].

In this paper, we will consider a class of Vlasov-type equations as abstract evolution equations (i.e., without discretization in space). In this context we will derive sufficient conditions such that the Strang splitting algorithm is convergent of order 2. We will then verify these conditions for the example of the Vlasov–Poisson equations in 1+1 dimensions and present some numerical results.

2. Setting. We will investigate the following (abstract) initial value problem

$$\begin{cases} f'(t) = (A + B)f(t) \\ f(0) = f_0. \end{cases} \tag{2.1}$$

We assume that A is an (unbounded) linear operator and that the non-linearity B has the form $Bf = B(f)f$, where $B(f)$ is an (unbounded) linear operator. We will consider this abstract initial value problem on a finite time interval $[0, T]$.

Problem (2.1) comprises the Vlasov–Poisson and the Vlasov–Maxwell equations for $A = -\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}}$ and appropriately chosen B as special cases. It is also general enough to include the gyrokinetic equations (as stated, for example, in [10]). The Vlasov–Poisson equations are considered in more detail in section 4.

2.1. The Strang splitting algorithm. Let $f_k \approx f(t_k)$ denote the numerical approximation to the solution of (2.1) at time $t_k = k\tau$ with step size τ. We assume that the differential equations $f' = Af$ and $g' = B_{k+1/2}g$, where $B_{k+1/2}$ is a suitable approximation to the operator $B\left(f(t_k + \frac{\tau}{2})\right)$, can be solved efficiently. In this paper we always make the choice $B_{k+1/2} = B(f_{k+1/2})$, where

$$f_{k+1/2} = \Psi(\tfrac{\tau}{2}, f_k) \tag{2.2}$$

is a first-order approximation to the solution of (2.1) at time $t = t_k + \frac{\tau}{2}$. Note that $f_{k+1/2}$ typically depends on f_k only. In the case of the Vlasov–Poisson equations, an appropriate choice is

$$f_{k+1/2} = \mathrm{e}^{\frac{\tau}{2}B(f_k)}\mathrm{e}^{\frac{\tau}{2}A} f_k$$

or even $f_{k+1/2} = \mathrm{e}^{\frac{\tau}{2}A} f_k$, as will be explained in the first paragraph of section 5.

The idea of Strang splitting is to advance the numerical solution by the recursion $f_{k+1} = S_k f_k$, where the (nonlinear) splitting operator S_k is given by

$$S_k = \mathrm{e}^{\frac{\tau}{2}A}\mathrm{e}^{\tau B_{k+1/2}}\mathrm{e}^{\frac{\tau}{2}A}. \tag{2.3}$$

The precise conditions on $f_{k+1/2}$ for proving convergence are given in section 3 below. Resolving this recursion, we can compute an approximation to the exact solution at time T by

$$f_n = \left(\prod_{k=0}^{n-1} S_k \right) f_0 = S_{n-1} \cdots S_0 f_0, \tag{2.4}$$

where n is an integer chosen together with the step size τ such that $T = n\tau$.

2.2. Preliminaries. For the convenience of the reader we collect some well known results that are used quite extensively in section 3.

To bound the remainder term $R_k(f)$ of a Taylor expansion

$$f(\tau) = f(0) + \tau f'(0) + \ldots + \frac{\tau^{k-1}}{(k-1)!} f^{(k-1)}(0) + \tau^k R_k(f),$$

we will use the integral form

$$R_k(f) = \frac{1}{(k-1)!} \int_0^1 f^{(k)}(\tau s)(1-s)^{k-1} \, \mathrm{d}s,$$

where $k \geq 1$. Note that it is implicitly understood that R_k is a function of τ as well. However, since we will work mostly with a fixed τ, it is convenient to drop it in the notation of R_k. For convenience we also define

$$R_0(f) = f(\tau).$$

For (unbounded) linear operators it is more convenient to work with the φ functions instead of the remainder term given above.

DEFINITION 2.1 (φ functions). *Suppose that the linear operator E generates a $\mathcal{C}_0$ semigroup. Then we define the bounded operators*

$$\begin{aligned} \varphi_0(\tau E) &= \mathrm{e}^{\tau E}, \\ \varphi_k(\tau E) &= \int_0^1 \mathrm{e}^{(1-\theta)\tau E} \frac{\theta^{k-1}}{(k-1)!} \, \mathrm{d}\theta \quad \text{for } k \geq 1. \end{aligned} \tag{2.5}$$

Since we are merely interested in bounds of such functions, we will never directly employ the definition given. Instead we will work exclusively with the following recurrence relation.

LEMMA 2.2. *The φ functions satisfy the recurrence relation*

$$\varphi_k(\tau E) = \frac{1}{k!} + \tau E \varphi_{k+1}(\tau E), \quad k \geq 0 \tag{2.6}$$

in X; in particular (for $\ell \in \mathbb{N}$) the representation

$$\mathrm{e}^{\tau E} = \sum_{k=0}^{\ell-1} \frac{\tau^k}{k!} E^k + \tau^\ell E^\ell \varphi_\ell(\tau E).$$

holds in the domain of $E^{\ell-1}$.

Proof. The first relation follows from integration by parts applied to (2.5). The second one results from using $\varphi_0 = \mathrm{e}^{(\cdot)}$ and applying the first relation repeatedly. ∎

The φ functions are used to expand the exponential of some linear operator. In the sense of the previous lemma, these functions play the same role for an exponential of a linear operator as does the remainder term in Taylor's theorem.

Suppose that the differential equation $g' = G(g)$ has (for a given initial value) a unique solution. In this case we denote the solution at time t with initial value $g(t_0) = g_0$ with the help of the evolution operator, i.e. $g(t) = E_G(t - t_0, g_0)$.

The Gröbner–Alekseev formula (also called the nonlinear variation-of-constants formula) will be employed quite heavily.

THEOREM 2.3 (Gröbner–Alekseev formula). *Suppose that there exists a unique f satisfying*

$$\begin{cases} f'(t) = G(f(t)) + R(f(t)) \\ f(0) = f_0 \end{cases}$$

and that $g' = G(g)$ has (for a given initial value) a unique solution. Then it holds that

$$f(t) = E_G(t, f_0) + \int_0^t \partial_2 E_G\left(t - s, f(s)\right) R\left(f(s)\right) \mathrm{d}s.$$

Proof. For linear (and possibly unbounded) G, this formula is proved in [13] by the fundamental theorem of calculus. Here, we prove the extension to nonlinear G. Let us assume that $u(t)$ is a solution of $u'(t) = G\left(u(t)\right)$. By differentiating

$$E_G\left(t - s, u(s)\right)) = u(t)$$

with respect to s we get

$$-\partial_1 E_G\left(t - s, u(s)\right) + \partial_2 E_G\left(t - s, u(s)\right) G\left(u(s)\right) = 0.$$

The initial value of u is now chosen such that $u(s) = f(s)$ which implies

$$-\partial_1 E_G\left(t - s, f(s)\right) + \partial_2 E_G\left(t - s, f(s)\right) G\left(f(s)\right) = 0.$$

Altogether we have for $\psi(s) = E_G(t-s, f(s))$ (by the fundamental theorem of calculus)

$$\begin{aligned} f(t) - E_G(t, f_0) &= \int_0^t \psi'(s) \, \mathrm{d}s \\ &= \int_0^t \Big(-\partial_1 E_G(t - s, f(s)) + \partial_2 E_G(t - s, f(s)) f'(s) \Big) \, \mathrm{d}s \\ &= \int_0^t \partial_2 E_G\left(t - s, f(s)\right) R\left(f(s)\right) \, \mathrm{d}s, \end{aligned}$$

as desired. □

Since anticommutator relations appear quite naturally in some expansions, we will employ the notation

$$\{E_1, E_2\} = E_1 E_2 + E_2 E_1,$$

for linear operators E_1 and E_2 (on a suitable domain).

In what follows C will denote a generic constant that may have different values at different occurrences.

3. Convergence analysis in the abstract setting. The problem of splitting an evolution equation into two parts, governed by linear and possibly unbounded operators, has already been investigated in some detail. In [11] it is shown that splitting methods with a given classical order retain this order in the stiff case (under suitable regularity assumptions).

An alternative analysis for Strang splitting in the linear case is given in [14]. The approach presented there is more involved, however, it demands less regularity on the solution. The purpose of this section is to extend this analysis to the abstract initial value problem given by (2.1).

3.1. Convergence. Our convergence proof will be carried out in an abstract Banach space X with norm $\|\cdot\|_X$. It relies on the classical concepts of consistency and stability. We begin by stating a suitable notion of consistency for our splitting operator. For this purpose, let

$$\widetilde{B}_{k+1/2} = B\left(f(t_k + \tfrac{\tau}{2})\right)$$

denote the non-linearity, evaluated at the exact solution at time $t_k + \frac{\tau}{2}$. With the help of this operator, we consider the modified scheme

$$\widetilde{S}_k = \mathrm{e}^{\frac{\tau}{2}A}\mathrm{e}^{\tau\widetilde{B}_{k+1/2}}\mathrm{e}^{\frac{\tau}{2}A}.$$

We are now in the position to define consistency for our numerical method.

DEFINITION 3.1 (Consistency of order p). *The Strang splitting algorithm* (2.3) *is consistent of order p if*

$$\|f(t_k + \tau) - \widetilde{S}_k f(t_k)\|_X \leq C\tau^{p+1}. \tag{3.1}$$

The constant C depends on the considered problem but is independent of τ and k for $0 \leq t_k = k\tau \leq T$.

Note that for algorithm (2.3), the order of consistency is not necessarily $p = 2$. The actual order depends on the properties of the involved operators, and order reduction can occur even in the linear case, see [14].

To estimate the global error, i.e. $f_{k+1} - f(t_{k+1})$, we employ the error recursion

$$f_{k+1} - f(t_{k+1}) = S_k\big(f_k - f(t_k)\big) + (S_k - \widetilde{S}_k)f(t_k) + \widetilde{S}_k f(t_k) - f(t_{k+1}). \tag{3.2}$$

The first two terms on the right-hand side of (3.2) are controlled by the linear and non-linear stability properties of the method, whereas the last difference is controlled by the consistency bound. For our abstract convergence result, we have to assume the stability bound

$$\|S_k\|_{X\leftarrow X} \leq 1 + C\tau \tag{3.3}$$

and the Lipschitz condition

$$\|S_k - \widetilde{S}_k\big\|_{X\leftarrow X} \leq C\tau\big\|f_{k+1/2} - f(t_k + \tfrac{\tau}{2})\big\|_X \tag{3.4}$$

with a constant C that is uniform in τ and k for $0 \leq t_k = k\tau \leq T$. These bounds will be verified in section 4.4 for the particular case of the Vlasov–Poisson equations.

We are now in the position to bound the global error.

THEOREM 3.2 (Convergence). *Suppose that the Strang splitting scheme* (2.3) *is consistent of order p and satisfies the bounds* (3.3) *and* (3.4). *Further assume*

that the auxiliary method (2.2) *is consistent of order* $p-1$ *and (locally) Lipschitz continuous with respect to its second argument. Then the Strang splitting scheme* (2.3) *is convergent of order* p*, i.e.*

$$\|f_k - f(t_k)\|_X \leq C\tau^p \tag{3.5}$$

with a constant C *that is independent of* τ *and* k *for* $0 \leq t_k = k\tau \leq T$*.*

Proof. The proof is quite standard. We apply the triangle inequality to the error recursion (3.2) and insert the bounds (3.3), (3.4), and the consistency bound (3.1). By our assumptions on method (2.2), we further obtain

$$\begin{aligned}\|f_{k+1/2} - f(t_k + \tfrac{\tau}{2})\|_X &= \|\Psi(\tfrac{\tau}{2}, f_k) - \Psi(\tfrac{\tau}{2}, f(t_k)) + \Psi(\tfrac{\tau}{2}, f(t_k)) - f(t_k + \tfrac{\tau}{2})\|_X \\ &\leq C\|f_k - f(t_k)\|_X + C\tau^p.\end{aligned}$$

This finally results in the recursion

$$\|f_{k+1} - f(t_{k+1})\|_X \leq (1 + C\tau)\|f_k - f(t_k)\|_X + C\tau^{p+1}$$

which is easily solved. Employing $f_0 = f(0)$ we obtain the desired bound. ∎

3.2. Consistency. It is the purpose of this section to formulate assumptions under which the consistency bound (3.1) holds for the abstract initial value problem (2.1). To make the derivations less tedious we will adhere to the notation laid out in the following remark.

Remark 3.3. *In this section we will denote the solution of* (2.1) *at a fixed time* t_k *by* f_0*. The notation* $f(s)$ *is then understood to mean* $f(t_k + s)$*. The function* f_0 *is a (possible) initial value for a single time step (i.e., a single application of the splitting operator* S_k*). It is not, in general, the initial value of the solution to the abstract initial value problem as in the previous sections. If we assert that a bound holds uniformly in* t_k*, it is implied that it holds for all* f_0 *in the sense defined here (remember that* $t_k \in [0, T]$*). Since* t_k *is fixed we will use the notation* $\widetilde{B}$ *and* $\widetilde{S}$ *instead of* $\widetilde{B}_{k+1/2}$ *and* $\widetilde{S}_k$*, respectively.*

Let us start with expanding the exact solution by using the Gröbner–Alekseev formula (this has been proposed in the context of the nonlinear Schrödinger equation in [15]). We consider the linear operator A as a perturbation of the differential equation given by the non-linear operator B. This choice is essential for the treatment given here, since it allows us to apply the expansion sequentially without any additional difficulties.

Lemma 3.4 (Expansion of the exact solution). *The exact solution of* (2.1) *has the formal expansion*

$$\begin{aligned}f(\tau) = E_B(\tau, f_0) &+ \int_0^\tau \partial_2 E_B(\tau - s, f(s)) A E_B(s, f_0)\,\mathrm{d}s \\ &+ \int_0^\tau \int_0^s \partial_2 E_B(\tau - s, f(s)) A \partial_2 E_B(s - \sigma, f(\sigma)) A E_B(\sigma, f_0)\,\mathrm{d}\sigma\mathrm{d}s \\ &+ \int_0^\tau \int_0^{\sigma_1} \int_0^{\sigma_2} \left(\prod_{k=0}^{2} \partial_2 E_B(\sigma_k - \sigma_{k+1}, f(\sigma_{k+1})) A \right) f(\sigma_3)\,\mathrm{d}\sigma_3\mathrm{d}\sigma_2\mathrm{d}\sigma_1,\end{aligned}$$

where $\sigma_0 = \tau$*.*

Proof. Apply the Gröbner–Alekseev formula three times to equation (2.1). ∎

Next we expand the splitting operator $\widetilde{S}$ in a form that is suitable for comparison with the exact solution.

LEMMA 3.5 (Expansion of the splitting operator). *The splitting operator $\widetilde{S}$ has the formal expansion*

$$\widetilde{S}f_0 = \mathrm{e}^{\tau\widetilde{B}}f_0 + \frac{\tau}{2}\left\{A, \mathrm{e}^{\tau\widetilde{B}}\right\}f_0 + \frac{\tau^2}{8}\left\{A, \left\{A, \mathrm{e}^{\tau\widetilde{B}}\right\}\right\}f_0 + R_3 f_0,$$

where

$$R_3 = \frac{\tau^3}{16}\int_0^1 (1-\theta)^2 \left\{A, \left\{A, \left\{A, \mathrm{e}^{\frac{\tau\theta}{2}A}\mathrm{e}^{\tau\widetilde{B}}\mathrm{e}^{\frac{\tau\theta}{2}A}\right\}\right\}\right\} \mathrm{d}\theta.$$

Proof. Let us define the function $g(s) = \mathrm{e}^{\frac{1}{2}sA}\mathrm{e}^{\tau\widetilde{B}}\mathrm{e}^{\frac{1}{2}sA}$. The first three derivatives of g are given by

$$\begin{aligned} g'(s) &= \frac{1}{2}\{A, g(s)\}, \\ g''(s) &= \frac{1}{4}\{A, \{A, g(s)\}\}, \\ g^{(3)}(s) &= \frac{1}{8}\{A, \{A, \{A, g(s)\}\}\}. \end{aligned}$$

From the observation that $\widetilde{S} = g(\tau)$ and by Taylor's theorem we obtain the result. ∎

Let us now give the conditions which, if satisfied, imply that the Strang splitting scheme, in our abstract setting, is consistent of order two.

THEOREM 3.6 (Consistency). *Suppose that the estimates*

$$\left\|\varphi_1^{\delta_{i1}}(\widetilde{B})\left(B(E_B(\tfrac{\tau}{2}, f_0)) - \tilde{B}\right) R_1^{\delta_{i0}}\left(E_B(\cdot, f_0)\right)\right\|_X \le C\tau, \quad i \in \{0,1\} \tag{3.6}$$

$$\sup_{0\le s\le\tau}\left\|\frac{\mathrm{d}^2}{\mathrm{d}s^2}\mathrm{e}^{s\widetilde{B}}\big(B(E_B(s,f_0)) - \widetilde{B}\big)E_B(s,f_0)\right\|_X \le C, \tag{3.7}$$

$$\left\|\left[B\left(E_B(\tfrac{\tau}{2}, f_0)\right) - \widetilde{B} + \tfrac{\tau}{2}B'(Af_0)\right]f_0\right\|_X \le C\tau^2, \tag{3.8}$$

and

$$\sup_{0\le s\le\tau}\left\|A^i\mathrm{e}^{(\tau-s)\widetilde{B}}(B-\widetilde{B})E_B(s,f_0)\right\|_X \le C\tau^{2-i}, \quad i \in \{1,2\} \tag{3.9}$$

$$\left\|(B(f_0) - \widetilde{B})Af_0\right\|_X \le C\tau, \tag{3.10}$$

$$\left\|A^{\delta_{i2}}\widetilde{B}^{1+\delta_{i0}}\varphi_{1+\delta_{i0}}(\tau\widetilde{B})A^{1+\delta_{i1}}f_0\right\|_X \le C, \quad i \in \{0,1,2\} \tag{3.11}$$

$$\left\|A^{\delta_{i2}}R_{1+\delta_{i0}}(\partial_2 E_B(\cdot, f_0))A^{1+\delta_{i1}}f_0\right\|_X \le C, \quad i \in \{0,1,2\} \tag{3.12}$$

hold uniformly in t, where δ_{ij} denotes the Kronecker delta. In addition, suppose that

the estimates

$$\sup_{0\le s\le\tau}\left\|\frac{\mathrm{d}^2}{\mathrm{d}s^2}\Big(\partial_2 E_B(\tau-s,f(s))AE_B(s,f_0)\Big)\right\|_X\le C, \quad (3.13)$$

$$\sup_{0\le\sigma\le s\le\tau}\left\|\frac{\partial}{\partial s}\Big(\partial_2 E_B(\tau-s,f(s))A\partial_2 E_B(s-\sigma,f(\sigma))AE_B(\sigma,f_0)\Big)\right\|_X\le C, \quad (3.14)$$

$$\sup_{0\le\sigma\le s\le\tau}\left\|\frac{\partial}{\partial \sigma}\Big(\partial_2 E_B(\tau-s,f(s))A\partial_2 E_B(s-\sigma,f(\sigma))AE_B(\sigma,f_0)\Big)\right\|_X\le C, \quad (3.15)$$

$$\left\|\Big(\prod_{k=0}^{2}\partial_2 E_B(\sigma_k-\sigma_{k+1},f(\sigma_{k+1}))A\Big)f(\sigma_3)\right\|_X\le C, \quad (3.16)$$

$$\sup_{0\le s\le\tau}\left\|\left\{A,\left\{A,\left\{A,e^{\frac{s}{2}A}e^{\tau B}e^{\frac{s}{2}A}\right\}\right\}\right\}f_0\right\|_X\le C, \quad (3.17)$$

hold uniformly in t for $0\le\sigma_3\le\sigma_2\le\sigma_1\le\sigma_0=\tau$.

Then the Strang splitting (2.3) *is consistent of order 2.*

Proof. We have to compare terms of order 0, 1, and 2 in Lemma 3.4 and Lemma 3.5 and show that the remaining terms of order 3 can be bounded as well.

Terms of order 0. We have to bound the difference

$$\mathrm{e}^{\tau\widetilde{B}}f_0-E_B(\tau,f_0). \quad (3.18)$$

For this purpose we denote $E_B(s,f_0)$ by $u(s)$ and make use of the fact that u satisfies the differential equation

$$u'=\widetilde{B}u+(B-\widetilde{B})u$$

with initial value f_0. Employing the variation-of-constants formula we get

$$u(\tau)=\mathrm{e}^{\tau\widetilde{B}}f_0+\int_0^\tau \mathrm{e}^{(\tau-s)\widetilde{B}}(B-\widetilde{B})E_B(s,f_0)\,\mathrm{d}s.$$

Now let us employ the midpoint rule; this yields

$$\begin{aligned}u(\tau)-\mathrm{e}^{\tau\widetilde{B}}f_0&=\tau\mathrm{e}^{\frac{\tau}{2}\widetilde{B}}\left(B(u(\tfrac{\tau}{2}))-B(f(\tfrac{\tau}{2}))\right)u(\tfrac{\tau}{2})+d\\&=\tau\left(B(u(\tfrac{\tau}{2}))-B(f(\tfrac{\tau}{2}))\right)f_0+\tfrac{\tau^2}{2}\varphi_1(\widetilde{B})\left(B(E_B(\tfrac{\tau}{2}))-\widetilde{B}\right)\\&\quad+\tfrac{\tau^2}{2}\left(B(E_B(\tfrac{\tau}{2}))-\widetilde{B}\right)R_1\left(E_B(\cdot,f_0)\right)+d.\end{aligned}$$

The second term is bounded by assumption (3.6) and the remainder term by assumption (3.7). We postpone the discussion of the first term until we have considered the terms of order 1.

Terms of order 1. For

$$g(s)=\mathrm{e}^{(\tau-s)\widetilde{B}}A\mathrm{e}^{s\widetilde{B}}f_0,\qquad k(s)=\partial_2 E_B(\tau-s,f(s))AE_B(s,f_0)$$

we get (by use of the trapezoidal rule)

$$\begin{aligned}&\frac{\tau}{2}\Big(g(0)+g(\tau)\Big)-\int_0^\tau k(s)\,\mathrm{d}s\\&\qquad=\frac{\tau}{2}\Big(g(0)-k(0)+g(\tau)-k(\tau)\Big)-\frac{\tau^3}{2}\int_0^1\theta(1-\theta)k''(\theta\tau)\,\mathrm{d}\theta.\end{aligned}$$

First, let us compare $g(\tau)$ and $k(\tau)$

$$g(\tau) - k(\tau) = A\big(\mathrm{e}^{\tau\widetilde{B}} f_0 - E_B(\tau, f_0)\big),$$

which is the same term that we encountered in (3.18), except that we have an additional A to the left of the expression. We thus can apply assumption (3.9) with $i = 1$. Second, we have to compare $g(0)$ and $k(0)$

$$g(0) - k(0) = \big(\mathrm{e}^{\tau\widetilde{B}} - \partial_2 E_B(\tau, f_0)\big) A f_0.$$

Expanding both terms

$$\begin{aligned}\mathrm{e}^{\tau\widetilde{B}} &= I + \tau\widetilde{B} + \tau^2\widetilde{B}^2\varphi_2(\tau\widetilde{B})\\ E_B(\tau, f_0) &= f_0 + \tau B f_0 + \tau^2 R_2(E_B(\cdot, f_0)),\end{aligned}$$

we get

$$\begin{aligned}g(0) - k(0) = &-\tau B'(Af_0)f_0 - \tau\big(B(f_0) - \widetilde{B}\big)Af_0\\ &+ \tau^2\left(\widetilde{B}^2\varphi_2(\tau\widetilde{B}) - R_2(\partial_2 E_B(\cdot, f_0))\right)Af_0.\end{aligned}$$

The first term is bounded by assumption (3.8) and the second term by assumption (3.10). The third term is bounded by assumption (3.11) with $i = 0$ and the fourth term by assumption (3.12) with $i = 0$.

Finally, we have to estimate the remainder term of the quadrature rule which is bounded by assumption (3.13).

Terms of order 2. For the functions

$$\begin{aligned}g(s, \sigma) &= \mathrm{e}^{(\tau-s)\widetilde{B}} A \mathrm{e}^{(s-\sigma)\widetilde{B}} A \mathrm{e}^{\sigma\widetilde{B}} f_0\\ k(s, \sigma) &= \partial_2 E_B(\tau - s, f(s)) A \partial_2 E_B(s - \sigma, f(\sigma)) A E_B(\sigma, f_0)\end{aligned}$$

we employ a quadrature rule (as in [14])

$$\begin{aligned}&\frac{\tau^2}{8}\Big(g(0,0) + 2g(\tau,0) + g(\tau,\tau)\Big) - \int_0^\tau\int_0^s k(s,\sigma)\,\mathrm{d}\sigma\mathrm{d}s\\ &\qquad = \frac{\tau^2}{8}\Big(g(0,0) + 2g(\tau,0) + g(\tau,\tau) - k(0,0) - 2k(\tau,0) - k(\tau,\tau)\Big) + d,\end{aligned}$$

where d is the remainder term. Consequently, we have to bound

$$\begin{aligned}g(\tau,\tau) - k(\tau,\tau) &= A^2\left(e^{\tau\widetilde{B}} f_0 - E_B(\tau, f_0)\right),\\ g(0,0) - k(0,0) &= \left(e^{\tau\widetilde{B}} - \partial_2 E_B(\tau, f_0)\right) A^2 f_0\\ &= \tau\left(\widetilde{B}\varphi_1(\tau\widetilde{B}) - R_1(\partial_2 E_B(\cdot, f_0))\right) A^2 f_0,\\ g(\tau,0) - k(\tau,0) &= A\left(\mathrm{e}^{\tau\widetilde{B}} - \partial_2 E_B(\tau, f_0)\right) A f_0\\ &= \tau A\left(\widetilde{B}\varphi_1(\tau\widetilde{B}) - R_1(\partial_2 E_B(\cdot, f_0))\right) A f_0.\end{aligned}$$

The first term can again be bounded by using assumption (3.9), now with $i = 2$. In addition, we can bound the second and third term using assumption (3.11) with $i = 1$ and $i = 2$ and assumption (3.12) with $i = 1$ and $i = 2$, respectively. Finally, the remainder term depends on the first partial derivatives of $k(s, \sigma)$ and can be bounded by (3.14) and (3.15).

Terms of order 3. In order to bound the remainder terms in the expansion of the exact solution as well as the approximate solution, we need assumption (3.16) and (3.17) respectively. ∎

4. Convergence analysis for the Vlasov–Poisson equations. We will consider the Vlasov–Poisson equations in 1+1 dimensions, i.e.

$$\begin{cases} \partial_t f(t,x,v) = -v\partial_x f(t,x,v) - \mathcal{E}(f(t,\cdot,\cdot),x)\partial_v f(t,x,v) \\ \partial_x \mathcal{E}(f(t,\cdot,\cdot),x) = \displaystyle\int_{\mathbb{R}} f(t,x,v)\,\mathrm{d}v - 1 \\ f(0,x,v) = f_0(x,v) \end{cases} \tag{4.1}$$

with periodic boundary conditions in space. For a function $g = g(x, v)$ the abstract differential operators A and B of the previous sections have thus the form

$$Ag(x,v) = -v\partial_x g(x,v), \qquad Bg(x,v) = -\mathcal{E}(g,x)\partial_v g(x,v).$$

The domain of interest is given by $(t, x, v) \in [0, T] \times [0, L] \times \mathbb{R}$. Thus, for all $x \in \mathbb{R}$

$$f(t,x,v) = f(t,x+L,v).$$

By equation (4.1) the electric field $\mathcal{E}$ is only determined up to a constant. This constant is chosen such that $\mathcal{E}$ has zero integral mean (electrostatic condition).

4.1. Definitions and notation. The purpose of this section is to introduce the notations and mathematical spaces necessary for giving existence, uniqueness, and regularity results as well as to conduct the estimates necessary for showing consistency and stability.

For the convergence proof we will use the Banach space $L^1([0, L] \times \mathbb{R})$ exclusively. This is reasonable as the solution f of (4.1) represents a probability density function. As such the L^1 norm is conserved for the exact (as well as the approximate) solution. Nevertheless, all the estimations could be done as well, for example, in $L^\infty([0, L] \times \mathbb{R})$.

However, we need some regularity of the solution. This can be seen from the assumptions of Theorem 3.6, where we have to apply a number of differential operators to the solution $f(t)$. Thus, we introduce the following spaces of continuously differentiable functions

$$\begin{aligned} \mathcal{C}^m_{\mathrm{per,c}} &= \left\{ g \in \mathcal{C}^m(\mathbb{R}^2, \mathbb{R}) \,;\, \forall x, v \colon (g(x+L,v) = g(x,v)) \wedge (\text{supp } g(x,\cdot) \text{ compact}) \right\}, \\ \mathcal{C}^m_{\mathrm{per}} &= \left\{ g \in \mathcal{C}^m(\mathbb{R}, \mathbb{R}) \,;\, \forall x \colon g(x+L) = g(x) \right\}. \end{aligned}$$

Together with the norm of uniform convergence of all derivatives up to order m, i.e.

$$\|g\|_{\mathcal{C}^m_{\mathrm{per,c}}} = \sum_{0 \le k+\ell \le m} \|\partial_x^k \partial_v^\ell g\|_\infty, \qquad \|g\|_{\mathcal{C}^m_{\mathrm{per}}} = \sum_{k=0}^{m} \|\partial_x^k g\|_\infty,$$

the spaces $\mathcal{C}^m_{\mathrm{per,c}}$ and $\mathcal{C}_{\mathrm{per}}$ are turned into Banach spaces.

We also have to consider spaces that involve time. To that end let us define

$$\mathcal{C}^m(0,T;C^m) = \left\{ f \in \mathcal{C}^m([0,T],C^0)\,;\, (f(t) \in C^m) \wedge (\sup_{t\in[0,T]} \|f(t)\|_{C^m} < \infty) \right\},$$

where C^m is taken as either $\mathcal{C}^m_{\mathrm{per,c}}$ or $\mathcal{C}^m_{\mathrm{per}}$. It should be noted that if it can be shown that the solution f of the Vlasov–Poisson equations lies in the space $\mathcal{C}^m(0,T;C^m)$, we can bound all derivatives (in space) up to order m uniformly in $t \in [0,T]$.

4.2. Existence, uniqueness, and regularity. In this section we recall the existence, uniqueness, and regularity results of the Vlasov–Poisson equations in 1+1 dimensions. The theorem is stated with a slightly different notation in [3] and [2].

THEOREM 4.1. *Assume that $f_0 \in \mathcal{C}^m_{\mathrm{per,c}}$ for some $m \in \mathbb{N} \cup \{0\}$ is non-negative, then $f \in \mathcal{C}^m(0,T;\mathcal{C}^m_{\mathrm{per,c}})$ and $\mathcal{E}(f(t,\cdot,\cdot),x)$ as a function of (t,x) lies in $\mathcal{C}^m(0,T;\mathcal{C}^m_{\mathrm{per}})$. In addition, we can find a number $Q(T) > 0$ such that for all $t \in [0,T]$ and $x \in \mathbb{R}$ it holds that* $\mathrm{supp} f(t,x,\cdot) \subset [-Q(T),Q(T)]$.

Proof. A proof can be found in [9]. ∎

We also need a regularity result for the electric field that does not directly result from a solution of the Vlasov–Poisson equations, but from some generic function g (e.g., computed from an application of a splitting operator to f_0).

COROLLARY 4.2. *For $g \in \mathcal{C}^m_{\mathrm{per,c}}$ it holds that $\mathcal{E}(g,\cdot) \in \mathcal{C}^m_{\mathrm{per}}$.*

Proof. The result follows from the proof of Theorem 4.1. In addition, in the 1+1 dimensional case it can also be shown easily by starting from the exact representation of the electromagnetic field that is given in (5.3) below. ∎

With the arguments contained in the proof of Theorem 4.1, the regularity results given can be extended to the differential equations generated by B and $\widetilde{B}$. Thus, Theorem 4.1 remains valid if $E_B(t,f_0)$ or $\mathrm{e}^{t\widetilde{B}} f_0$ is substituted for $f(t)$.

4.3. Consistency. The most challenging task in proving the assumptions of Theorem 3.6 is to control the derivative of E_B with respect to the initial value. The following lemma accomplishes that.

LEMMA 4.3. *The map*

$$\begin{aligned} \mathcal{C}^m_{\mathrm{per,c}} \times \mathcal{C}^\ell_{\mathrm{per,c}} &\to \mathcal{C}^{\min(m-1,\ell)}_{\mathrm{per,c}} \\ (u_0, g) &\mapsto \partial_2 E_B(t,u_0) g, \end{aligned}$$

is well-defined, i.e. $\partial_2 E_B(t,u_0)g$ lies in the codomain given for $u_0 \in \mathcal{C}^m_{\mathrm{per,c}}$ and $g \in \mathcal{C}^\ell_{\mathrm{per,c}}$.

Proof. We consider $u'(t) = Bu(t)$ with $u(0) = u_0$. Motivated by the method of characteristics we can write

$$\begin{aligned} \partial_t V_{u_0}(t,x,v) &= -\mathcal{E}(u(t,\cdot,\cdot),x) \\ V_{u_0}(0,x,v) &= v \\ u(t,x,v) &= u_0(x, V_{u_0}(t,x,v)), \end{aligned}$$

where $V_{u_0}(t,x,v)$ is given implicitly as the unique solution of the equations stated above. To show that V_{u_0} depends affinely on the initial value u_0, let us integrate $u'(t) = Bu(t)$ with respect to the velocity; this gives at once

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{-\infty}^{\infty} u(t)\,\mathrm{d}v = -\mathcal{E}(u(t,\cdot,\cdot),x) \int_{-\infty}^{\infty} \partial_v u(t)\,\mathrm{d}v$$

which using integration by parts and the fact that $u(t)$ has compact support (in the velocity direction) shows that the time derivative on the left hand side vanishes. Therefore,

$$\mathcal{E}(u(t,\cdot,\cdot),x) = \mathcal{E}(u_0,x),$$

which implies that V_{u_0} depends affinely on the initial value u_0.

Computing the Gâteaux derivative with respect to the direction g we get

$$\begin{aligned}\partial_h E_B(t,u_0+hg)(x,v)|_{h=0} = &\left(\partial_2 u_0\right)\left(x, V_{u_0}(t)(x,v)\right)\left(V_g(t)(x,v)-v\right)\\ &+ g(x,V_{u_0}(t)(x,v)),\end{aligned}$$

since V is affine with respect to the initial value. From this representation the result follows. □

The following two lemmas present time derivatives up to order two of Bf, $\widetilde{B}f$ and $E_B(t,f_0)$ which follow from a simple calculation. Let us start with the derivatives of the operator B and $\widetilde{B}$ applied to the exact solution $f(t) = f(t,\cdot,\cdot)$.

Lemma 4.4. *For f sufficiently often continuously differentiable, we have*

$$\begin{aligned}\partial_t Bf(t,x,v) &= -\mathcal{E}\left(f'(t),x\right)\partial_v f(t,x,v) - \mathcal{E}(f(t),x)\partial_{vt} f(t,x,v)\\ \partial_t^2 Bf(t,x,v) &= -\mathcal{E}\left(f''(t),x\right)\partial_v f(t,x,v)\\ &\quad - 2\mathcal{E}\left(f'(t),x\right)\partial_{vt} f(t,x,v) - \mathcal{E}(f(t),x)\partial_{vtt} f(t,x,v)\end{aligned}$$

and

$$\begin{aligned}\partial_t \widetilde{B}f(t,x,v) &= -\mathcal{E}\left(f'(t+\tfrac{\tau}{2}),x\right)\partial_v f(t,x,v) - \mathcal{E}(f(t+\tfrac{\tau}{2}),x)\partial_{vt} f(t,x,v)\\ \partial_t^2 \widetilde{B}f(t,x,v) &= -\mathcal{E}\left(f''(t+\tfrac{\tau}{2}),x\right)\partial_v f(t,x,v)\\ &\quad - 2\mathcal{E}\left(f'(t+\tfrac{\tau}{2}),x\right)\partial_{vt} f(t,x,v) - \mathcal{E}(f(t+\tfrac{\tau}{2}),x)\partial_{vtt} f(t,x,v)\end{aligned}$$

Proof. From the relations $Bf(t,x,v) = -\mathcal{E}(f(t),x)\partial_v f(t,x,v))$ and $\widetilde{B}f(t,x,v) = -\mathcal{E}(f(t+\frac{\tau}{2},x)\partial_v f(t,x,v)$ the result follows by the product rule. □

Further, we have to compute some derivatives of the evolution operator $E_B(t,f_0)$ with respect to time.

Lemma 4.5. *For f sufficiently often continuously differentiable, we have*

$$\begin{aligned}\partial_t E_B(t,f_0) &= BE_B(t,f_0)\\ &= -\mathcal{E}(E_B(t,f_0),\cdot)\partial_v E_B(t,f_0)\\ \partial_t^2 E_B(t,f_0) &= -\mathcal{E}(E_B(t,f_0),\cdot)\partial_v(BE_B(t,f_0)) - \mathcal{E}(BE_B(t,f_0),\cdot)\partial_v E_B(t,f_0)\\ \partial_t(\partial_2 E_B(t,f_0)) &= -\mathcal{E}(E_B(t,f_0),\cdot)\partial_v(\partial_2 E_B(t,f_0)) - \mathcal{E}(\partial_2 E_B(t,f_0),\cdot)\partial_v E_B(t,f_0)\\ \partial_t^2(\partial_2 E_B(t,f_0)) &= -\mathcal{E}(BE_B(t,f_0),\cdot)\partial_v(\partial_2 E_B(t,f_0))\\ &\quad - \mathcal{E}(E_B(t,f_0),\cdot)\partial_v(\partial_t(\partial_2 E_B(t,f_0)))\\ &\quad - \mathcal{E}(\partial_t(\partial_2 E_B(t,f_0)),\cdot)\partial_v E_B(t,f_0)\\ &\quad - \mathcal{E}(\partial_2 E_B(t,f_0),\cdot)\partial_v BE_B(t,f_0).\end{aligned}$$

Proof. From the relation $Bf(t,x,v) = -\mathcal{E}(f(t),x)\partial_v f(t,x,v))$ the result follows by a simple calculation. □

It is also necessary to investigate the behavior of the φ functions introduced in Definition 2.1.

LEMMA 4.6. *For the Vlasov–Poisson equations the functions $\varphi_i(\tau E)$ with $E \in \{A, \widetilde{B}\}$ are maps from $\mathcal{C}^m_{\mathrm{per,c}}$ to $\mathcal{C}^m_{\mathrm{per,c}}$ for all $\tau \geq 0$ and $i \in \mathbb{N}$.*

Proof. For $i = 0$ we have

$$\mathrm{e}^{-\tau v \partial_x} f_0(x, v) = f_0(x - \tau v, v),$$

and

$$\mathrm{e}^{-\tau \mathcal{E}\left(f(\frac{\tau}{2}), x\right)\partial_v} f_0(x, v) = f_0\left(x, v - \tau \mathcal{E}(f(\tfrac{\tau}{2}), x)\right).$$

This clearly doesn't change the differentiability properties.

For the φ functions the desired result follows at once from the representation given in (2.5). ∎

Now we are able to show that all the assumptions of Theorem 3.6 are fulfilled and that we thus have consistency of order 2. This is the content of the following theorem.

THEOREM 4.7. *Suppose that $f_0 \in \mathcal{C}^3_{\mathrm{per,c}}$ is non-negative. Then the Strang splitting scheme* (2.3) *for the Vlasov–Poisson equations is consistent of order 2 in the norm of $L^1([0, L] \times \mathbb{R})$.*

Proof. The proof proceeds by noting that the solution has compact support (for a finite time interval), i.e., we can estimate v by some constant Q. On the other hand it is clear that for $f_0 \in \mathcal{C}^{m+1}_{\mathrm{per,c}}$ we get $Af_0 \in \mathcal{C}^m_{\mathrm{per,c}}$ and $\widetilde{B} f_0 \in \mathcal{C}^m_{\mathrm{per,c}}$. The same is true for B as can be seen by Corollary 4.2. Therefore, we can establish the bounds (3.6), (3.7), and (3.8). Noting that, by Lemma 4.4, terms of the form $R_i(\partial_2 E_B)$ are mappings from $\mathcal{C}^{m+i}_{\mathrm{per,c}}$ to $\mathcal{C}^m_{\mathrm{per,c}}$ and that, by Lemma 4.6, the φ functions are mappings from $\mathcal{C}^m_{\mathrm{per,c}}$ to $\mathcal{C}^m_{\mathrm{per,c}}$ we can conclude that after applying all operators in assumptions (3.9), (3.10), (3.11), and (3.12) we get a continuous function. By the regularity results we can bound these functions uniformly in time. The same argument also shows the validity of the bound in assumption (3.17).

Finally, with the help of Lemmas 4.3 and 4.5 together with the above observations we can verify the bounds in assumptions (3.13), (3.14), (3.15), and (3.16). ∎

4.4. Stability. We have to verify that the Strang splitting scheme (2.3) satisfies the conditions (3.3) and (3.4). The stability bound (3.3) is obviously fulfilled since

$$\left\|\mathrm{e}^{\frac{\tau}{2}A}\mathrm{e}^{\tau B_{k+1/2}}\mathrm{e}^{\frac{\tau}{2}A} f(t)\right\|_1 \leq \|f(t)\|_1.$$

This follows from the proof of Lemma 4.6 as the above operators can be represented as translations only (note that a translation does not change the L^1 norm).

To verify (3.4), which can be seen as a substitute for non-linear stability, it remains to be shown that

$$\|g\big(x, v - \tau\mathcal{E}(f_{k+1/2}, x)\big) - g\big(x, v - \tau\mathcal{E}(f(t_k + \tfrac{\tau}{2}), x))\big)\|_1 \leq \tau \|f_{k+1/2} - f(t_k + \tfrac{\tau}{2})\|_1$$

for $g(x, v) = \mathrm{e}^{\frac{\tau}{2}A} f(t_k, x, v) = f(t_k, x - \frac{\tau}{2} v, v)$. This follows at once from the Lipschitz continuity of $\mathrm{e}^{\frac{\tau}{2}A} f$ and the explicit form of $\mathcal{E}$ in (5.3) below.

4.5. Convergence. We are now in the position to prove second-order convergence of Strang splitting for the Vlasov–Poisson equations in L^1. The same result holds literally in L^∞ (or any other L^p space).

THEOREM 4.8. *Suppose that $f_0 \in \mathcal{C}^3_{\mathrm{per,c}}$ is non-negative and that the auxiliary method* (2.2) *is first-order consistent and (locally) Lipschitz continuous with respect*

to its second argument. Then Strang splitting for the Vlasov–Poisson equations is second-order convergent in the norm of $L^1([0, L] \times \mathbb{R})$.

Proof. The result follows from Theorem 4.7, the bounds given in section 4.4 and Theorem 3.2. ∎

Note that the two auxiliary methods (5.1) and (5.2) below are indeed first-order consistent. If they are employed for the computation of $f_{k+1/2}$, the resulting Strang splitting is second-order convergent.

5. Numerical experiments. In this section we present some numerical experiments. Even if we neglect space discretization for the moment, we still have to settle the choice of $f_{k+1/2}$ which has to be a first-order approximation to $f(t_k + \frac{\tau}{2})$. This can be achieved by Taylor series expansion, interpolation of previously computed values, or by making an additional Lie–Trotter time step of length $\tau/2$. Since we are interested in time integration only, we choose the latter. This method is trivial to implement (once the Strang splitting scheme is implemented) and doesn't suffer from the numerical differentiation problems of a Taylor expansion. Thus, one possible choice would be to use

$$f_{k+1/2} = \mathrm{e}^{\frac{\tau}{2} B(f_k)} \mathrm{e}^{\frac{\tau}{2} A} f_k \tag{5.1}$$

in our simulations. That this is indeed a first-order approximation follows in the same way as our convergence proof for Strang splitting. We omit the details.

However, since the semigroup generated by $B(f_k)$ can be represented as a translation in velocity (see the proof of Lemma 4.6) and the electric field depends only on the average of the density function with respect to velocity, i.e. it holds that

$$\int_{\mathbb{R}} \mathrm{e}^{\frac{\tau}{2} B(f_k)} \mathrm{e}^{\frac{\tau}{2} A} f_k \, \mathrm{d}v = \int_{\mathbb{R}} \mathrm{e}^{\frac{\tau}{2} A} f_k \, \mathrm{d}v,$$

it is possible to drop the first factor in (5.1) without affecting the resulting electric field. Consequently, our choice is

$$f_{k+1/2} = \mathrm{e}^{\frac{\tau}{2} A} f_k. \tag{5.2}$$

Note that this is not a first-order approximation to $f(k + \frac{\tau}{2})$; however, the electric field, which is exclusively used in the subsequent steps of the algorithm, is equal to the electric field computed with the help of the first-order scheme given in (5.1).

Since the computation of (5.2) is the first step in the Strang splitting algorithm, this leads to a computationally efficient scheme. This scheme is also employed in [16], for example. However, no argument why second-order accuracy is retained is given there.

To compute the electric field we will use the following formula (see e.g. [3])

$$\begin{aligned} \mathcal{E}(f(t, \cdot, \cdot), x) &= \int_0^L K(x, y) \left(\int_{\mathbb{R}} f(t, y, v) \mathrm{d}v - 1 \right) \mathrm{d}y, \\ K(x, y) &= \begin{cases} \frac{y}{L} - 1 & 0 \leq x < y, \\ \frac{y}{L} & y < x \leq L. \end{cases} \end{aligned} \tag{5.3}$$

For space discretization we will employ a discontinuous Galerkin method (based on the description given in [16]). The approximation is of second-order with 80 cells in both the space and velocity direction. In [16] the coefficients for discretizations up

to order 2 are given. However, it is not difficult to employ a computer program to compute the coefficients for methods of arbitrary order.

Note that it is unnecessary to impose boundary conditions in the velocity direction. This is due to the fact that for a function f_0 with compact support in the velocity direction the solution will continue to have compact support for all finite time intervals $[0, T]$ (see Theorem 4.1 above). Periodic boundary conditions in space will be employed in all simulations conducted in this section.

5.1. Landau damping. The Vlasov–Poisson equations in 1+1 dimensions together with the initial value

$$f_0(x, v) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-v^2/2} \left(1 + \alpha \cos(0.5x)\right),$$

is called Landau damping. For $\alpha = 0.01$ the problem is called linear or weak Landau damping and for $\alpha = 0.5$ it is referred to as strong or non-linear Landau damping. As can be seen, for example, in [8, 6] and [18] Landau damping is a popular test problem for Vlasov codes. We solve this problem on the domain $(t, x, v) \in [0, 1] \times [0, 4\pi] \times [-6, 6]$.

For comparison we display the error of the Strang splitting algorithm together with the error for first-order Lie–Trotter splitting. Since we are mainly interested in the time integration error and there is no analytical solution of the problem available, we compare the error for different step sizes with a reference solution computed with $\tau = 3.9 \cdot 10^{-3}$. The correctness of our code was verified with an upwind scheme on a fine grid with up to 2560 grid points in the x- and v-direction, respectively. For this experiment, the time step size was determined by the CFL condition to be approximately $\tau = 6 \cdot 10^{-4}$. The error is computed in the discrete L^1 norm at time $t = 1$. The results given in Figure 5.1 are in line with the theoretical convergence results derived in this paper.

6. Conclusion. In this paper sufficient conditions are given that guarantee convergence of order 2 for the Strang splitting algorithm in the case of Vlasov-type equations. It is also shown that the Vlasov–Poisson equations in 1+1 dimensions is an example of a Vlasov-type equation, i.e., they fit into the framework of the analysis conducted. For the simulation on a computer, however, a further approximation has to be made (i.e., some sort of space discretization has to be introduced). This approximation is not included in the analysis done here. Nevertheless, the numerical experiments suggest that second-order convergence is retained in the fully discretized case as well.

Acknowledgments. The authors thank the referees for providing numerous suggestions that helped to improve the presentation of this paper.

REFERENCES

[1] E.A. Belli, *Studies of numerical algorithms for gyrokinetics and the effects of shaping on plasma turbulence*, PhD thesis, Princeton University, 2006.

[2] N. Besse, *Convergence of a semi-Lagrangian scheme for the one-dimensional Vlasov-Poisson system*, SIAM J. Numer. Anal., 42 (2005), pp. 350–382.

[3] ———, *Convergence of a high-order semi-Lagrangian scheme with propagation of gradients for the one-dimensional Vlasov-Poisson system*, SIAM J. Numer. Anal., 46 (2008), pp. 639–670.

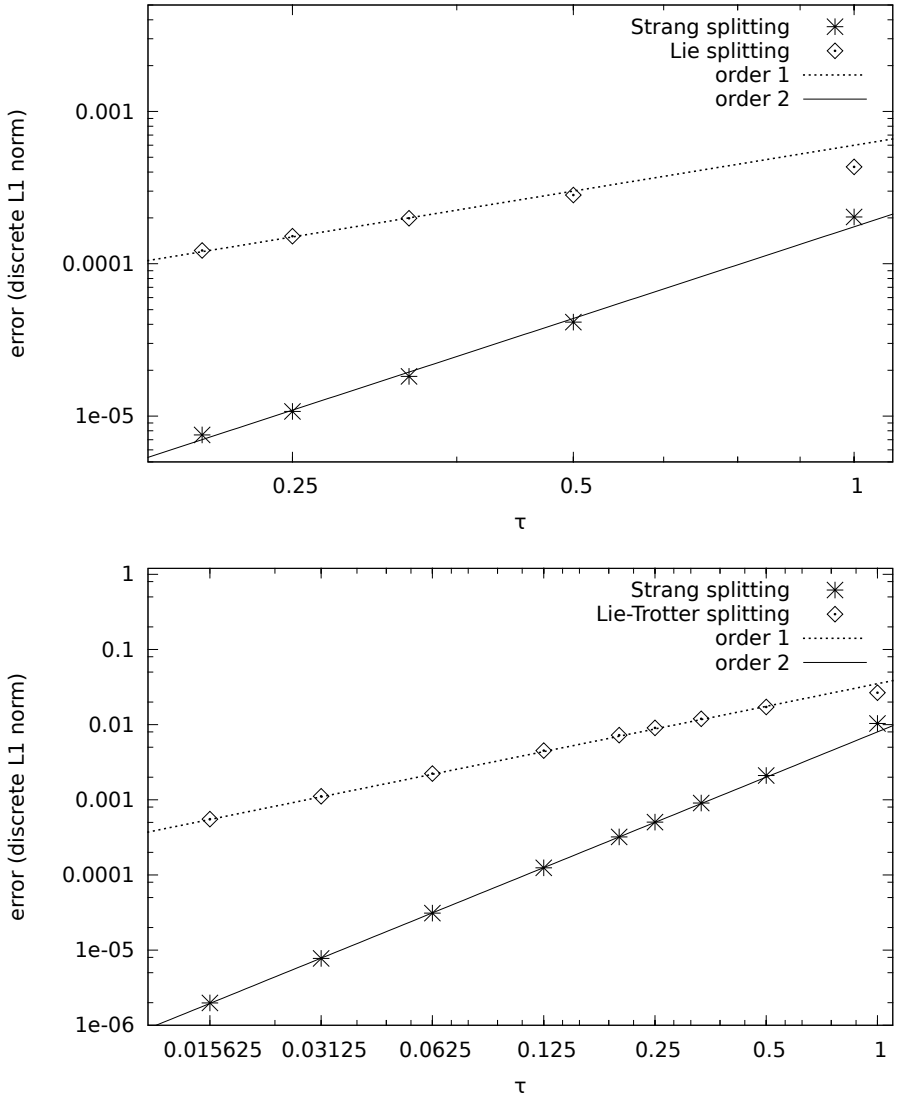

FIG. 5.1. *Error of the particle density function $f(1, \cdot, \cdot)$ for Strang and Lie–Trotter splitting respectively, where $\alpha = 0.01$ (top) and $\alpha = 0.5$ (bottom).*

[4] M. BOSTAN AND N. CROUSEILLES, *Convergence of a semi-Lagrangian scheme for the reduced Vlasov-Maxwell system for laser-plasma interaction*, Numer. Math., 112 (2009), pp. 169–195.

[5] C.Z. CHENG AND G. KNORR, *The integration of the Vlasov equation in configuration space*, J. Comput. Phys., 22 (1976), pp. 330–351.

[6] N. CROUSEILLES, E. FAOU, AND M. MEHRENBERGER, *High order Runge-Kutta-Nyström splitting methods for the Vlasov-Poisson equation.* http://hal.inria.fr/inria-00633934/PDF/cfm.pdf.

[7] M.R. FAHEY AND J. CANDY, *GYRO: A 5-d gyrokinetic-Maxwell solver*, Proceedings of the ACM/IEEE SC2004 Conference, (2008), p. 26.

[8] F. FILBET AND E. SONNENDRÜCKER, *Comparison of Eulerian Vlasov solvers*, Computer Physics

Communications, 150 (2003), pp. 247–266.
[9] R. T. Glassey, *The Cauchy Problem in Kinetic Theory*, SIAM, 1996.
[10] T.S. Hahm, L. Wang, and J. Madsen, *Fully electromagnetic nonlinear gyrokinetic equations for tokamak edge turbulence*, Physics of Plasmas, 16 (2009), p. 022305.
[11] E. Hansen and A. Ostermann, *Dimension splitting for evolution equations*, Numer. Math., 108 (2008), pp. 557–570.
[12] R.E. Heath, I.M. Gamba, P.J. Morrison, and C. Michler, *A discontinuous Galerkin method for the Vlasov-Poisson system*, J. Comput. Phys., 231 (2012), pp. 1140–1174.
[13] H. Holden, C. Lubich, and N.H. Risebro, *Operator splitting for partial differential equations with Burgers nonlinearity*. To appear in Math. Comp.
[14] T. Jahnke and C. Lubich, *Error bounds for exponential operator splittings*, BIT, 40 (2000), pp. 735–744.
[15] C. Lubich, *On splitting methods for Schrödinger-Poisson and cubic nonlinear Schrödinger equations*, Math. Comput., 77 (2008), pp. 2141–2153.
[16] A. Mangeney, F. Califano, C. Cavazzoni, and P. Travnicek, *A numerical scheme for the integration of the Vlasov-Maxwell system of equations*, J. Comput. Phys., 179 (2002), pp. 495–538.
[17] T. Respaud and E. Sonnendrücker, *Analysis of a new class of forward semi-Lagrangian schemes for the 1D Vlasov Poisson equations*, Numer. Math., 118 (2011), pp. 329–366.
[18] J.A. Rossmanith and D.C. Seal, *A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov-Poisson equations*, J. Comput. Phys., 230 (2011), pp. 6203–6232.

B Convergence analysis of a discontinuous Galerkin/Strang splitting approximation for the Vlasov–Poisson equations

Journal	SIAM Journal on Numerical Analysis
Authors	Lukas Einkemmer, Alexander Ostermann
submitted	12.11.2012

CONVERGENCE ANALYSIS OF A DISCONTINUOUS GALERKIN/STRANG SPLITTING APPROXIMATION FOR THE VLASOV–POISSON EQUATIONS

LUKAS EINKEMMER* AND ALEXANDER OSTERMANN*

Abstract. A rigorous convergence analysis of the Strang splitting algorithm with a discontinuous Galerkin approximation in space for the Vlasov–Poisson equations is provided. It is shown that under suitable assumptions the error is of order $\mathcal{O}\left(\tau^2 + h^q + h^q/\tau\right)$, where τ is the size of a time step, h is the cell size, and q the order of the discontinuous Galerkin approximation. In order to investigate the recurrence phenomena for approximations of higher order as well as to compare the algorithm with numerical results already available in the literature a number of numerical simulations are performed.

Key words. Strang splitting, discontinuous Galerkin approximation, convergence analysis, Vlasov–Poisson equations, recurrence

AMS subject classifications. 65M12, 82D10, 65L05, 65M60

1. Introduction. In astro- and plasma physics the behavior of a collisionless plasma is modeled by the Vlasov equation (see e.g. [2])

$$\partial_t f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(t, \boldsymbol{x}, \boldsymbol{v}) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f(t, \boldsymbol{x}, \boldsymbol{v}) = 0, \tag{1.1}$$

a kinetic model that in certain applications is also called the collisionless Boltzmann equation. It is posed in a $3+3$ dimensional phase space, where $\boldsymbol{x}$ denotes the position and $\boldsymbol{v}$ the velocity. The density function f is the sought-after particle distribution, and the (force) term $\boldsymbol{F}$ describes the interaction of the plasma with the electromagnetic field.

In this paper we will study the convergence properties of a full discretization of the so called Vlasov–Poisson equations, where the force term

$$\boldsymbol{F} = -\nabla_{\boldsymbol{x}} \phi$$

is the gradient of the self-consistent electric potential ϕ. This simplified model is used in various applications, e.g. in the context of Landau damping.

For the numerical solution of (1.1), various methods have been considered in the literature, for example particle methods and in particular the particle-in-cell method, see [12, 13, 16]. Another prevalent approach consists in employing splitting methods, first proposed in the context of the Vlasov–Poisson equations by [8] and later extended to the full Vlasov–Maxwell equations in [19]. Both papers use second-order Strang splitting. In the seminal paper [17], the convergence properties of Strang-splitting for evolution equations were analyzed with the help of the variation-of-constants formula. This approach was recently extended to Vlasov-type equations in [11]. In [3, 4, 24, 7] semi-Lagrangian methods are combined with Strang splitting. Convergence is shown in the case of the 1+1 dimensional Vlasov–Poisson equations, and in [6] for a special case of the one-dimensional Vlasov–Maxwell equation. In these papers usually Hermite or spline interpolation is employed.

*Department of Mathematics, University of Innsbruck, Technikerstraße 13, Innsbruck, Austria (lukas.einkemmer@uibk.ac.at, alexander.ostermann@uibk.ac.at). The first author was supported by a scholarship of the Vizerektorat für Forschung, University of Innsbruck, and by the Austrian Science Fund (FWF), project id: P25346.

On the other hand, discontinuous Galerkin approximations in space have been studied for the Vlasov–Poisson equations as well. In [15] and [16] the weak version of the Vlasov–Poisson equations is discretized by a discontinuous Galerkin scheme and integrated in time by Runge–Kutta methods. The convergence of such methods has also been studied (see e.g. [27]). In [25] a higher order semi-Lagrangian method in time is combined with a discontinuous Galerkin approximation in space. However, no convergence analysis is given. A direct Strang splitting scheme with a discontinuous Galerkin approximation is implemented in [19]. Since only a single value per cell is advanced in time this leads to a Van Leer scheme. The advantage of that method is that there is no memory overhead as compared to a finite difference implementation (note that the method under consideration in this paper stores the coefficients of the Legendre polynomials up to order ℓ which leads to an increased memory consumption, but is expected to result in a scheme that is less dissipative, see e.g. [20]). In the before mentioned paper a numerical study of the Van Leer scheme is conducted.

Our goal in this paper is to provide the missing convergence analysis for a high order discontinuous Galerkin approximation in space which is combined with the (direct) Strang splitting scheme. Since such an approximation does not result in a function that is continuous across cell boundaries, the methods which are employed to show convergence for Hermite or spline interpolation are not applicable. In this paper, we will show, using the abstract (time) convergence result in [11], that this scheme converges with order $\mathcal{O}\left(\tau^2 + h^q + h^q/\tau\right)$. Our main result is stated in Theorem 2.9 below. In addition, we will discuss some numerical aspects of the discontinuous Galerkin discretization in section 3.

2. Vlasov–Poisson equations in 1+1 dimensions. In this section we perform the convergence analysis of Strang splitting in time with a discontinuous Galerkin approximation in space for the Vlasov–Poisson equations in 1+1 dimensions. To that end we first describe the setting as well as give the necessary regularity results (sections 2.1 to 2.3). We then describe the time (section 2.4) and space discretization (sections 2.5 and 2.6). In section 2.7 we will extend a commonly employed approximation result from $\mathcal{C}^{\ell+1}$ functions to piecewise polynomials with a small jump discontinuity. This extension is crucial to show consistency (which is done in section 2.8). Finally, convergence is established in section 2.9.

2.1. Setting. We will consider the Vlasov–Poisson equations in 1+1 dimensions, i.e.

$$\begin{cases} \partial_t f(t,x,v) = -v\partial_x f(t,x,v) - E(f(t,\cdot,\cdot),x)\partial_v f(t,x,v) \\ \partial_x E(f(t,\cdot,\cdot),x) = \displaystyle\int_{\mathbb{R}} f(t,x,v)\,\mathrm{d}v - 1 \\ f(0,x,v) = f_0(x,v) \end{cases} \tag{2.1}$$

with periodic boundary conditions in space. The domain of interest is given by $(t,x,v) \in [0,T] \times [0,L] \times \mathbb{R}$. The periodic boundary conditions imply

$$\forall x \in \mathbb{R}\colon f(t,x,v) = f(t,x+L,v).$$

It is physically reasonable to assume at least an algebraic decay of f_0 in the velocity direction. Thus, we can approximate (to arbitrary precision) f_0 by an initial value with compact support. As will be apparent in the next section it is unnecessary to impose boundary conditions in the velocity direction for initial values with compact

support. This is due to the fact that for such an initial value the solution will continue to have compact support for all finite time intervals $[0,T]$ (see Theorem 2.1).

For most of this presentation it will be more convenient to work with the following abstract initial value problem

$$\begin{cases} \partial_t f(t) = (A+B)f(t) \\ f(0) = f_0, \end{cases} \tag{2.2}$$

where we assume that A is an (unbounded) linear operator. In addition, we assume that B can be written in the form $Bf = B(f)f$, where $B(f)$ is an (unbounded) linear operator. For the Vlasov–Poisson equations in 1+1 dimensions the obvious choice is $Af = -v\partial_x f$ and $Bf = -E(f(t,\cdot,\cdot),x)\partial_v f$.

In $1+1$ dimensions an explicit representation of the electric field is given by the following formula

$$\begin{aligned} E(f(t,\cdot,\cdot),x) &= \int_0^L K(x,y)\left(\int_{\mathbb{R}} f(t,y,v)\mathrm{d}v - 1\right)\mathrm{d}y, \\ K(x,y) &= \begin{cases} \frac{y}{L} & 0 < y < x, \\ \frac{y}{L} - 1 & x < y < L, \end{cases} \end{aligned} \tag{2.3}$$

where we have assumed that E is chosen to have zero integral mean (electrostatic condition) and the plasma is globally neutral, i.e.

$$\int_0^L \left(\int_{\mathbb{R}} f(t,y,v)\,\mathrm{d}v - 1\right)\,\mathrm{d}y = 0.$$

From the latter condition, we can deduce that $E(0) = E(L)$, i.e. the electric field is periodic in space (see, e.g. [5]). This representation allows us to get a simple estimate of the electric field in terms of the particle density f.

2.2. Definitions and notation. The purpose of this section is to introduce the notations and mathematical spaces necessary for giving existence, uniqueness, and regularity results as well as to conduct the estimates necessary for showing consistency and convergence.

We will use $\|\cdot\|$ to denote the infinity norm and $\|\cdot\|_p$ to denote the L^p norm on $[0,L]\times\mathbb{R}$. For estimating the errors in space and velocity we will use the Banach space $L^\infty([0,L]\times[-v_{\max},v_{\max}])$. Note that consistency bounds in the physically more reasonable L^1 norm are a direct consequence of the bounds we derive in the infinity norm. The situation is more involved in the case of stability (this is discussed in section 2.9).

For our convergence analysis we need some regularity of the solution. To that end, we introduce the following spaces of continuously differentiable functions

$$\begin{aligned} \mathcal{C}^m_{\mathrm{per,c}} &:= \left\{ g \in \mathcal{C}^m(\mathbb{R}^2,\mathbb{R}),\, \forall x,v\colon (g(x+L,v) = g(x,v)) \wedge (\operatorname{supp} g(x,\cdot)\ \text{compact}) \right\}, \\ \mathcal{C}^m_{\mathrm{per}} &:= \left\{ g \in \mathcal{C}^m(\mathbb{R},\mathbb{R}),\, \forall x\colon g(x+L) = g(x) \right\}. \end{aligned}$$

Equipped with the norm of uniform convergence of all derivatives up to order m, $\mathcal{C}^m_{\mathrm{per,c}}$ and $\mathcal{C}_{\mathrm{per}}$ are Banach spaces.

We also have to consider spaces that involve time. To that end let us define for any subspace $Z \subset C^m(\mathbb{R}^d,\mathbb{R})$ the space

$$\mathcal{C}^m(0,T;Z) := \left\{ g \in \mathcal{C}^m([0,T],C^0),\, (g(t)\in Z) \wedge \left(\sup_{t\in[0,T]} \|g(t)\|_Z < \infty\right) \right\}.$$

Below, we will either take the choice $Z = \mathcal{C}^m_{\text{per,c}}$ or $Z = \mathcal{C}^m_{\text{per}}$. It should be noted that functions in $\mathcal{C}^m(0,T;Z)$ possess spatial derivatives up to order m that are uniformly bounded in $t \in [0,T]$.

2.3. Existence, uniqueness, and regularity. In this section we recall the existence, uniqueness, and regularity results for the Vlasov–Poisson equations in 1+1 dimensions. The following theorem is stated with a slightly different notation in [4] and [3].

THEOREM 2.1. *Assume that $f_0 \in \mathcal{C}^m_{\text{per,c}}$ is non-negative, then $f \in \mathcal{C}^m(0,T;\mathcal{C}^m_{\text{per,c}})$ and $E(f(t,\cdot,\cdot),x)$ as a function of (t,x) lies in $\mathcal{C}^m(0,T;\mathcal{C}^m_{\text{per}})$. In addition, we can find a number $Q(T)$ such that for all $t \in [0,T]$ and $x \in \mathbb{R}$ it holds that* $\operatorname{supp} f(t,x,\cdot) \subset [-Q(T),Q(T)]$.

Proof. A proof can be found in [14, Chap. 4]. ∎

We also need a regularity result for the electric field that does not directly result from a solution of the Vlasov–Poisson equations, but from some generic function f (e.g., an f computed from an application of a splitting operator to f_0).

COROLLARY 2.2. *For $f \in \mathcal{C}^m_{\text{per,c}}$ it holds that $E(f,\cdot) \in \mathcal{C}^m_{\text{per}}$.*

Proof. The result follows from the proof of Theorem 2.1. In addition, in the 1+1 dimensional case it can also be followed from the exact representation of the electromagnetic field that is given in equation (2.3). ∎

It should also be noted that due to the proof of Theorem 2.1, the regularity results given can be extended to the differential equations generated by B and $B(g)$ (for any sufficiently regular g). Thus, Theorem 2.1 remains valid if $E_B(t,f_0)$ or $\mathrm{e}^{tB(g)}f_0$ is substituted for $f(t)$, where $E_B(t,f_0)$ denotes the solution of the differential equation $\partial_t g(t) = Bg(t)$ at time t with initial value $g(0) = f_0$.

2.4. Time discretization. We use Strang splitting for the time discretization of (2.2). This results in the scheme

$$f_{k+1} = S_k f_k, \tag{2.4a}$$

where f_k is the numerical approximation to $f(t)$ at time $t = k\tau$ with step size τ. The splitting operator S_k is the composition of three abstract operators

$$S_k = S^{(A)} S^{(B)}_k S^{(A)}, \tag{2.4b}$$

where

$$S^{(A)} = \mathrm{e}^{\frac{\tau}{2}A}, \qquad S^{(B)}_k = \mathrm{e}^{\tau B(f_{k+1/2})} \tag{2.4c}$$

with $f_{k+1/2} = \mathrm{e}^{\frac{\tau}{2}B(f_k)}\mathrm{e}^{\frac{\tau}{2}A} f_k$. The choice of $f_{k+1/2}$ is such as to retain second order in the non-linear case while still only advection problems have to be solved in the numerical approximation (for more details see e.g. [11]). Note that since $\mathrm{e}^{\frac{\tau}{2}B(f_k)}$ can be represented by a translation in the velocity direction only (which has no effect on the computation of the electric field) we can use here

$$f_{k+1/2} = S^{(A)} f_k. \tag{2.4d}$$

This is convenient as the computation of $S^{(A)}f_k$ incurs no performance overhead in the actual computation.

To conclude this section, let us emphasize that we interpret the evolutions $S^{(A)}$ and $S^{(B)}$ as shifts (i.e., we consider the differential equations generated by A and B in the weak sense).

2.5. Space discretization. We proceed in two steps. First, we introduce a cutoff in the velocity direction, i.e. we fix $v_{\max}$ and consider the problem on the domain $[0, L] \times [-v_{\max}, v_{\max}]$. Note that for an initial value with compact support with respect to velocity and a sufficiently large $v_{\max}$ this is still exact.

Second, we introduce a discontinuous Galerkin approximation in both the space and velocity direction. For simplicity, we consider a uniform rectangular grid. In this case, the cell boundaries are given by the following relations

$$\begin{aligned} x_i &= ih_x, \qquad 0 \le i \le N_x, \\ v_j &= jh_v - v_{\max}, \qquad 0 \le j \le N_v. \end{aligned}$$

Within each cell, i.e. a square $R_{ij} = [ih_x, (i+1)h_x] \times [jh_v - v_{\max}, (j+1)h_v - v_{\max}]$, $0 \le i < N_x$, $0 \le j < N_v$, we perform an orthogonal projection with respect to the basis of Legendre polynomials of degree at most ℓ in x and v. To be more precise, suppose that $g \in L^2([0, L] \times [-v_{\max}, v_{\max}])$; then the operator P is defined such that Pg restricted to R_{ij} for all i, j is the (unique) polynomial that results from the projection of g onto the $(\ell + 1)(\ell + 1)$ dimensional subspace generated by the (appropriately translated and scaled) Legendre polynomials up to degree ℓ. It is well known that this projection operator is given by

$$Pg|_{R_{ij}} = \sum_{k=0}^{\ell} \sum_{m=0}^{\ell} b_{km}^{ij} P_k^{(1)}(x) P_m^{(2)}(v) \tag{2.5a}$$

with coefficients

$$b_{km}^{ij} = \frac{(2k+1)(2m+1)}{h_x h_v} \int_{R_{ij}} P_m^{(1)}(x) P_k^{(2)}(v) g(x, v) \, \mathrm{d}(x, v). \tag{2.5b}$$

The translated and scaled Legendre polynomials are here defined as

$$P_l^{(1)}(\xi) = p_l \left(\frac{2(\xi - x_i)}{h_x} - 1 \right), \qquad P_l^{(2)}(\xi) = p_l \left(\frac{2(\xi - v_j)}{h_v} - 1 \right),$$

where p_l denote the Legendre polynomials with the standard normalization, i.e.

$$\int_{-1}^{1} p_l(y) p_j(y) \mathrm{d}y = \frac{2}{2l+1} \delta_{lj}.$$

It should be emphasized that the projection in a single cell is independent from the projection in any other cell. As this is not true for Hermite or spline interpolation it gives the discontinuous Galerkin scheme a computational advantage (see [19] and section 2.6).

Now we have to introduce an approximation to the abstract splitting operator (2.4b) that takes the space discretization into account. We use the decomposition

$$\tilde{S}_k = \tilde{S}^{(A)} \tilde{S}_k^{(B)} \tilde{S}^{(A)}, \tag{2.6a}$$

where

$$\tilde{S}^{(A)} = P S^{(A)}, \qquad \tilde{S}_k^{(B)} = P \mathrm{e}^{\tau B(\tilde{f}_{k+1/2})} \tag{2.6b}$$

with

$$\tilde{f}_{k+1/2} = \tilde{S}^{(A)} \tilde{f}_k. \tag{2.6c}$$

The fully discrete scheme then reads

$$\tilde{f}_{k+1} = \tilde{S}_k \tilde{f}_k, \qquad \tilde{f}_0 = P f_0. \tag{2.6d}$$

Note that $\tilde{f}_k$ represents the full approximation in time and space at time t_k.

2.6. Translation and projection. The principle algorithm has already been laid out in sections 2.4 and 2.5. However, the description given so far is certainly not sufficient as the straightforward implementation (first computing an exact solution and then projection onto a finite dimensional subspace) is clearly not a viable option. Thus, the purpose of this section is to describe in more detail the computation of

$$\tilde{S}^{(A)} f(x,v) = P \mathrm{e}^{\frac{\tau}{2} A} f(x,v) = P f\left(x - \frac{\tau}{2} v, v\right)$$

and

$$\tilde{S}_k^{(B)} f(x,v) = P \mathrm{e}^{\tau B(\tilde{f}_{k+1/2})} f(x,v) = P f\left(x, v - \tau E(\tilde{f}_{k+1/2}, x)\right).$$

Without loss of generality let us consider a translation of the form $f\left(x - \tau g(v), v\right)$. In addition, we fix the cell of interest as $[0,h] \times [0,h]$. Now we are primarily interested in an interval of length h and thus define $P_l(x) = p_l(\frac{2x}{h} - 1)$. Then we have

$$\int_0^h P_l(x) P_j(x) \mathrm{d}x = \frac{h}{2l+1} \delta_{lj}.$$

We have to first translate and then project a function that can be expanded as

$$f(x,v) = \sum_{m=0}^{M} \sum_{n=0}^{N} b_{mn} P_m(x) P_n(v)$$

onto the finite dimensional approximation space. Our goal is to compute the coefficients of $f(x - \tau g(v), v)$. These are given by

$$\begin{aligned} a_{lj} &= \frac{(2l+1)(2j+1)}{h^2} \int_0^h \int_0^h P_l(x) P_j(v) f(x - \tau g(v), v) \,\mathrm{d}x \,\mathrm{d}v \\ &= \frac{(2l+1)(2j+1)}{h} \sum_{m,n} b_{mn} \int_0^h P_j(v) P_n(v) \left(\frac{1}{h} \int_0^h P_l(x) P_m(x - \tau g(v)) \,\mathrm{d}x \right) \mathrm{d}v \\ &= \frac{(2l+1)(2j+1)}{h} \sum_{m,n} b_{mn} \int_0^h P_j(v) P_n(v) H_{lm}(g(v)\tau/h) \,\mathrm{d}v, \end{aligned} \tag{2.7}$$

where

$$H_{lm}(\delta) = \frac{1}{h} \int_0^h P_l(x) P_m(x - \delta h) \,\mathrm{d}x, \qquad \delta = \frac{g(v)\tau}{h}.$$

For a fixed v the function H_{lm} can be evaluated explicitly. This is done, up to order 3, in [19]. We will instead use a Mathematica program which can generate a representation of H_{lm} (up to arbitrary order) in `C` code that can then be embedded in the `C++` implementation. Note that it is sufficient to only evaluate H_{lm} for $0 < \delta < 1$ as the negative values of δ follow by a symmetry argument and integer multiplies

correspond to a shift of the cells only. Also, the computation of $H_{lm}(\delta)$ for $-1 < \delta < 1$ shows that only two terms from the sum in (2.7) do not vanish. That is, we need only the data from the same cell as well as a single neighboring cell (either the right or left neighbor) to compute an application of a splitting operator. This follows easily from the fact that the support of the Legendre basis functions are within a single cell only. More details are given in [19]. It remains to evaluate the remaining integral in equation (2.7). Since $g(v)$ is at most a polynomial of degree ℓ (in a single cell) we have to integrate a polynomial of degree at most ℓ^2. We use a Gauss–Legendre quadrature rule of appropriate order.

Note that in order to guarantee the stability of our scheme it is of vital importance that we can compute the exact result of the integral in equation (2.7). If only an approximation is used instabilities can occur (see section 3.4 and [21]).

To conclude this section, let us note that alternative strategies have been introduced in the literature (see, e.g. [23, 10]).

2.7. Polynomial approximation of functions with a small jump discontinuity. In this section our goal is to prove a bound concerning the approximation of piecewise polynomials of degree ℓ with a single jump discontinuity. For notational simplicity we will be concerned with a function of a single variable only; the general case is a simple tensor product of the situation described in this section. Thus, the operator P is here understood as the orthogonal projection with respect to the one-dimensional Legendre polynomials of degree less or equal to ℓ. The starting point of our investigation is the result in Theorem 2.3, which is applicable only if we can assume that g is $\ell+1$ times continuously differentiable. This assumption is not satisfied for the discontinuous Galerkin approximation considered in this paper. However, we will use the result as a stepping stone to prove a similar bound for the approximation of functions with a small jump discontinuity.

THEOREM 2.3. *Suppose that $g \in \mathcal{C}^{\ell+1}([0,h])$. Then*

$$\left\| g^{(k)} - (Pg)^{(k)} \right\| \leq C h^{\ell-k+1} \|g^{(\ell+1)}\|$$

for all $k \in \{0 \ldots, \ell\}$.

Proof. In [22, p. 59] it is shown that $Pg - g$ changes sign $\ell + 1$ times. From this, it follows that $(Pg)^{(k)} - g^{(k)}$ changes sign $\ell + 1 - k$ times. Therefore, $(Pg)^{(k)}$ is an interpolation polynomial of $g^{(k)}$ of degree $\ell + 1 - k$. Using the standard error representation for polynomial interpolation we get the desired result. ∎

For numerical methods that rely on a smooth approximation of the solution (for example, using Hermite or spline interpolation as in [5]) sufficient regularity in the initial condition implies the bound given in Theorem 2.3 for any approximation that has to be made in the course of the algorithm.

This assumption, however, is violated if we consider a discontinuous Galerkin approximation as, even if the initial condition is sufficiently smooth, the approximation will include a jump discontinuity at the cell boundary. Thus, we are interested in a bound that still gives us an equivalent result to that stated in Theorem 2.3 in the case of a function with a small jump discontinuity. The following theorem is thus the central result of this section. For simplicity, we consider a single cell only.

THEOREM 2.4. *Suppose that $g\colon [0,h] \to \mathbb{R}$ is piecewise polynomial of degree ℓ with a single discontinuity at $x_0 \in [0,h]$. In addition, we assume that the jump heights $\varepsilon^{(k)} = g^{(k)}(x_0+) - g^{(k)}(x_0-)$ satisfy $|\varepsilon^{(k)}| \leq c h^{\ell-k+1}$ for all $k \in \{0, \ldots, \ell\}$.*

Then,

$$\left\| g^{(k)} - (Pg)^{(k)} \right\| \leq C h^{\ell - k + 1},$$

for all $k \in \{0, \ldots, \ell\}$*. Note that the constant* C *only depends on* c*,* ℓ*, and the constant in Theorem 2.3.*

Proof. Let us assume that $x_0 \in (0, h)$ (otherwise the result is immediate). We smooth the piecewise constant function $g^{(\ell)}$ in the following way

$$p^{(\ell)}(x) = \frac{\varepsilon^{(\ell)}}{h} x + g^{(\ell)}(0). \tag{2.8}$$

Now, upon integration we get

$$p(x) = \frac{\varepsilon^{(\ell)}}{h} \frac{x^{\ell+1}}{(\ell+1)!} + \sum_{k=0}^{\ell} a_k x^k,$$

where we choose the coefficients in such a way that the Taylor polynomial of g expanded at 0 matches the first ℓ terms of g, i.e. $a_k = \frac{g^{(k)}(0)}{k!}$. This gives us the following representation

$$p(x) = \frac{\varepsilon^{(\ell)}}{h} \frac{x^{\ell+1}}{(\ell+1)!} + \sum_{k=0}^{\ell} \frac{g^{(k)}(0)}{k!} x^k.$$

Now let us consider the integral (for $x > x_0$)

$$\begin{aligned} \int_0^x p^{(m)}(y) - g^{(m)}(y) \, \mathrm{d}y &= p^{(m-1)}(x) - g^{(m-1)}(x) - p^{(m-1)}(0) + g^{(m-1)}(0) \\ &\quad + g^{(m-1)}(x_0+) - g^{(m-1)}(x_0-) \\ &= p^{(m-1)}(x) - g^{(m-1)}(x) + \varepsilon^{(m-1)}, \end{aligned}$$

where the last identity follows from the choice we made above. Now we know that (for $s_{\ell-1} > x_0$)

$$\int_0^{s_{\ell-1}} p^{(\ell)}(s_\ell) - g^{(\ell)}(s_\ell) \, \mathrm{d}s_\ell = p^{(\ell-1)}(s_{\ell-1}) - g^{(\ell-1)}(s_{\ell-1}) + \varepsilon^{(\ell-1)}$$

and further (for $s_{\ell-2} > x_0$)

$$\begin{aligned} \int_0^{s_{\ell-2}} p^{(\ell-1)}(s_{\ell-1}) - g^{(\ell-1)}(s_{\ell-1}) + \varepsilon^{(\ell-1)} \, \mathrm{d}s_{\ell-1} &= p^{(\ell-2)}(s_{\ell-2}) - g^{(\ell-2)}(s_{\ell-2}) \\ &\quad + \varepsilon^{(\ell-2)} + \varepsilon^{(\ell-1)} s_{\ell-2}. \end{aligned}$$

By an induction argument we can then estimate the approximation error as

$$\begin{aligned} |p(x) - g(x)| &\leq \left| \int_0^x \int_0^{s_1} \cdots \int_0^{s_{\ell-1}} p^{(\ell)}(s_\ell) - g^{(\ell)}(s_\ell) \, \mathrm{d}s_\ell \ldots \mathrm{d}s_2 \mathrm{d}s_1 \right| + \sum_{k=0}^{\ell-1} |\varepsilon^{(k)}| h^k \\ &\leq \int_0^x \int_0^{s_1} \cdots \int_0^{s_{\ell-1}} \left| p^{(\ell)}(s_\ell) - g^{(\ell)}(s_\ell) \right| \mathrm{d}s_\ell \ldots \mathrm{d}s_2 \mathrm{d}s_1 + c\ell h^{\ell+1} \\ &\leq C h^{\ell+1}. \end{aligned}$$

In addition we easily follow from equation (2.8) that

$$\|p^{(\ell+1)}\| \leq \frac{|\varepsilon^{(\ell)}|}{h} \leq c.$$

Now let us estimate the approximation error

$$\begin{aligned}\|Pg - g\| &= \|Pg - Pp + Pp - p + p - g\| \\ &\leq \|P(g-p)\| + \|Pp - p\| + \|p - g\| \\ &\leq Ch^{\ell+1},\end{aligned}$$

where in the last line we have used Theorem 2.3 and the well known fact that the projection operator P is a bounded operator in the infinity norm. The latter can be seen, for example, by estimating (2.5).

To get the corresponding result for the kth derivative we follow largely the same argument. The last estimate is then given by

$$\begin{aligned}\left\|(Pg)^{(k)} - g^{(k)}\right\| &\leq \left\|(P(g-p))^{(k)}\right\| + \left\|p^{(k)} - g^{(k)}\right\| + \left\|(Pp)^{(k)} - p^{(k)}\right\| \\ &\leq Ch^{-k}\|P(g-p)\| + Ch^{\ell-k+1} + Ch^{\ell-k+1}\left\|p^{(\ell+1)}\right\| \\ &\leq Ch^{\ell-k+1},\end{aligned}$$

where the estimate for the first term follows by the well-known Markov inequality (see e.g. [26]). □

Let us discuss the principle of applying Theorem 2.4. First the operator P is applied to $f(j\tau)$, i.e. a point on the exact solution, and we can assume the necessary regularity to apply Theorem 2.3. Consequently, we get a jump discontinuity of heights at most

$$|\varepsilon^{(k)}| \leq 2\left\|f^{(k)}(j\tau) - Pf^{(k)}(j\tau)\right\| \leq Ch^{\ell-k+1}\left\|f^{(\ell+1)}(j\tau)\right\| \leq Ch^{\ell-k+1}, \quad 0 \leq k \leq \ell.$$

Now the projected function is translated by a splitting operator (the example $g(x) = (Pf(j\tau))(x - v\tau)$ is illustrated in Figure 2.1) and projected back on the finite dimensional subspace. The resulting error up to the ℓth derivative is then given by (see Theorem 2.4)

$$\left\|g^{(k)} - (Pg)^{(k)}\right\| \leq Ch^{\ell-k+1}.$$

From this we can also follow that the new jump heights $\varepsilon_1^{(k)}$ are at most

$$|\varepsilon_1^{(k)}| \leq 2\|(Pg)^{(k)} - g^{(k)}\| \leq Ch^{\ell-k+1}, \qquad 0 \leq k \leq \ell.$$

Since we only have to repeat this procedure a finite number of times (i.e. for a single step of the Strang splitting algorithm) and the assumptions of Theorem 2.4 are satisfied uniformly for all $f(t)$, we can find a uniform constant C such that the desired estimate holds.

Strictly speaking this argument is only valid for a constant advection (i.e. where v is fixed). However, we can always decompose the projection operator as $P = P_v P_x$; that is, into a projection in the x-direction (that depends on the parameter v) and a subsequent projection in the v-direction. Due to the special form of the advection

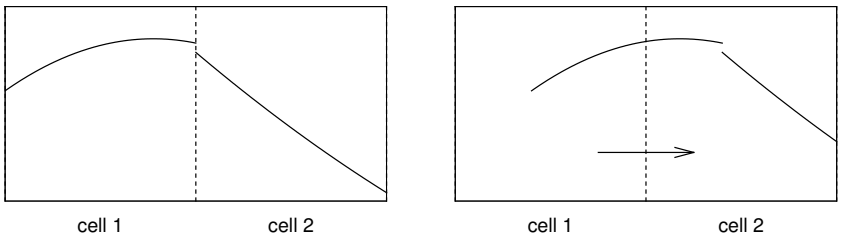

FIG. 2.1. *Projected smooth function with a jump discontinuity at the cell boundary (left) and translation with a discontinuity inside the cell (right). Only two cells in the x direction are shown.*

(see section 2.6), we have to consider a function that, restricted to a single cell, is discontinuous on a set which can be parametrized by a sufficiently often differentiable curve. Since, for any fixed v, the results in this section can be applied, we are left with a sufficiently often differentiable function (in both x and v). Therefore, the projection in the v-direction poses no difficulty (and can be conducted, for example, by Gauss–Legendre quadrature, as discussed in section 2.6).

2.8. Consistency. It is the purpose of this section to formulate assumptions under which we can show a consistency bound for the initial value problem given in equation (2.1). For notational simplicity, we will denote in this section the solution of (2.1) at a fixed time $t_k = k\tau$ by f_0. The function $\tilde{f}_0$ defined as Pf_0 is a (possible) initial value for a single time step (i.e., a single application of the splitting operator S_k or $\tilde{S}_k$). Since we consider consistency we are interested in the non-linear operator S that is given by

$$S(\cdot) = S^{(A)} \mathrm{e}^{\tau B\left(S^{(A)}(\cdot)\right)} S^{(A)}(\cdot),$$

and the corresponding spatially discretized operator

$$\tilde{S}(\cdot) = \tilde{S}^{(A)} P \mathrm{e}^{\tau B\left(\tilde{S}^{(A)}(\cdot)\right)} \tilde{S}^{(A)}(\cdot).$$

Let us first give a simple consequence of the variation-of-constants formula.

LEMMA 2.5. *Suppose that f_0 is absolutely continuous and that $f_{1/2}, \tilde{f}_{1/2}$ are integrable. Then*

$$S^{(B(f_{1/2}))} f_0 - S^{(B(\tilde{f}_{1/2}))} f_0 = \int_0^\tau \mathrm{e}^{(\tau-\sigma)B(\tilde{f}_{1/2})} \left(B(f_{1/2}) - B(\tilde{f}_{1/2})\right) \mathrm{e}^{\sigma B(f_{1/2})} f_0 \,\mathrm{d}\sigma.$$

Proof. Let $g(\tau) = S^{(B(f_{1/2}))} f_0$. Then

$$g' = B(f_{1/2})g = B(\tilde{f}_{1/2})g + \left(B(f_{1/2}) - B(\tilde{f}_{1/2})\right) g,$$

which can be rewritten by the variation-of-constants formulas as

$$S^{(B(f_{1/2}))} f_0 = S^{(B(\tilde{f}_{1/2}))} f_0 + \int_0^\tau \mathrm{e}^{(\tau-\sigma)B(\tilde{f}_{1/2})} \left(B(f_{1/2}) - B(\tilde{f}_{1/2})\right) g(\sigma) \,\mathrm{d}\sigma,$$

from which the desired result follows immediately. ∎

The next two lemmas will be the crucial step to prove consistency. First, we consider the error made by the (exact) splitting operators due to the space discretization. Note that the assumptions are exactly the same as those needed for the development in section 2.7 to hold.

LEMMA 2.6. *Suppose that* $f_0 \in \mathcal{C}^{\ell+1}_{\mathrm{per,c}}$. *Then*

$$\|Sf_0 - S\tilde{f}_0\| \leq C\left(\tau h^{\ell+1} + h^{\ell+1}\right),$$

where C *depends on* $\|f_0\|_{\mathcal{C}^{\ell+1}_{\mathrm{per,c}}}$ *(but not on* τ *and* h*).*

Proof. Let us define $\hat{f}_{1/2} = S^{(A)}\tilde{f}_0$. Then, we can write

$$\begin{aligned}
\|Sf_0 - S\tilde{f}_0\| &= \left\|S^{(A)}S^{(B(f_{1/2}))}S^{(A)}f_0 - S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}\tilde{f}_0\right\| \\
&\leq \left\|S^{(A)}\left(S^{(B(f_{1/2}))} - S^{(B(\hat{f}_{1/2}))}\right)S^{(A)}f_0\right\| + \left\|S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}(f_0 - \tilde{f}_0)\right\| \\
&\leq \left\|S^{(A)}\left(S^{(B(f_{1/2}))} - S^{(B(\hat{f}_{1/2}))}\right)S^{(A)}f_0\right\| + \|f_0 - \tilde{f}_0\|.
\end{aligned}$$

By using Lemma 2.5 and the definition of B we get

$$\begin{aligned}
&\left\|S^{(A)}\left(S^{(B(f_{1/2}))} - S^{(B(\hat{f}_{1/2}))}\right)S^{(A)}f_0\right\| \\
&\qquad = \left\|\int_0^\tau S^{(A)}\mathrm{e}^{(\tau-\sigma)B(\hat{f}_{1/2})}\left(B(f_{1/2}) - B(\hat{f}_{1/2})\right)\mathrm{e}^{\sigma B(f_{1/2})}S^{(A)}f_0\,\mathrm{d}\sigma\right\| \\
&\qquad \leq \tau\|E(f_{1/2}) - E(\hat{f}_{1/2})\| \max_{\sigma\in[0,\tau]}\left\|\partial_v\left(\mathrm{e}^{\sigma B(f_{1/2})}S^{(A)}f_0\right)\right\|.
\end{aligned}$$

Finally, since E is given by equation (2.3) it follows that

$$\|E(f_{1/2}) - E(\hat{f}_{1/2})\| \leq C\|f_{1/2} - \hat{f}_{1/2}\| \leq C\|f_0 - \tilde{f}_0\| \leq Ch^{\ell+1},$$

which concludes the proof. ∎

Second, we consider the error made due to the approximation of the (exact) splitting operators. Note that the assumptions are exactly the same as those needed for the development in section 2.7.

LEMMA 2.7. *Suppose that* $f_0 \in \mathcal{C}^{\ell+1}_{\mathrm{per,c}}$. *Then*

$$\|S\tilde{f}_0 - \tilde{S}\tilde{f}_0\| \leq C\left(\tau h^{\ell+1} + h^{\ell+1}\right),$$

where C *depends on* $\|f_0\|_{\mathcal{C}^{\ell+1}_{\mathrm{per,c}}}$ *(but not on* τ *and* h*).*

Proof. Let $\hat{f}_{1/2} = S^{(A)}\tilde{f}_0$ and $\tilde{f}_{1/2} = \tilde{S}^{(A)}\tilde{f}_0$. Then, we can write

$$\begin{aligned}
S\tilde{f}_0 - \tilde{S}\tilde{f}_0 &= S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}\tilde{f}_0 - \tilde{S}^{(A)}\tilde{S}^{\left(B(\tilde{f}_{1/2})\right)}\tilde{S}^{(A)}\tilde{f}_0 \\
&= \left(S^{(A)} - \tilde{S}^{(A)}\right)S^{(B(\hat{f}_{1/2}))}S^{(A)}\tilde{f}_0 && \text{(2.9a)} \\
&\quad + \tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))} - \tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}\tilde{f}_0 && \text{(2.9b)} \\
&\quad + \tilde{S}^{(A)}\tilde{S}^{(B(\tilde{f}_{1/2}))}\left(S^{(A)} - \tilde{S}^{(A)}\right)\tilde{f}_0. && \text{(2.9c)}
\end{aligned}$$

Now we estimate the three terms in (2.9) independently. For (2.9a) we get

$$\begin{aligned}
&\left\|\left(S^{(A)} - \tilde{S}^{(A)}\right)S^{(B(\hat{f}_{1/2}))}S^{(A)}\tilde{f}_0\right\| \\
&\quad = \left\|(P-1)\left(S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}\tilde{f}_0\right)\right\| \\
&\quad \leq \left\|(P-1)\left(S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}(P-1)f_0\right)\right\| + \left\|(P-1)\left(S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}f_0\right)\right\|.
\end{aligned}$$

Now for the first term in this expression we can employ classical results (Theorem 2.3) as f_0 is sufficiently often differentiable. This yields

$$\left\|(P-1)\left(S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}(P-1)f_0\right)\right\| \leq Ch^{\ell+1}.$$

To bound (2.9b), we write

$$\begin{aligned}(P-1)\left(S^{(A)}S^{(B(\hat{f}_{1/2}))}S^{(A)}f_0\right) &= (P-1)\left(S^{(A)}S^{(B(f_{1/2}))}S^{(A)}f_0\right)\\ &\quad + (P-1)\left(S^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-S^{(B(f_{1/2}))}\right)S^{(A)}f_0\right),\end{aligned}$$

where the first part can be bounded by a classical result. For the second part, we get (by employing Lemma 2.5)

$$\begin{aligned}&\left\|(P-1)\left(S^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-S^{(B(f_{1/2}))}\right)S^{(A)}f_0\right)\right\|\\ &\qquad\leq C\tau\|E(\hat{f}_{1/2})-E(f_{1/2})\|\max_{\sigma\in[0,\tau]}\left\|\partial_v\left(\mathrm{e}^{\sigma B(f_{1/2})}S^{(A)}f_0\right)\right\|,\end{aligned}$$

where the derivative is bounded as it is applied to a differentiable function.

With the help of Theorem 2.4, the term (2.9c) can be bounded by

$$\begin{aligned}\left\|\tilde{S}^{(A)}\tilde{S}^{(B(\tilde{f}_{1/2}))}\left(S^{(A)}-\tilde{S}^{(A)}\right)\tilde{f}_0\right\| &\leq C\left\|\left(S^{(A)}-\tilde{S}^{(A)}\right)\tilde{f}_0\right\|\\ &= C\left\|(1-P)S^{(A)}\tilde{f}_0\right\|\\ &\leq Ch^{\ell+1}.\end{aligned}$$

To estimate (2.9b) we write

$$\begin{aligned}\tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-\tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}\tilde{f}_0 &= \tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-\tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}(P-1)f_0\\ &\quad+\tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-\tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}f_0.\end{aligned}$$

The first part can be estimated by classical results. For the second part, we have

$$\begin{aligned}&\left\|\tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-\tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}f_0\right\|\\ &\quad\leq\left\|\tilde{S}^{(A)}\left(S^{(B(\hat{f}_{1/2}))}-S^{(B(f_{1/2}))}\right)S^{(A)}f_0\right\|\\ &\qquad+\left\|\tilde{S}^{(A)}\left(\tilde{S}^{(B(f_{1/2}))}-\tilde{S}^{(B(\tilde{f}_{1/2}))}\right)S^{(A)}f_0\right\|\\ &\qquad+\left\|\tilde{S}^{(A)}(P-1)S^{(B(f_{1/2}))}S^{(A)}f_0\right\|\\ &\quad\leq C\tau\|E(\hat{f}_{1/2})-E(f_{1/2})\|\max_{\sigma\in[0,\tau]}\left\|\partial_v\left(\mathrm{e}^{\sigma B(f_{1/2})}S^{(A)}f_0\right)\right\|\\ &\qquad+C\tau\|E(\tilde{f}_{1/2})-E(f_{1/2})\|\max_{\sigma\in[0,\tau]}\left\|\partial_v\left(\mathrm{e}^{\sigma B(f_{1/2})}S^{(A)}f_0\right)\right\|+Ch^{\ell+1},\end{aligned}$$

which is again a consequence of Lemma 2.5.

As in the last lemma E is given by equation (2.3) and thus it follows that

$$\|E(\hat{f}_{1/2})-E(f_{1/2})\|\leq C\|\hat{f}_{1/2}-f_{1/2}\|=C\|S^{(A)}(P-1)f_0\|\leq Ch^{\ell+1}$$

and

$$\|E(\tilde{f}_{1/2}) - E(f_{1/2})\| \leq C\|\tilde{f}_{1/2} - f_{1/2}\| \leq C(\|(P-1)S^{(A)}\tilde{f}_0\| + \|(P-1)f_0\|) \leq Ch^{\ell+1},$$

which concludes the proof. ∎

THEOREM 2.8 (Consistency). *Suppose that* $f_0 \in \mathcal{C}^{\max\{\ell+1,3\}}_{\text{per,c}}$. *Then*

$$\|Pf(h) - \tilde{S}\tilde{f}_0\| \leq C\left(\tau^3 + \tau h^{\ell+1} + h^{\ell+1}\right),$$

where C *depends on* $\|f_0\|_{\mathcal{C}^{\max\{\ell+1,3\}}_{\text{per,c}}}$ *(but not on* τ *and* h*).*

Proof. We write

$$\begin{aligned}\|Pf(h) - \tilde{S}\tilde{f}_0\| &= \|Pf(h) - PSf_0 + PSf_0 - Sf_0 + Sf_0 - S\tilde{f}_0 + S\tilde{f}_0 - \tilde{S}\tilde{f}_0\| \\ &\leq \|P\left(f(h) - Sf_0\right)\| + \|PSf_0 - Sf_0\| + \|Sf_0 - S\tilde{f}_0\| + \|S\tilde{f}_0 - \tilde{S}\tilde{f}_0\| \\ &\leq C\tau^3 + Ch^{\ell+1} + \|Sf_0 - S\tilde{f}_0\| + \|S\tilde{f}_0 - \tilde{S}\tilde{f}_0\|,\end{aligned}$$

where the first term was bounded by Theorem 4.9 in [11]. The two remaining terms can be bounded by Lemmas 2.6 and 2.7 to give the desired estimate. ∎

2.9. Convergence. To show consistency it was most convenient to bound all terms in the infinity norm. Bounds in the L^1 or L^2 norms then follow since we consider a compact domain in space and velocity. However, for stability (and thus convergence) we need to bound the operator norm of the projection operator P by 1. Since such a bound is readily available in the L^2 norm (as an orthogonal projection is always non-expansive in the corresponding norm) we will use it to show convergence. Note that this is not a peculiarity of our discontinuous Galerkin scheme. For example, in [5] stability for two schemes based respectively on spline and Lagrange interpolation is shown in the L^2 norm only.

THEOREM 2.9 (Convergence). *For the numerical solution of* (2.1) *we employ the scheme* (2.6). *Suppose that the initial value* $f_0 \in \mathcal{C}^{\max\{\ell+1,3\}}$ *is non-negative and compactly supported in velocity. Then, the global error satisfies the bound*

$$\sup_{0\leq n\leq N}\left\|\left(\prod_{k=0}^{n-1}\tilde{S}_k\right)\tilde{f}_0 - f(n\tau)\right\|_2 \leq C\left(\tau^2 + \frac{h^{\ell+1}}{\tau} + h^{\ell+1}\right),$$

where C *depends on* T *but is independent of* τ, h, n *for* $0 \leq n\tau \leq N\tau = T$.

Proof. From (2.6) we get

$$\tilde{f}_{n+1/2} = \tilde{S}^{(A)}\left(\prod_{m=0}^{n-1}\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(\tilde{f}_{m+1/2})}\tilde{S}^{(A)}\right)\tilde{f}_0.$$

Now we can derive a recursion for the error in the L^2 norm

$$\begin{aligned}e_{n+1} &= \|\tilde{f}_{n+1} - f(n\tau+\tau)\|_2 \\ &= \|\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(\tilde{f}_{n+1/2})}\tilde{S}^{(A)}\tilde{f}_n - f(n\tau+\tau)\|_2 \\ &\leq \|\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(\tilde{f}_{n+1/2})}\tilde{S}^{(A)}\tilde{f}_n - \tilde{S}^{(A)}\mathrm{Pe}^{\tau B(P\mathrm{e}^{\frac{\tau}{2}A}Pf(n\tau))}\tilde{S}^{(A)}\tilde{f}_n\|_2 \\ &\quad + \|\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(P\mathrm{e}^{\frac{\tau}{2}A}Pf(n\tau))}\tilde{S}^{(A)}(\tilde{f}_n - f(n\tau))\|_2 \\ &\quad + \|\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(P\mathrm{e}^{\frac{\tau}{2}A}Pf(n\tau))}\tilde{S}^{(A)}(1-P)f(n\tau)\|_2 \\ &\quad + \|\tilde{S}^{(A)}\mathrm{Pe}^{\tau B(P\mathrm{e}^{\frac{\tau}{2}A}Pf(n\tau))}\tilde{S}^{(A)}Pf(n\tau) - Pf(n\tau+\tau)\|_2 \\ &\quad + \|(P-1)f(n\tau+\tau)\|_2.\end{aligned}$$

Since $\tilde{f}_n$ is piecewise polynomial and thus absolutely continuous almost everywhere, we can estimate the first term with the help of Lemma 2.5. The fourth term can be estimated by using Theorem 2.8. This gives us

$$\begin{aligned} e_{n+1} &\leq C\tau \|\tilde{f}_{n+1/2} - P\mathrm{e}^{\frac{\tau}{2}A} P f(n\tau)\|_2 + \|\tilde{f}_n - f(n\tau)\|_2 + C\left(\tau^3 + \tau h^{\ell+1} + h^{\ell+1}\right) \\ &\leq (1 + C\tau) e_k + C\left(\tau^3 + \tau h^{\ell+1} + h^{\ell+1}\right). \end{aligned}$$

Applying a discrete Gronwall lemma to the above recursion then gives

$$\begin{aligned} e_{n+1} &\leq \mathrm{e}^{CT} e_0 + C\left(\tau^2 + h^{\ell+1} + \frac{h^{\ell+1}}{\tau}\right) \\ &\leq C\left(\tau^2 + h^{\ell+1} + \frac{h^{\ell+1}}{\tau}\right), \end{aligned}$$

which is the desired bound as the constant C can be chosen uniformly in $[0, T]$. This follows from the regularity result (Theorem 2.1) which gives us the desired bound for Theorem 2.8 if $f_0 \in \mathcal{C}^{\max\{\ell+1,3\}}$ is non-negative and compactly supported with respect to velocity. □

2.10. Extension to higher dimensions. In three dimensions the splitting scheme is given by (for simplicity we consider a single time step only and thus drop the corresponding indices)

$$S^{(A)} f(\boldsymbol{x}, \boldsymbol{v}) = f\left(\boldsymbol{x} - \frac{\tau}{2}\boldsymbol{v}, \boldsymbol{v}\right), \tag{2.10}$$

$$S^{(B)} f(\boldsymbol{x}, \boldsymbol{v}) = f(\boldsymbol{v}, \boldsymbol{x} - \tau \boldsymbol{E}(f_{1/2}, \boldsymbol{x})). \tag{2.11}$$

The expression in equation (2.10) can be easily decomposed into three translation in a single dimension, i.e.

$$S^{(A)} = \mathrm{e}^{\frac{\tau}{2}A_x} \mathrm{e}^{\frac{\tau}{2}A_y} \mathrm{e}^{\frac{\tau}{2}A_z}$$

with $A_x = -v_x \partial_x$, $A_y = -v_y \partial_y$, and $A_z = -v_z \partial_z$.

The discussion is more subtle for the expression in equation (2.11). In this case we can still use the decomposition given above; however, if we introduce a space discretization we will have to project back not onto a $1 + 1$ dimensional space but onto $1 + 3$ dimensional space. This is an important implementation detail; however, the convergence proof is (except for notational difficulties) unaffected.

Most of the derivation in this paper is conducted within the framework of abstract operators. Such results are equally applicable to the three dimensional case. Note that in Lemmas 2.6 and 2.7 we have to consider a more general differentiation operator (i.e. a directional derivative). However, since the existence and regularity results are not restricted to the $1 + 1$ dimensional case (see [14]), we expect that these results remain valid as well.

It then remains to generalize the discussion given in [11] to multiple dimensions in the case of the Vlasov–Poisson equation. The abstract results hold independently of the dimension and the specific details of the operators A and B. However, the remaining computations, which have to be conducted to show that the three dimensional Vlasov–Poisson equations in fact fits into the framework given in [11], are at the very least tedious in nature.

In summary, we expect the convergence result presented here to be valid also in the three-dimensional case. A formal proof of this statement would be interesting in its own. However, such an undertaking is beyond the scope of the current paper.

3. Numerical simulations. The purpose of this section is to perform a number of numerical simulations in order to establish the validity of the implementation. The recurrence phenomenon in the context of higher order implementations in space is discussed in section 3.1. In section 3.2 the order of the method in the strong Landau damping problem is investigated. We will also reproduce some medium time integration results for linear Landau damping (section 3.3) and investigate the stability for the Molenkamp–Crowley test problem (section 3.4).

The computer program used for the numerical simulations performed in this section is implemented in `C++`. It employs heavily the concept of templates and operator overloading to provide a compact implementation that is easily extendable to the multi dimensional case. As a result, the core program consists of only about 800 lines of source code (excluding unit tests but including all the logic needed to carry out the simulations in this section) while still maintaining an implementation with adequate performance to carry out a wide range of numerical experiments.

For all simulations conducted in this section, we employ periodic boundary conditions in both the space- and the velocity-direction. The value for $v_{\max}$ is chosen, by conducting numerical experiments, such that it does not intefer with the accuracy of the simulations conducted.

3.1. Recurrence. It is well known that piecewise constant approximations in velocity space lead to a recurrence phenomenon that is purely numerical in origin. This behavior has been investigated, for example, in [8] and [18]. In [28] it is demonstrated by a number of numerical experiments that in the weak Landau damping problem this phenomenon is also purely a numerical artefact.

From an analytical point of view the recurrence phenomenon is most easily understood for an advection equation, i.e. a function $f(t, x, v)$ satisfying the equation

$$\partial_t f = -v\partial_x f. \tag{3.1}$$

For its numerical solution consider a piecewise constant approximation of f in velocity space. This approximation results in slices in velocity space that correspond to the average velocity in a particular cell. Let us further assume that the velocity space $[-v_{\max}, v_{\max}]$ consists of an odd number of cells and that the interval $[0, 4\pi]$ is employed in the space direction. Then the solution of (3.1) is a periodic function in time and the period p is easily determined to be

$$h_v p = 4\pi.$$

That this is only a numerical artefact is a simple consequence of the fact that p tends to infinity as h_v tends to 0. However, for the purpose of this section it is instructive to compute the exact solution for the following initial value

$$f_0(x, v) = \frac{\mathrm{e}^{-v^2/2}}{\sqrt{2\pi}}\big(1 + 0.01\cos(0.5x)\big).$$

The solution of (3.1) is then given by

$$f(t, x, v) = \frac{\mathrm{e}^{-v^2/2}}{\sqrt{2\pi}}\big(1 + 0.01\cos(0.5x - 0.5vt)\big).$$

This function, however, is not periodic in time (with a period being independent of v). To represent this more clearly, we compute the electric energy

$$\mathcal{E}(t) = \int_0^{4\pi} E(t, x)^2 \mathrm{d}x = \frac{\pi}{1250}\mathrm{e}^{-0.25t^2}, \tag{3.2}$$

where the electric field $E(t,x)$ is determined as before by

$$E(t,x) = \int_0^L K(x,y) \left(\int_{\mathbb{R}} f(t,y,v) \mathrm{d}v - 1 \right) \mathrm{d}y.$$

Note that the kernel $K(x,y)$ is defined in equation (2.3). Thus, the electric energy is exponentially decreasing for the exact solution (but periodic in time for the numerical solution). One might naively expect that this phenomenon vanishes as soon as one considers an approximation of degree at least 1 in the velocity direction. While it is true that the solution is no longer periodic, as can be seen from Figure 3.1, errors in the velocity approximation still result in a *damped recurrence* of the electric field. Note that the size of this recurrence effect seems to be determined by the space discretization error.

As mentioned before the recurrence phenomenon is also visible in the Landau damping problem. This is shown in Figure 3.2.

3.2. Order. We can investigate both, the order of convergence in time (i.e. where the space error is small enough over the range of step sizes τ we are interested in) and the order of convergence in space (i.e. where the step size is chosen small enough such that the time integration error is negligible). The order of the time integration has already been investigated in [11]. Thus, we focus on the convergence order in space.

Let us consider the Vlasov–Poisson equations in 1+1 dimensions together with the initial value

$$f_0(x,v) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-v^2/2} \big(1 + \alpha \cos(0.5x)\big).$$

This problem is called Landau damping. For $\alpha = 0.01$ the problem is called linear or weak Landau damping and for $\alpha = 0.5$ it is referred to as strong or non-linear Landau damping. As can be seen, for example, in [9, 13] and [25] Landau damping is a popular test problem for Vlasov codes.

In our numerical simulations, all errors are computed with respect to a reference solution (such as to exclude unwanted effects from the time discretization). The reference solution uses 512 cells with $\ell = 2$ and a time step of $\tau = 0.1$. The results for strong Landau damping are given, up to order 3, in Figure 3.3. It can be seen that the accuracy improves with the desired order as the cell size decreases. Thus, the results are in good agreement with the theory developed in this paper.

3.3. Landau damping. The Landau damping problem has already been introduced in the previous section. In this section we are not interested in the desired order of the numerical algorithm but in the comparison with the exact solution of the Vlasov–Poisson equations. However, since an exact solution of the full Vlasov–Poisson equations is not known we will instead use a result that gives us the asymptotic decay rate γ of the electric field in the case of weak Landau damping (see e.g. [1]). Thus, we compare the decay of the energy stored in the electric field with the graph of $\mathrm{e}^{-2\gamma t}$, where $\gamma \approx 0.1533$. A number of such simulations have already been conducted (see e.g. [28]). However, due to the recurrence effect usually a large number of cells have to be employed in order to get accurate results for medium to long time intervals. For reference we note that [28] uses $N_x = N_v = 1024$ whereas [16] uses up to $N_x = 2000, N_v = 1600$. The results of our simulation are shown in Figure 3.4 (the number of cells is $N_x = N_v = 256$ and $N_x = N_v = 128$ for $\ell = 1, 2$ and $N_x = N_v = 64$ for $\ell = 1, 2, 4$). This experiment clearly shows that high-order approximations in space and velocity pay off.

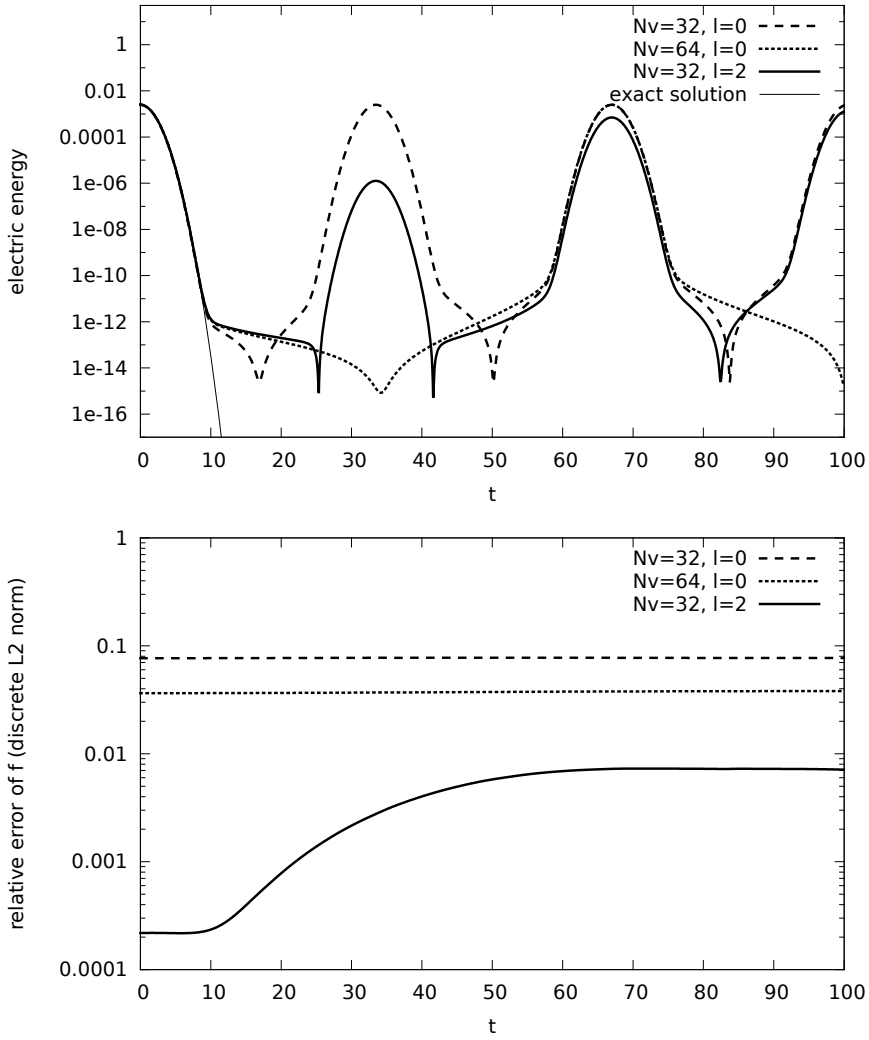

FIG. 3.1. *Recurrence phenomenon for the advection equation (top). Note that while there is no periodicity in the second-order approximation a recurrence-like effect from the finite cell size is still visible. The (absolute) error as compared with the exact solution given in* (3.2) *in the discrete* L^2 *norm (bottom) behaves as expected. In all simulations* 32 *cells and an approximation of order* 2 *(i.e.* $\ell = 1$*) have been employed in the space direction. The number of cells and the order of discretization in the velocity direction is indicated in the legend. In all computations* $\tau = 0.05$ *is used.*

3.4. Stability. In advection dominated problems instabilities can occur if the numerical integration is not performed exactly. In [21] this is shown for the Molenkamp–Crowley test problem, i.e.

$$\begin{cases} \partial_t f(t,x,y) = 2\pi(y\partial_x - x\partial_y) f(t,x,y) \\ \quad f(0,x,v) = f_0(x,y), \end{cases} \tag{3.3}$$

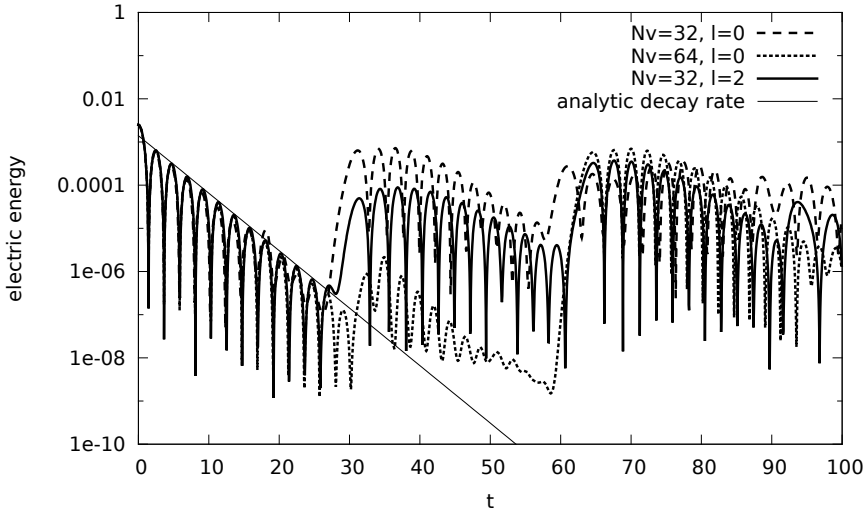

FIG. 3.2. *The recurrence phenomenon for Landau damping. Note that even though a higher order of approximation in the velocity direction improves the solution a recurrence-like effect is still visible. In all computations* $\tau = 0.05$ *is used.*

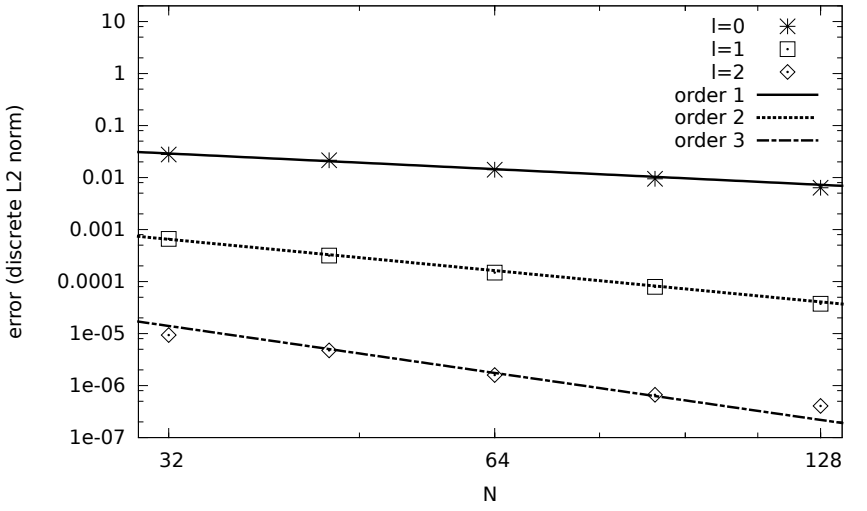

FIG. 3.3. *Error of the particle density function* $f(1, \cdot, \cdot)$ *for non-linear Landau damping on the domain* $[0, 4\pi] \times [-6, 6]$ *as a function of* N*, the number of cells in both the space and velocity direction.*

where

$$f_0(x, y) = \begin{cases} \cos^2(2\pi r) & r \leq \frac{1}{4} \\ 0 & \text{else} \end{cases}$$

with $r^2 = (x + \frac{1}{2})^2 + y^2$. The solution travels along a circle with period 1. We will solve the same problem using the algorithm presented in this paper and show that no

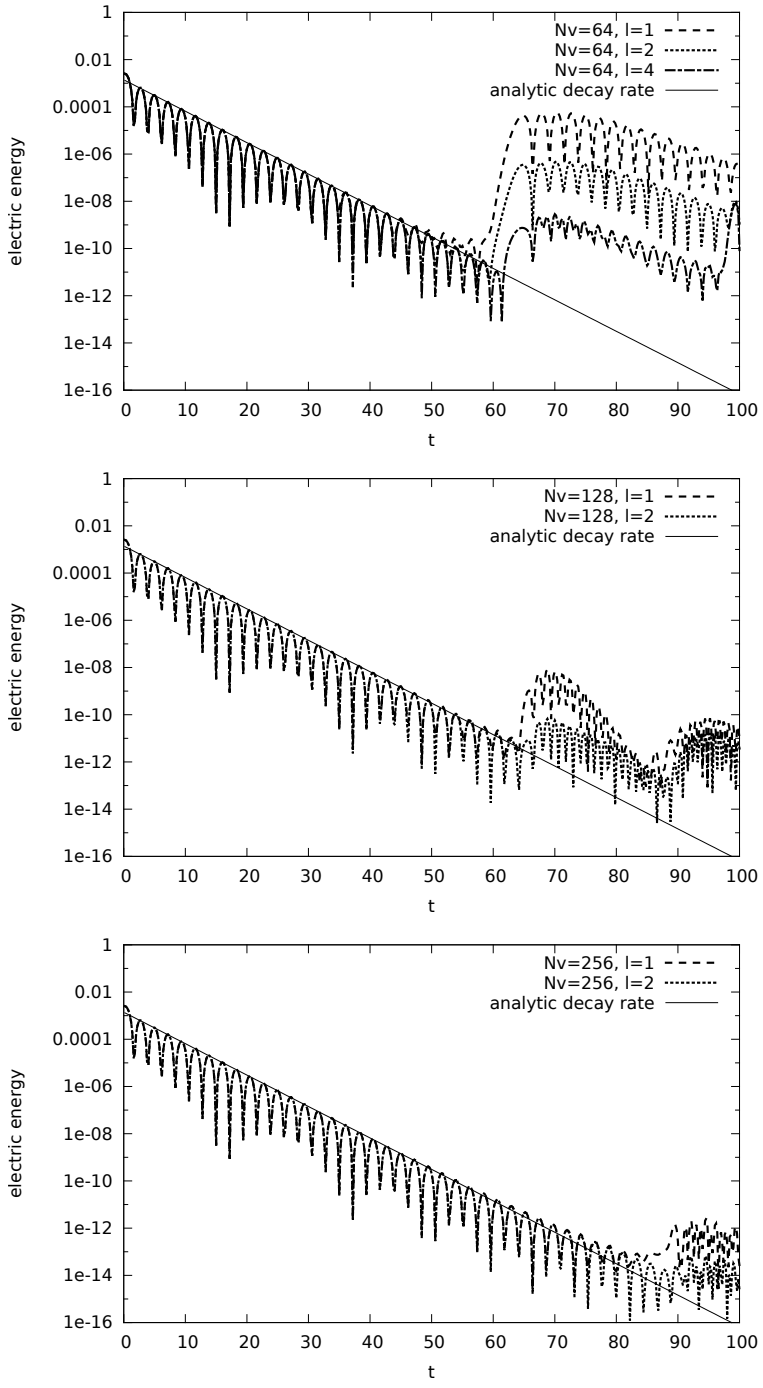

FIG. 3.4. *The decay of the electric field is shown for $N_x = N_v = 256$ (top), $N_x = N_v = 128$ (middle), and $N_x = N_v = 64$ (bottom). In all cases a relatively large time step of $\tau = 0.2$ is employed.*

instabilities occur if a quadrature rule of appropriate order is used. This results are given in Figure 3.5. Note that this is exactly what is expected based on the theoretical analysis done in section 2. However, it is not true that such stability results hold for arbitrary schemes (see e.g. [21], where a finite element scheme of order 2 is shown to be unstable for most quadrature rules).

4. Conclusion. In the present paper we have extended the convergence analysis conducted in [11] to the fully discretized case using a discontinuous Galerkin approximation in space. The results are only presented in case of the $1 + 1$ dimensional Vlasov–Poisson equation. However, we have given a short argument outlining the reasons why we expect that the extension to multiple dimensions, although tedious, is possible. In addition, we have presented a number of numerical simulations that investigate the behavior of the proposed algorithm. These simulations suggest, in line with similar results for other high order schemes, that the algorithm is advantageous (in that fewer data points for the space discretization are required) as compared to similar schemes which employ a piecewise constant approximation in space and velocity. In addition, the desirable features of the discontinuous Galerkin method, namely the locality of the approximation, is preserved.

Acknowledgments. We thank the referees for giving constructive remarks that helped us to improve the presentation of this paper.

REFERENCES

[1] T.D. Arber and R. Vann, *A critical comparison of Eulerian-grid-based Vlasov solvers*, J. Comput. Phys., 180 (2002), pp. 339–357.
[2] E.A. Belli, *Studies of numerical algorithms for gyrokinetics and the effects of shaping on plasma turbulence*, PhD thesis, Princeton University, 2006.
[3] N. Besse, *Convergence of a semi-Lagrangian scheme for the one-dimensional Vlasov-Poisson system*, SIAM J. Numer. Anal., 42 (2005), pp. 350–382.
[4] ———, *Convergence of a high-order semi-Lagrangian scheme with propagation of gradients for the one-dimensional Vlasov-Poisson system*, SIAM J. Numer. Anal., 46 (2008), pp. 639–670.
[5] N. Besse and M. Mehrenberger, *Convergence of classes of high-order semi-Lagrangian schemes for the Vlasov-Poisson system*, Math. Comp., 77 (2008), pp. 93–123.
[6] M. Bostan and N. Crouseilles, *Convergence of a semi-Lagrangian scheme for the reduced Vlasov-Maxwell system for laser-plasma interaction*, Numer. Math., 112 (2009), pp. 169–195.
[7] F. Charles, B. Després, and M. Mehrenberger, *Enhanced convergence estimates for semi-Lagrangian schemes. Application to the Vlasov–Poisson equation*, SIAM J. Numer. Anal., 51 (2013), pp. 840–863.
[8] C.Z. Cheng and G. Knorr, *The integration of the Vlasov equation in configuration space*, J. Comput. Phys., 22 (1976), pp. 330–351.
[9] N. Crouseilles, E. Faou, and M. Mehrenberger, *High order Runge–Kutta–Nyström splitting methods for the Vlasov–Poisson equation.* http://hal.inria.fr/inria-00633934/PDF/cfm.pdf.
[10] N. Crouseilles, M. Mehrenberger, and F. Vecil, *Discontinuous Galerkin semi-Lagrangian method for Vlasov–Poisson*, in ESAIM Proc., CEMRACS 2010, vol. 32, 2011, pp. 211–230.
[11] L. Einkemmer and A. Ostermann, *Convergence analysis of Strang splitting for Vlasov–type equations.* To appear in SIAM J. Numer. Anal., 2013.
[12] M.R. Fahey and J. Candy, *GYRO: A 5-d gyrokinetic-Maxwell solver*, Proceedings of the ACM/IEEE SC2004 Conference, (2008), p. 26.
[13] F. Filbet and E. Sonnendrücker, *Comparison of Eulerian Vlasov solvers*, Comput. Phys. Comm., 150 (2003), pp. 247–266.
[14] R.T. Glassey, *The Cauchy Problem in Kinetic Theory*, SIAM, Philadelphia, 1996.
[15] R.E. Heath, *Analysis of the Discontinuous Galerkin Method Applied to Collisionless Plasma Physics*, PhD thesis, The University of Texas at Austin, 2007.

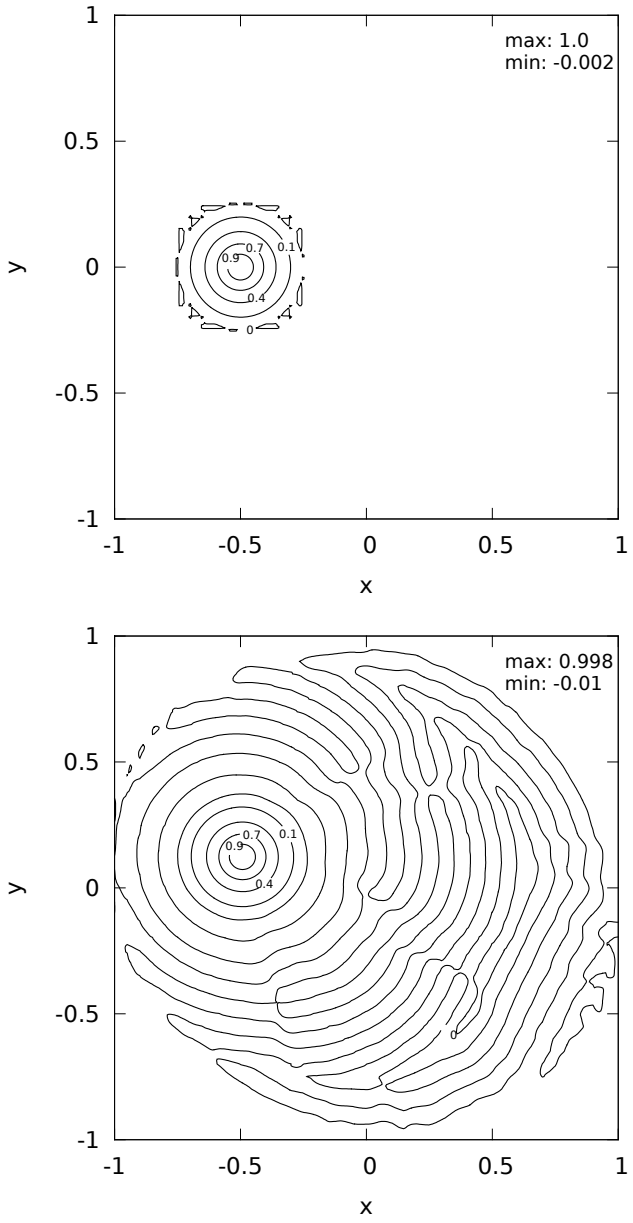

FIG. 3.5. *Stability for the Molenkamp–Crowley test problem. The initial value is displayed at the top. The numerical solution after 60 revolutions with* $\tau = 0.02$, $N_x = N_v = 40$, *and* $\ell = 2$ *is shown at the bottom. As expected no numerical instabilities are observed. The negative values in the numerical solution are a consequence of the space discretization error and are propagated in space by the numerical algorithm (see the complex contour line of* 0 *at the bottom). However, this fact has no influence on the stability of the scheme.*

[16] R.E. Heath, I.M. Gamba, P.J. Morrison, and C. Michler, *A discontinuous Galerkin method for the Vlasov-Poisson system*, J. Comput. Phys., 231 (2011), pp. 1140–1174.
[17] T. Jahnke and C. Lubich, *Error bounds for exponential operator splittings*, BIT, 40 (2000), pp. 735–744.
[18] S.M.H. Jenab, I. Kourakis, and H. Abbasi, *Fully kinetic simulation of ion acoustic and dust-ion acoustic waves*, Phys. Plasmas, 18 (2011), p. 073703.
[19] A. Mangeney, F. Califano, C. Cavazzoni, and P. Travnicek, *A numerical scheme for the integration of the Vlasov-Maxwell system of equations*, J. Comput. Phys., 179 (2002), pp. 495–538.
[20] M. Mehrenberger, C. Steiner, L. Marradi, N. Crouseilles, E. Sonnendrücker, and B. Afeyan, *Vlasov on GPU (VOG project)*, arXiv preprint, arXiv:1301.5892, (2013).
[21] K.W. Morton, A. Priestley, and E. Süli, *Stability of the Lagrange-Galerkin method with non-exact integration*, Modél. Math. Anal. Numér., 22 (1988), pp. 625–653.
[22] G.M. Phillips, *Interpolation and Approximation by Polynomials*, Springer, 2003.
[23] J.M. Qiu and C.W. Shu, *Positivity preserving semi-Lagrangian discontinuous Galerkin formulation: theoretical analysis and application to the Vlasov–Poisson system*, J. Comput. Phys., 230 (2011), pp. 8386–8409.
[24] T. Respaud and E. Sonnendrücker, *Analysis of a new class of forward semi-Lagrangian schemes for the 1D Vlasov Poisson equations*, Numer. Math., 118 (2011), pp. 329–366.
[25] J.A. Rossmanith and D.C. Seal, *A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov–Poisson equations*, J. Comput. Phys., 230 (2011), pp. 6203–6232.
[26] A. Shadrin, *Twelve proofs of the Markov inequality*, in Approximation Theory: A volume dedicated to Borislav Bojanov, D.K. Dimitrov et al., eds., Marin Drinov Acad. Publ. House, Sofia, 2004, pp. 233–298.
[27] H. Yang and F. Li, *Error estimates of Runge–Kutta discontinuous Galerkin methods for the Vlasov–Maxwell system*, arXiv preprint, arXiv:1306.0636, (2013).
[28] T. Zhou, Y. Guo, and C.W. Shu, *Numerical study on Landau damping*, Phys. D, 157 (2001), pp. 322–333.

C Exponential integrators on graphic processing units

Journal	Proceedings of the 2013 International Conference on High Performance Computing & Simulation (HPCS 2013)
Authors	Lukas Einkemmer, Alexander Ostermann
submitted	26.02.2013
accepted	23.04.2013

Exponential Integrators on Graphic Processing Units

Lukas Einkemmer, Alexander Ostermann
Department of Mathematics
University of Innsbruck
Innsbruck, Austria
lukas.einkemmer@uibk.ac.at, alexander.ostermann@uibk.ac.at

***Abstract*—In this paper we revisit stencil methods on GPUs in the context of exponential integrators. We further discuss boundary conditions, in the same context, and show that simple boundary conditions (for example, homogeneous Dirichlet or homogeneous Neumann boundary conditions) do not affect the performance if implemented directly into the CUDA kernel. In addition, we show that stencil methods with position-dependent coefficients can be implemented efficiently as well. As an application, we discuss the implementation of exponential integrators for different classes of problems in a single and multi GPU setup (up to 4 GPUs). We further show that for stencil based methods such parallelization can be done very efficiently, while for some unstructured matrices the parallelization to multiple GPUs is severely limited by the throughput of the PCIe bus.**

***Keywords*—GPGPU, exponential integrators, time integration of differential equations, stencil methods, multi GPU setup**

I. INTRODUCTION

The emergence of graphic processing units as a massively parallel computing architecture as well as their inclusion in high performance computing systems have made them an attractive platform for the parallelization of well established computer codes.

Many problems that arise in science and engineering can be modeled as differential equations. In most circumstances the resulting equations are sufficiently complex such that they can not be solved exactly. However, an approximation computed by the means of a given numerical scheme can still serve as a valuable tool for scientists and engineers. The collection of techniques generally referred to as general-purpose computing on graphics processing units (GPGPU) provide the means to speed up such computations significantly (see e.g. [1] or [2]).

If a finite difference approximation in space is employed (such methods are widely used in computational fluid dynamics, for example), stencil methods provide an alternative to storing the matrix in memory (see e.g. [3]). In many instances, this is advantageous both from a memory consumption as well as from a performance standpoint. The resulting system of ordinary differential equations then has to be integrated in time.

Much research has been devoted to the construction of efficient time integration schemes as well as their implementation (see e.g. [4] and [5]). The implementation of Runge–Kutta methods, which are the most widely known time integration schemes, on GPUs for ordinary differential equations can result in a significant speedup (see [1]). However, a class of problems has been identified, so called stiff problems, where standard integration routines (such as the above mentioned Runge–Kutta methods) are inefficient (see e.g. [6]).

Exponential integrators are one class of methods that avoid the difficulties of Runge–Kutta method if applied to stiff problems. For such schemes analytical functions (e.g. the exponential function) of large matrices have to be computed. Exponential integrators and some of their applications are discussed in detail in [6]. In this paper we will consider a polynomial interpolation scheme to compute the matrix functions; this essentially reduces the problem of efficiently implementing exponential integrators to sparse matrix-vector multiplication as well as computing the nonlinearity of a given differential equation. The computation of matrix-vector multiplications, e.g. by using stencil methods, is usually the most time intensive part of any exponential integrator; thus, an efficient implementation of stencil methods is vital.

A. Research problems & Results

In the literature, see section II-B, stencil methods are considered for trivial boundary conditions in the context of a differential operator with constant coefficients (i.e. the Laplacian). Such simplifying assumptions, however, are usually not satisfied in a given application. It is not clear from the literature how much stencil methods can be extended beyond the situation described above while still maintaining an efficient implementation. We propose a method based on the integration of boundary conditions and position-dependent coefficients directly into the CUDA kernel and show that such methods can be applied widely without a significant impact on performance.

In addition, it has been shown in [2] that stencil methods can be efficiently parallelized to at least 4 GPUs. Our objective is to show that such results can be generalized to implementations of exponential integrators for a large class of nonlinearities.

The remainder of this paper is structured as follows. In section II-A a introduction explaining the GPU architecture and

the corresponding programming model is given. In addition, we discuss previous work which considers the implementation of stencil methods on GPUs and elaborate on the necessary steps to efficiently implement an exponential integrator (sections II-B and II-C, respectively). In section III we present our results as summarized below.

- Stencil methods that include simple, but non-trivial, boundary conditions, such as those required in many applications, can still be efficiently implemented on GPUs (section III-A). For homogeneous boundary conditions on the C2075 33.5 Gflops/s are observed.
- Position dependent coefficients (such as a position dependent diffusion) can efficiently be implemented on the GPU provided that the coefficients are not extremely expensive to compute (section III-B). For a real world example 16 Gflops/s are observed.
- A wide class of nonlinearities can be computed efficiently on the GPU (section III-C).
- The parallelization of exponential integrators to multiple GPUs can be conducted very efficiently for discretized differential operators (perfect scaling to at least 4 GPUs) and is mainly limited by the throughput of the PCIe express bus for unstructured matrices (section III-D).

Finally, we conclude in section IV.

II. BACKGROUND & MOTIVATION

A. GPU architecture

A graphic processing unit (GPU) is a massively parallel computing architecture. At the time of writing two frameworks to program such systems, namely OpenCL and NVIDIA's CUDA, are in widespread use. In this section we will discuss the hardware architecture as well as the programming model of the GPU architecture using NVIDIA's CUDA (all our implementations are CUDA based). Note, however, that the principles introduced here can, with some change in terminology, just as well be applied to the OpenCL programming model. For a more detailed treatment we refer the reader to [7].

The hardware consists of so called SM (streaming multiprocessors) that are divided into cores. Each core is (as the analogy suggests) an independent execution unit that shares certain resources (for example shared memory) with other cores, which reside on the same streaming multiprocessor. For example, in case of the C2075, the hardware consists in total of 448 cores that are distributed across 14 streaming multiprocessors of 32 cores each.

For scheduling, however, the hardware uses the concept of a warp. A warp is a group of 32 threads that are scheduled to run on the same streaming multiprocessor (but possibly on different cores of that SM). On devices of compute capability 2.0 and higher (e.g. the C2075) each SM consists of 32 cores (matching each thread in a warp to a single core). However, for the C1060, where 240 cores are distributed across 30 SM of 8 core each, even threads in the same warp that take exactly the same execution path are not necessarily scheduled to run in parallel (on the instruction level).

To run a number of threads on a single SM has the advantage that certain resources are shared among those threads; the most notable being the so called shared memory. Shared memory essentially acts as an L1 cache (performance wise) but can be fully controlled by the programmer. Therefore, it is often employed to avoid redundant global memory access as well as to share certain intermediate computations between cores. In addition, devices of compute capability 2.0 and higher are equipped with a non-programmable cache.

The global memory is a RAM (random access memory) that is shared by all SM on the entire GPU. For the C1060 and C2075 GPUs used in this paper the size of the global memory is 4 GB and 6 GB respectively (with memory bandwidth of 102.4 GB/s and 141.7 GB/s respectively), whereas the shared memory is a mere 16 KB for the C1060 and about 50 KB for the C2075. However, this memory is available per SM.

From the programmer some of these details are hidden by the CUDA programming model (most notably the concept of SM, cores, and warps). If a single program is executed on the GPU we refer to this as a grid. The programmer is responsible for subdividing this grid into a number of blocks, whereas each block is further subdivided into threads. A thread in a single block is executed on the same SM and therefore has access to the same shared memory and cache.

GPUs are therefore ideally suited to problems which are compute bound. However, also memory bound problems, such as sparse matrix-vector multiplication, can significantly benefit from GPUs. We will elaborate on this statement in the next section.

B. Stencil methods and matrix-vector products on GPUs

The parallelization of sparse matrix-vector products to GPUs has been studied in some detail. Much research effort in improving the performance of sparse matrix-vector multiplication on GPUs has focused on developing more efficient data structures (see e.g. [8] or [9]). This is especially important on GPUs as coalesced memory access is of paramount importance if optimal performance is to be achieved. Data structures, such as ELLRT, facilitate coalesced memory access but require additional memory. This is somewhat problematic as on a GPU system memory is limited to a greater extend than on traditional clusters. To remedy this situation a more memory efficient data structure has been proposed, for example, in [10]. Nevertheless, all such methods are extremely memory bound.

On the other hand, the parallelization of finite difference computations (called stencil methods in this context) to GPUs has been studied, for example, in [2] and [3]. Even though such methods do not have to store the matrix in memory they are still memory bound; for example, in [3] the flops per byte ratio is computed to be 0.5 for a seven-point stencil (for double

precision computations) which is still far from the theoretical rate of 3.5 that a C0275 can achieve. In [3] a performance of 36.5 Gflops/s has been demonstrated for a seven-point stencil on a GTX280.

Both papers mentioned above do not consider boundary conditions in any detail. However, in applications of science and engineering where exponential integrators are applied at least simple boundary conditions have to be imposed (see e.g. [6]). In addition, in the literature stated above only the discretization of the Laplacian is considered. However, often position-dependent coefficients have to be employed (to model a position-dependent diffusion as in [11], for example). In this case it is not clear if stencil methods retain their superior performance characteristics (as compared to schemes that store the matrix in memory). We will show in sections III-A and III-B that for many applications both of these difficulties can be overcome and stencil methods on GPUs can be implemented efficiently.

C. Exponential integrators

The step size for the time integration of stiff ordinary differential equations (or the semidiscretization of partial differential equations) is usually limited by a stability condition. In order to overcome this difficulty, implicit schemes are employed that are usually stable for much larger step sizes; however, such schemes have to solve a nonlinear system of equations in each time step and are thus costly in terms of performance. In many instances the stiffness of the differential equation is located in the linear part only. In this instance, we can write our differential equation as a semilinear problem

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) + Au(t) = g(u(t)), \tag{1}$$

where in many applications A is a matrix with large negative eigenvalues and g is a nonlinear function of $u(t)$; it is further assumed that appropriate initial conditions are given. The boundary conditions are incorporated into the matrix A. Since the linear part can be solved exactly, a first-order method, the exponential Euler method, is given by

$$u_{n+1} = \mathrm{e}^{-hA}u_n + h\varphi_1\left(-hA\right)g(u_n), \tag{2}$$

where φ_1 is an entire function. In [6] a review of such methods, called exponential integrators, is given and various methods of higher order are discussed. The main advantage, compared to Runge–Kutta methods, is that an explicit method is given for which the step size is only limited by the nonlinearity. It has long been believed that the computation of the matrix functions in (2) can not be carried out efficiently. However, if a bound of the field of values of A is known a priori, for example, polynomial interpolation is a viable option. In this case the application of Horner's scheme reduces the problem to repeated matrix-vector products of the form

$$(\alpha A + \beta I)x, \tag{3}$$

where $A \in \mathbb{K}^{n\times n}$ is a sparse matrix, I is the identity matrix, $x \in \mathbb{K}^n$, and $\alpha, \beta \in \mathbb{K}$ with $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. That such a product can be parallelized to small clusters has been shown in [12] (for an advection-diffusion equation that is discretized in space by finite differences).

Finally, let us discuss the evaluation of the nonlinearity. In many instances the nonlinearity can be computed pointwise. In this case its evaluation is expected to be easily parallelizable to GPUs. In section III-C this behavior is confirmed by numerical experiments. If the nonlinearity does include differential operators, such as in Burgers' equation, the evaluation is essentially reduced to sparse-matrix vector multiplication, which we will discuss in some detail in this paper (in the context of stencil methods).

III. RESULTS

A. Stencil methods with boundary conditions

Let us focus our attention first on the standard seven-point stencil resulting from a discretization of the Laplacian in three dimensions, i.e.

$$\begin{aligned}(\Delta x)^2\left(Au\right)_{i_x,i_y,i_z} = &-6u_{i_x,i_y,i_z}\\ &+ u_{i_x+1,i_y,i_z} + u_{i_x-1,i_y,i_z}\\ &+ u_{i_x,i_y+1,i_z} + u_{i_x,i_y-1,i_z}\\ &+ u_{i_x,i_y,i_z+1} + u_{i_x,i_y,i_z-1},\end{aligned}$$

where Δx is the spacing of the grid points. The corresponding matrix-vector product given in (3) can then be computed without storing the matrix in memory. For each grid point we have to perform at least 2 memory operations (a single read and a single store) as well as 10 floating point operations (6 additions and 2 multiplication for the matrix-vector product as well as single addition and multiplication for the second part of (3)).

One could implement a stencil method that employs 8 memory transactions for every grid point. Following [3] we call this the *naive* method. On the other hand we can try to minimize memory access by storing values in shared memory or the cache (note that the C1060 does not feature a cache but the C2075 does). Since no significant 3D slice fits into the relatively limited shared memory/cache of both the C1060 and C2075, we take only a 2D slice and iterate over the remaining index. Similar methods have been implemented, for example, in [3] and [2]. We will call this the *optimized* method.

To implement boundary conditions we have two options. First, a stencil method can be implemented that considers only grid points that lie strictly in the interior of the domain. Second, we can implement the boundary conditions directly into the CUDA kernel. The approach has the advantage that all computations can be done in a single kernel launch. However, conditional statements have to be inserted into the kernel. Since the kernel is memory bound, we do not expect a significant performance decrease at least for boundary conditions that do not involve the evaluation of complicated functions.

The results of our numerical experiments (for both the naive and optimized method) are given in Table I. Before we discuss the results let us note that on a dual socket Intel Xeon E5355 system the aggressively hand optimized stencil method implemented in [3] gives 2.5 Gflops/s.

In [3] a double precision performance of 36.5 Gflops/s is reported for a GTX280 of compute capability 1.3. However, the theoretical memory bandwidth of the GTX280 is 141.7 GB/s and thus, as we have a memory bound problem, it has to be compared mainly to the C2075 (which has the same memory bandwidth as the GTX280). Note that the C1060 (compute capability 1.1) has only a memory bandwidth of 102.4 GB/s. In our case we get 39 Gflops/s for no boundary conditions and 33.5 Gflops/s for homogeneous Dirichlet boundary conditions. Since we do not solve exactly the same problem, a direct comparison is difficult. However, it is clear that the implemented method is competitive especially since we do not employ any tuning of the kernel parameters.

We found it interesting that for the C2075 (compute capability 2.0) there is only a maximum of 30% performance decrease if the naive method is used instead of the optimized method for none or homogeneous boundary conditions (both in the single and double precision case). Thus, the cache implemented on a C2075 works quite efficiently in this case. However, we can get a significant increase in performance for more complicated boundary conditions by using the optimized method. In the single precision case the expected gain is offset, in some instances, by the additional operations that have to be performed in the kernel (see Table I).

Finally, we observe that the performance for homogeneous Dirichlet boundary conditions is at most 10% worse than the same computation which includes no boundary conditions at all. This difference completely vanishes if one considers the optimized implementation. This is no longer true if more complicated boundary conditions are prescribed. For example, if we set

$$f(x, y, z) = z(1 - z)xy,$$

for $(x, y, z) \in \partial([0, 1]^3)$ or

$$f(x, y, z) = \sin(\pi z) \exp(-xy),$$

for $(x, y, z) \in \partial([0, 1]^3)$, the performance is decreased by a factor of about 2 for the C2075 and by a factor of 5-7 for the C1060. Thus, in this case it is clearly warranted to perform the computation of the boundary conditions in a separate kernel launch. Note, however, that the direct implementation is still faster by a factor of 3 as compared to CUSPARSE and about 40 % better than the ELL format (see [13]). The memory requirements are an even bigger factor in favor of stencil methods; a grid of dimension 512^3 would already require 10 GB in the storage friendly CSR format. Furthermore, the implementation of such a kernel is straight forward and requires no division of the domain into the interior and the boundary.

B. Stencil methods with a position-dependent coefficient

Let us now discuss the addition of a position-dependent diffusion coefficient, i.e. we implement the discretization of $D(x, y, z)\Delta u$ as a stencil method (this is the diffusive part of $\nabla \cdot (D\nabla u)$). Compared to the previous section we expect that the direct implementation of the position-dependent diffusion coefficient in the CUDA kernel, for a sufficiently complicated D, results in an compute bound problem. For the particular choice of $D(x, y, z) = 1/\sqrt{1 + x^2 + y^2}$, taken from [11], the results are shown in Table II.

TABLE II
TIMING OF A SINGLE STENCIL BASED MATRIX-VECTOR COMPUTATION FOR A POSITION DEPENDENT DIFFUSION COEFFICIENT GIVEN BY $D(x, y, z) = 1/\sqrt{1 + x^2 + y^2}$. ALL COMPUTATIONS ARE PERFORMED WITH $n = 256^3$.

Double precision

Device	Method	Time
C1060	Stencil (naive)	37 ms (4.5 Gflops/s)
	Stencil (optimized)	42 ms (4 Gflops/s)
C2075	Stencil (naive)	10.7 ms (15.5 Gflops/s)
	Stencil (optimized)	10.5 ms (16 Gflops/s)

Single precision

Device	Method	Time
C1060	Stencil (naive)	37 ms (4.5 Gflops/s)
	Stencil (optimized)	45 ms (3.5 Gflops/s)
C2075	Stencil (naive)	10.2 ms (16.5 Gflops/s)
	Stencil (optimized)	10.3 ms (16 Gflops/s)

Thus, a performance of 16 Gflops/s can be achieved for this particular position-dependent diffusion coefficient. This is a significant increase in performance as compared to a matrix-based implementation. In addition, the same concerns regarding storage requirements, as raised above, still apply equally to this problem. No significant difference between the naive and optimized implementation can be observed; this is due to the fact that this problem is now to a large extend compute bound.

Finally, let us note that the results obtained in Tables I and II are (almost) identical for the $n = 512^3$ case. Thus, for the sake of brevity, we choose to omit those results.

C. Evaluating the nonlinearity on a GPU

For an exponential integrator, usually the most time consuming part is evaluating the exponential and φ_1 function. Fortunately, if the field of values of A can be estimated a priori, we can employ polynomial interpolation to reduce that problem to matrix-vector multiplication; a viable possibility is interpolation at Leja points (see [6]). Then, our problem reduces to the evaluation of a series of matrix-vector products of the form given in (3) and discussed in the previous section and the evaluation of the nonlinearity for a number of intermediate approximations. In this section we will be

TABLE I
TIMING OF A SINGLE STENCIL BASED MATRIX-VECTOR COMPUTATION FOR A NUMBER OF IMPLEMENTATIONS AND BOUNDARY CONDITIONS. THE CORRESPONDING GFLOPS/S ARE SHOWN IN PARENTHESES. ALL COMPUTATIONS ARE PERFORMED WITH $n = 256^3$.

Device	Boundary	Method	Double	Single
C1060	None	Stencil (naive)	13.8 ms (12 Gflops/s)	8.2 ms (20.5 Gflops/s)
		Stencil (optimized)	7.6 ms (22 Gflops/s)	8.0 ms (21 Gflops/s)
	Homogeneous Dirichlet	Stencil (naive)	13.4 ms (12.5 Gflops/s)	7.6 ms (22 Gflops/s)
		Stencil (optimized)	8.8 ms (19 Gflops/s)	9.2 ms (18 Gflops/s)
	$z(1-z)xy$	Stencil (optimized)	36 ms (4.5 Gflops/s)	39 ms (4.5 Gflops/s)
	$\sin(\pi z)\exp(-xy)$	Stencil (optimized)	54 ms (3 Gflops/s)	56 ms (3 Gflops/s)
C2075	None	Stencil (naive)	5.5 ms (30.5 Gflops/s)	3.1 ms (54 Gflops/s)
		Stencil (optimized)	4.3 ms (39 Gflops/s)	2.9 ms (58 Gflops/s)
	Homogeneous Dirichlet	Stencil (naive)	6 ms (28 Gflops/s)	3.5 ms (48 Gflops/s)
		Stencil (optimized)	5 ms (33.5 Gflops/s)	3.9 ms (43 Gflops/s)
	$z(1-z)xy$	Stencil (naive)	12.3 ms (13.5 Gflops/s)	6.9 ms (24 Gflops/s)
		Stencil (optimized)	7 ms (24 Gflops/s)	6.0 ms (28 Gflops/s)
	$\sin(\pi z)\exp(-xy)$	Stencil (naive)	14.3 ms (11.5 Gflops/s)	13.8 ms (12 Gflops/s)
		Stencil (optimized)	9.7 ms (17.5 Gflops/s)	6.8 ms (24.5 Gflops/s)

concerned with the efficient evaluation of the nonlinearity on a GPU.

Since the nonlinearity is highly problem dependent, let us – for the sake of concreteness – take a simple model problem, namely the reaction-diffusion equation modeling combustion in three dimensions (see [14, p. 439])

$$u_t = \Delta v + g(u) \tag{4}$$

with nonlinearity

$$g(u) = \frac{1}{4}(2-u)\mathrm{e}^{20\left(1-\frac{1}{u}\right)}$$

and appropriate boundary conditions as well as an initial condition.

In addition to the discretization of the Laplacian which can be conducted by stencil methods (as described in section III-A) the parallelization of the nonlinearity can be conducted pointwise on the GPU. That is (in a linear indexing scheme) we have to compute

$$\frac{1}{4}(2-u_i)\mathrm{e}^{20\left(1-\frac{1}{u_i}\right)}, \qquad 0 \leq i < n. \tag{5}$$

This computation requires only two memory operations per grid point (one read and one store); however, we have to perform a single division and a single exponentiation. Since those operations are expensive, the problem is expected to be compute bound. The results of our numerical experiments are shown in Table III.

As expected, the GPU has a significant advantage over our CPU based system in this case. Fast math routines can be employed if precision is not critical and the evaluation of the nonlinearity contributes significantly to the runtime of the program. Let us duly note that the speedups observed here can not be extended to the entire exponential integrator as the sparse-matrix vector multiplication is usually the limiting factor.

TABLE III
TIMING OF A SINGLE COMPUTATION OF THE NONLINEARITY GIVEN IN (5). RESULTS FOR BOTH FULL PRECISION COMPUTATIONS AS WELL AS THE FAST MATH ROUTINES IMPLEMENTED IN THE GPU ARE LISTED. AS A REFERENCE A COMPARISON TO A DUAL SOCKET INTEL XENON E5620 SETUP IS PROVIDED.

Double precision

Device	Method	$n = 256^3$	$n = 512^3$
2x Xenon E5620	OpenMP	480 ms	4 s
C1060	Full precision	14.6 ms	120 ms
	Fast math	6.9 ms	55 ms
C2075	Full precision	4.2 ms	33 ms
	Fast math	2.4 ms	20 ms

Single precision

Device	Method	$n = 256^3$	$n = 512^3$
2x Xenon E5620	OpenMP	515 ms	4 s
C1060	Full precision	15.4 ms	120 ms
	Fast math	7.6 ms	61 ms
C2075	Full precision	2.6 ms	34 ms
	Fast math	1.6 ms	19 ms

The nonlinearity of certain semi-linear PDEs resemble more the performance characteristics of the stencil methods discussed in sections III-A and III-B. For example, Burgers' equation, where $g(\boldsymbol{u}) = (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$, falls into this category. Such nonlinearities can be efficiently implemented by the methods discussed in sections III-A and III-B.

If we combine sections III-A, III-B and III-C we have all ingredients necessary to conduct an efficient implementation of exponential integrators on a single GPU. The specific performance characteristics depend on the form of the linear as well as the nonlinear part of the differential equation under consideration. In the next section we will turn our attention to the parallelization of exponential integrators to multiple GPUs.

D. Multiple GPU implementation of exponential integrators

If we consider the problem introduced in (4) to be solved with an exponential integrator, we have at least two possibilities to distribute the workload to multiple GPUs. First, one could compute the different matrix functions on different GPUs. However, since even for higher order schemes we only have to evaluate a small number of distinct matrix functions, this approach is not very flexible and depends on the method under consideration. However, if we are able to implement a parallelization of the matrix-vector product and the nonlinearity onto multiple GPUs, a much more flexible approach would result.

Such an undertaking however is limited by the fact that in the worst case we have to transfer

$$(m-1)n \tag{6}$$

floating point numbers over the relatively slow PCIe bus (m is the number of GPUs whereas n is, as before, the problem size). However, in the case of differential operators only a halo region has to be updated after every iteration and thus the actual memory transfer is a tiny fraction of the value given by (6). Such a procedure was suggested in [12] for use on a cluster, where parallelization is mainly limited by the interconnection between different nodes. For performance reasons on a GPU it is advantageous to first flatten the halo regions in memory and copy it via a single call to `cudaMemcpy` to the device. Then the vector is updated by using that information in a fully parallelized way on the GPU. As can be seen from the results given in Table IV, the problem in (4) shows good scaling behavior (at least) up to 4 GPUs.

TABLE IV
PERFORMANCE COMPARISON FOR THE COMBUSTION MODEL DISCUSSED IN SECTION III-C FOR A SINGLE TIME STEP USING 40 MATRIX-VECTOR PRODUCTS (A TOLERANCE OF TOL $= 10^{-4}$ WAS PRESCRIBED FOR A TIME STEP OF SIZE 10^{-4}). A FINITE DIFFERENCE DISCRETIZATION WITH $n = 256^3$ HAS BEEN USED.

Double precision

Device	Method	Number units	Time
2x Xenon E5620	CSR/OpenMP	2	9.5 s
C1060	Stencil hom. Dirichlet	1	1.5 s
		4	320 ms
C2075	Stencil hom. Dirichlet	1	1.2 s

Single precision

Device	Method	Number units	Time
2x Xenon E5620	CSR/OpenMP	2	5.6 s
C1060	Stencil hom. Dirichlet	1	1.2 s
		4	540 ms
C2075	Stencil hom. Dirichlet	1	210 ms

Let us now discuss a different example. In certain discrete quantum systems, for example, the solution of (see, e.g., [15])

$$\partial_t \psi = H(t)\psi$$

is to be determined, where ψ is a vector with complex entries in a high dimensional vector space and $H(t)$ a Hermitian matrix. Such equations are efficiently solved by using Magnus integrators. In this paper we will use the example of a two spin system in a spin bath. In this case $H(t)$ is independent of time and thus we can, in principle, take arbitrarily large time steps. The matrix H is generated beforehand and stored in the generic CSR format; for 21 spins this yields a vector with $n = 2^{21}$ complex entries and a matrix with approximately $83.9 \cdot 10^6$ non-zero complex entries (the storage requirement is about 2 GB in the double precision and 1 GB in the single precision case). This gives a sparsity of $2{\cdot}10^{-5}$. Note, however, that such quantum systems couple every degree of freedom with every other degree of freedom. Thus, we are in the worst case and have to transfer $(m-1)n$ floating point numbers over the PCIe bus after each iteration.

The results of our numerical experiments are shown in Table V. The implementation used is based on the code given in [16]. However, we have found that for the problem under consideration using a full warp for every row of our matrix results in a performance reduction. Therefore, we use only four threads per row which results, for the specific problem under consideration, in a performance increase of approximately 50%, as compared against the CUSPARSE library (see [17]). Apart from this consideration the code has been adapted to compute the problem stated in (3), which includes an additional term as compared to the sparse matrix-vector multiplication considered in [16]. Clearly the scaling behavior

TABLE V
PERFORMANCE COMPARISON FOR A SYSTEM WITH 21 SPINS. INTEGRATION IS PERFORMED UP TO $t = 10$ WITH A TOLERANCE OF TOL $= 10^{-5}$.

Double precision

Device	Method	Number units	Time
2x Xenon E5620	CSR/OpenMP	2	46 s
C1060	CSR	1	23 s
		2	15 s
		4	15 s
C2075	CSR	1	7.7 s

Single precision

Device	Method	Number units	Time
2x Xenon E5620	CSR/OpenMP	2	44 s
C1060	CSR	1	15 s
		2	10 s
		4	7.5 s
C2075	CSR	1	4 s

in this case is limited by the overhead of copying between the different GPUs. For two GPUs a speedup of about 1.5 can be observed. For any additional GPU no performance gain can be observed. In total a speedup of 3 for double precision and 6 for single precision as compared to a dual socket Xenon configuration is achieved on four C1060 graphic processing units. This is only about 50% better than the speedup of 2 (double precision) and 3 (single precision) achieved with a single C1060 GPU. It should be noted that a GPU centric data format (as discussed in section II-B) could be employed instead of the CSR format. However, also in this case the overhead of copying between different GPUs would persist.

Thus, in this instance the speedups that are achievable in both single and multi GPU configurations are a consequence of an unstructured matrix that makes coalesced memory access as well as parallelizability between different GPUs difficult. The dramatically better performance of the C2075 as shown in Table V is thus expected.

IV. CONCLUSION & OUTLOOK

We have shown that exponential integrators can be efficiently implemented on graphic processing units. For many problems, especially those resulting from the spatial discretization of partial differential equations, this is true for both single and multi GPU setups.

In addition, we have considered stencil based implementations that go beyond periodic boundary conditions and constant diffusion coefficients. Such problems can not be handled by implementations based on the fast Fourier transform, for example. Moreover, section III-A shows that for non-homogeneous boundary conditions the code handling the interior as well as the boundary of the domain has to be separated if optimal performance is to be achieved. However, for homogeneous or piecewise constant boundary condition an implementation directly into the CUDA kernel does not result in any significant performance decrease.

The results presented in this paper show that exponential integrators, for many realistic settings, can be efficiently implemented on GPUs with significant speedups compared to more traditional implementations. Therefore, GPU computing provides a viable way to increase the efficiency of simulations in which exponential integrators are employed. The implementation of exponential integrators on the current generation of GPUs would conceivably result in a further performance increase of our memory bound stencil implementation, as compared to the C2075, as the Kepler architecture offers a memory throughput of up to 250 GB/s. Furthermore, such an implementation is expected to be relatively straightforward as the cache implemented on the newer generations of GPUs works quite well in the case of stencil methods.

REFERENCES

[1] L. Murray, "GPU acceleration of Runge-Kutta integrators," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 94–101, 2012.

[2] P. Micikevicius, "3D Finite Difference Computation on GPUs using CUDA," in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, Washington, DC, USA, 2009, pp. 79–84.

[3] K. Datta, M. Murphy, V. Volkov, S. Williams, and J. Carter, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures," *Journal of Parallel and Distributed Computing*, vol. 69, no. 9, pp. 762–777, 2009.

[4] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed. Springer-Verlag, Berlin, 1993.

[5] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd ed. Springer-Verlag, Berlin, 1996.

[6] M. Hochbruck and A. Ostermann, "Exponential integrators," *Acta Numer.*, vol. 19, pp. 209–286, 2010. [Online]. Available: http://www.journals.cambridge.org/abstract_S0962492910000048

[7] "CUDA C Programming Guide," http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf (Last retrieved September 18, 2013).

[8] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. New York, USA: ACM Press, 2009, p. 18. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1654059.1654078

[9] M. Baskaran and R. Bordawekar, "Optimizing sparse matrix-vector multiplication on GPUs," *IBM Research Report*, vol. RC24704, 2009.

[10] A. Dziekonski, A. Lamecki, and M. Mrozowski, "A memory efficient and fast sparse matrix vector product on a GPU," *Progress in Electromagnetics Research*, vol. 116, pp. 49–63, 2011.

[11] M. Vazquez, A. Berezhkovskii, and L. Dagdug, "Diffusion in linear porous media with periodic entropy barriers: A tube formed by contacting spheres," *J. Chem. Phys*, vol. 129, no. 4, p. 046101, 2008.

[12] M. Caliari, M. Vianello, and L. Bergamaschi, "Interpolating discrete advectionâĂŞdiffusion propagators at Leja sequences," *J. Comput. Appl. Math.*, vol. 172, no. 1, pp. 79–99, Nov. 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0377042704000998

[13] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA NVR-2008-004, Tech. Rep., 2008.

[14] W. Hundsdorfer and J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, 2nd ed. Springer-Verlag, Berlin, Heidelberg, New York, 2007.

[15] H. De Raedt and K. Michielsen, "Computational methods for simulating quantum computers," *arXiv preprint quant-ph/0406210*, 2008.

[16] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA Corporation, NVIDIA Technical Report NVR-2008-004, Dec. 2008.

[17] "CUDA CUSPARSE Library," http://docs.nvidia.com/cuda/pdf/CUDA_CUSPARSE_Users_Guide.pdf (Last retrieved September 18, 2013).

D HPC research overview (book chapter)

Titel	Splitting methods for the Vlasov–Poisson & Vlasov–Maxwell equations
In	Scientific Computing @ uibk (Marco Barden, Alexander Ostermann)
Authors	Lukas Einkemmer, Alexander Ostermann
Publisher	Innsbruck University Press

3 Splitting methods for the Vlasov-Poisson and Vlasov-Maxwell equations

Lukas Einkemmer, Alexander Ostermann

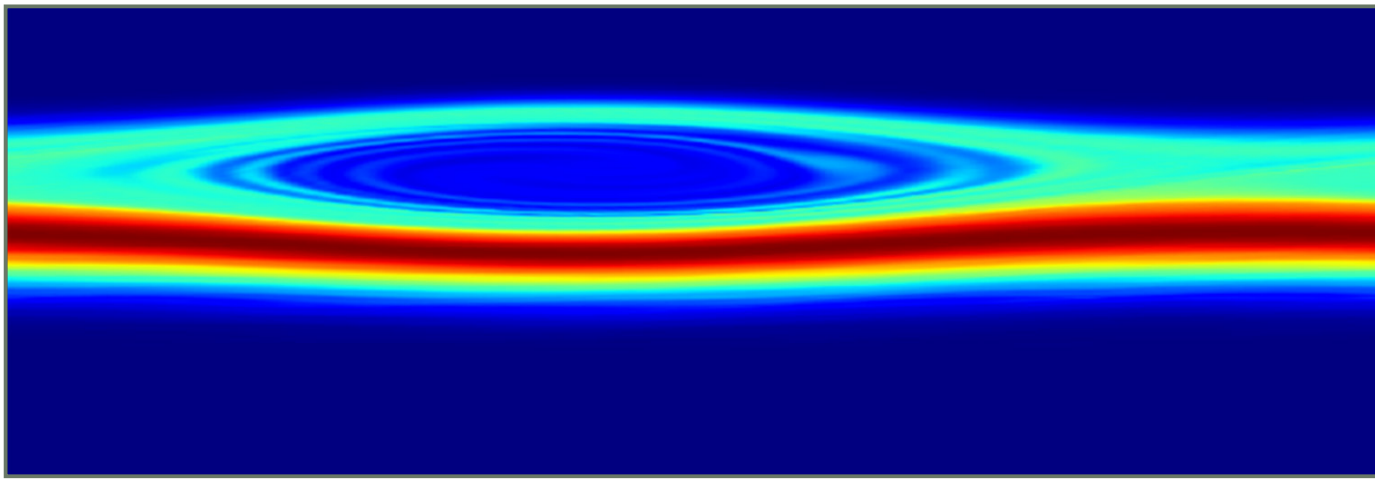

Figure 3-1: The bump-on-tail instability leads to a travelling vortex in phase space. The horizontal axis represents space and the vertical one velocity.

The Vlasov equation describes the evolution of an ensemble of particles, more properly speaking their density distribution, subject to the laws of classical mechanics. If this equation is coupled to the electromagnetic field, a highly non-linear system of equations results, which is referred to as the Vlasov-Maxwell equations. In the case that magnetic effects are negligible a simplified model, the Vlasov-Poisson equations, can be used instead. These two equations represent the most fundamental description of a collisionless (classical) plasma. Their efficient numerical solution is thus of paramount importance in simulations ranging from controlled fusion to laser-plasma interactions.

Mathematical formulation

The Vlasov-Maxwell or Vlasov-Poisson equations are formulated mathematically as non-linear partial differential equations which are coupled by the charge density (in the case of the Vlasov-Poisson equations) and the current (in case of the Vlasov-Maxwell equations). However, contrary to fluid models, the phase space is up to six dimensional as the position and the velocity of the particles have to be resolved. Even if simplified models are considered, as is often possible in applications, at least two dimensions in phase space (one space and one velocity direction) have to be resolved for the Vlasov-Poisson equations. Since there is no magnetism in a one dimensional system, we require at least one dimension in position and two in velocity for the Vlasov-Maxwell equations.

Due to the high dimensional phase space and the limited amount of memory available on previous generation computer systems, historically particle methods have been the methods of choice for integrating the Vlasov equation. A particle method proceeds by advancing a large number of particles, say one million, in phase space; to compute the result of the simulation and to calculate the electromagnetic fields the particle density is then reconstructed from this limited number of particles. In fact, in realistic plasma system used for controlled fusion, for example, the number of particles is on the order of 1015 per cubic centimeter. Nevertheless, simulations conducted with such particle methods were able to provide much insight into many areas of plasma physics. However, they also exhibit serious disadvantageous that render them ill-suited for many applications. For example, the tail of the density distribution suffers from numerical noise that only decays as the square root of the number of particles.

Computational difficulties

Since particle methods suffer from a number of shortcomings, interest in grid based methods started, i.e. methods where a static grid in both position and velocity is used as the computational domain, as soon as sufficient computational resources became available.

However, the numerical solution of the Vlasov equation using grid based methods does present significant challenges which limit their widespread adoption:

- The high dimensionality implies that, in the worst case, the amount of memory needed to store the simulation scales as the sixth power in the number of grid points. If a naive approximation scheme is used, the desired accuracy is usually not achievable; not even on modern supercomputers. To mitigate this situation a high order approximation in space is essential to get sufficiently accurate results.
- The Vlasov equation is, as the semidiscretization of a partial differential equation, a stiff system and thus the step size of the computationally advantageous explicit schemes (such as Runge-Kutta methods) are limited by the CFL condition. Thus, it is important to develop methods that are computationally attractive and allow us to take larger steps in time (as compared to the CFL condition). Splitting methods provide a promising approach in that direction.
- Due to the coupling of the Vlasov equation to the Maxwell/Poisson equation the system is non-linear. Even though this nonlinearity is only quadratic, examples from ordinary differential equations, such as

the Lorenz system, show that such systems can exhibit very surprising and interesting physical phenomena. This is true for the Vlasov-Poisson and the Vlasov-Maxwell equations as well. In this case only a small deviation from an equilibrium distribution can result in a rich structure in phase space. For example, for an only slightly modulated Gaussian initial distribution an exponential decay can be observed in the electric energy (the famous Landau damping). In the non-linear regime it is impossible to study this behavior in an analytical framework by employing Fourier methods, for example; thus, numerical simulations are the only way to obtain theoretical insight into such systems.

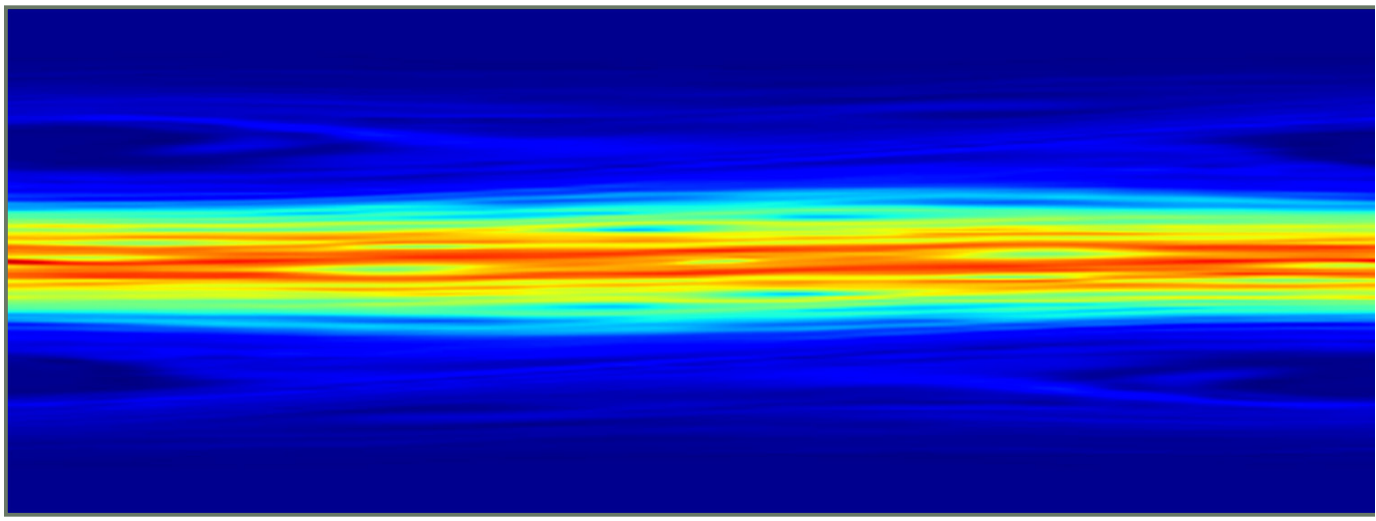

Figure 3-2: Simulation of the filamentation of phase space as is studied, for example, in the context of Landau damping. The horizontal axis represents space and the vertical one velocity.

Furthermore, the appearance of smaller and smaller scales in phase space (a phenomenon called filamentation) further increases the difficulty in computing a reliable numerical solution. In addition, many mathematical properties, such as energy conservation of the numerical methods employed, are not very well understood. A more thorough understanding could contribute to the design of more efficient schemes in the future.

In summary, numerical simulations of the Vlasov-Poisson and Vlasov-Maxwell equations require, except for the simplest examples, the use of substantial computational resources. Those difficulties have to be alleviated by parallelizing algorithms to larger clusters, by exploiting modern hardware systems such as GPUs, as well as by improving the space and time discretization algorithms employed in such simulations.

Results & Outlook

It has been demonstrated that a discontinuous Galerkin approximation in space combined with high order splitting methods in time leads to schemes which exhibit good results with respect to both accuracy and efficiency. This can be mathematically verified and in fact high order splitting schemes for the Vlasov-Poisson equations have been employed successfully in the literature. However, due to the more complicated structure of the Vlasov-Maxwell equations the second order Strang splitting scheme is used almost exclusively in the literature. Thus, one of our goals is to construct high order methods that are applicable to the Vlasov-Maxwell equations as well. In pursuing that goal it was shown that the numerical techniques developed can be applied to equations that have a similar mathematical structure but model such diverse physical systems as charged particles, relativistic computations of planetary motion, or the simulation of spatially varying chemical reactions.

Funding

- FWF P25346 “Splitting Verfahren für Vlasov-Poisson und Vlasov-Maxwell Gleichungen”

Affiliations

- Research focal point Scientific Computing
- Department of Mathematics

References

- L. Einkemmer, A. Ostermann, An almost symmetric Strang splitting scheme for the construction of high order composition methods, arXiv:1306.1169
- L. Einkemmer, A. Ostermann, Convergence analysis of a discontinuous Galerkin/Strang splitting approximation for the Vlasov-Poisson equations, arXiv:1211.2353
- L. Einkemmer, A. Ostermann, Convergence analysis of Strang splitting for Vlasov-type equations, arXiv:1207.2090

Contact

Dipl.Ing. Lukas Einkemmer
Department of Mathematics
Technikerstraße 19a
A-6020 Innsbruck

Phone
+43 512 507 53802

Email
lukas.einkemmer@uibk.ac.at

Webpage
http://www.uibk.ac.at/mathematik