QUASI-EXPLICIT, UNCONDITIONALLY STABLE, DISCONTINUOUS GALERKIN SOLVERS FOR CONSERVATION LAWS

PHILIPPE HELLUY[1,2,3], PIERRE GERHARD[1,2,3], VICTOR MICHEL-DANSAC[3], BRUNO WEBER[4]

Abstract. We have developed in a previous work [4] a parallel and quasi-explicit Discontinuous Galerkin (DG) kinetic scheme for solving hyperbolic systems of conservation laws. The solver is unconditionally stable (i.e., the CFL number can be arbitrary), has the complexity of an explicit scheme. It can be applied to any hyperbolic system of balance laws. In this work, we improve the parallel scaling of the method thanks to an implicit-explicit subdomain decomposition strategy.

1. Kinetic approximation of first order conservations laws

In this work, we are interested in the numerical approximation of a hyperbolic system of m conservation laws in dimension d

$$\partial_t W + \sum_{i=1}^{d} \partial_i Q^i(W) = 0, \tag{1}$$

where the unknown is a vector $W(X,t) \in \mathbb{R}^m$ depending on the space variable: $X = (x_1 \dots x_d)$ and the time variable: t. For the partial derivatives, we use the notation $\partial_i = \frac{\partial}{\partial x_i}, \quad \partial_t = \frac{\partial}{\partial t}$.

This kind of system is generally difficult to approximate numerically. One of the difficulties is that explicit schemes are subject to restrictive time steps conditions. Implicit schemes do not suffer from time step conditions but require solving large sets of linear equations. In previous works (see [4] and included references), we have proposed a method, based on a kinetic approach, for avoiding this constraint. We first recall the principles of the kinetic representation.

We consider a set of $d+1$ (or more) kinetic velocities V_k, $k = 0 \dots d$, associated to **vectorial** kinetic functions $F_k(W) \in \mathbb{R}^m$. We also define "Maxwellian" equilibrium functions $M_k(W) \in \mathbb{R}^m$. The kinetic BGK representation is given by transport equations with relaxation source terms [2, 1]

$$\partial_t F_k + V_k \cdot \nabla_X F_k = \frac{1}{\tau} \left(M_k(W) - F_k \right). \tag{2}$$

When the relaxation time $\tau \to 0^+$, the kinetic model (2) is formally equivalent to the initial system of conservation laws (1) provided that

$$W = \sum_k M_k(W), \quad \sum_k V_k^i M_k(W) = Q^i(W), \quad i = 1, \dots, d. \tag{3}$$

Conditions (3) constitute a set of $m(d+1)$ equations with $m(d+1)$ unknowns for finding the Maxwellian. It possesses a unique solution. Theoretical arguments show that the formal limit is a true limit, under a so-called sub-characteristic condition [2, 1]. This condition states that the kinetic velocities have to be greater than the wave speeds λ_r of the underlying hyperbolic system: $\forall k, \quad |V_k| > \max_r |\lambda_r|$.

In practice it is difficult to solve directly the BGK system (2). It is better to split the equations into transport and a collision steps. This leads to the following kinetic algorithm for advancing one time step:

(1) Solve, for a duration Δt , the free transport equation

$$\partial_t F_k + V_k \cdot \nabla_X F_k = 0.$$

(2) Solve, for the same duration, the relaxation (or collision) step

$$\partial_t F_k = \frac{1}{\tau} \left(M_k(W) - F_k \right).$$

This algorithm is iterated in order to compute an approximation of $W = \sum_k F_k$. The presented splitting algorithm is only first order accurate in time. But it is possible to improve its order, for instance by using an over-relaxation algorithm [4]. It has been observed, since a long time that these kinds of kinetic schemes are free of CFL conditions. See for instance [3]. However, this interesting property is rarely exploited in practical applications.

This work was supported by: IRMIA++ `https://irmiapp.unistra.fr/` and France Relance. We also thank the Mathematisches Forschungsinstitut of Oberwolfach where it was first presented.

2. Unconditionally stable DG approximations

The kinetic algorithm presented in Section 1 relies on transport steps and relaxation steps. The relaxation step is generally easy to implement at each interpolation point of the approximation. In addition, it is embarrassingly parallel. The most complicated part of the kinetic algorithm requires solving transport equations of the form

$$\partial_t f + V \cdot \nabla f = 0. \tag{4}$$

If the computational domain has a simple shape and if the solution is computed on a structured Cartesian grid, it is natural to solve this transport equation by the characteristic method. With well-chosen time step Δt and kinetic velocities V_k, this approach leads to the so-called Lattice Boltzmann method.

In a domain Ω with a complex geometry and for unstructured grid, the characteristic method is no more a good choice because it leads to difficulties such as instabilities or loss of conservation. In addition, the treatment of boundary conditions is not natural in this framework. In the unstructured case, we prefer to rely on a DG approximation of (4).

We consider an unstructured mesh of the computational domain Ω made of tetrahedral cells. The transported function f is approximated in cell L by a linear expansion on basis functions $f(x,t) \simeq f_L(x,t) = \sum_j f_{L,j}(t)\psi_j^L(x), \quad x \in L$. The unknowns are the coefficients $f_{L,j}(t)$ of the linear expansion. After a DG in space approximation of (4), the DG scheme read as follows [4]

$$\mathbb{K}\mathbb{F}'(t) = 0, \tag{5}$$

where $\mathbb{F}(t)$ is a large vector containing all the coefficients $f_{L,j}(t)$, and $\mathbb{K}$ is the large matrix arising from the DG approximation of the transport equation. We then have to solve a large set of linear Ordinary Differential Equations (ODE).

Explicit-in-time approximations of this set of ODEs suffer from constraining stability conditions on the time step Δt. In order to suppress the stability condition, we can use an implicit time scheme for solving (5) for going from time step $n-1$ to time step n. For simplicity, we describe the case of an implicit first order Euler method. The strategy can be extended to other more accurate schemes, such as the Crank-Nicolson scheme (that we use) or DIRK (Diagonally implicit Runge--Kutta) approaches. With $\mathbb{F}^n \simeq \mathbb{F}(n\Delta t)$, the implicit Euler scheme reads

$$(\mathbb{I} + \Delta t \mathbb{K})\mathbb{F}^{n+1} = \mathbb{F}^n \tag{6}$$

It seems that one would need to assembly and solve a large linear system for computing $\mathbb{F}^{n+1}$ from $\mathbb{F}^n$. But when the numerical flux of the DG solver is the upwind flux, then the matrix $\mathbb{K}$ is block-triangular. In practice there is thus an explicit algorithm, the downwind algorithm, for solving the system (6) efficiently. See [4].

3. Subdomain parallelism

We have implemented the downwind and the kinetic algorithms in a parallel software based on a work stealing strategy [4]. We have observed a decreasing efficiency of the method when the number of threads increases. This is because the parallel scaling of the downwind algorithm is limited, at a given point, by the dependencies in the computations.

In order to increase the parallel scaling, we now describe a subdomain strategy that relaxes the computational dependencies. The main idea is to apply the above time-implicit downwind algorithm in each subdomain, but with a time-explicit coupling between the subdomains, for suppressing some dependencies. Because of the explicit coupling, it will be necessary to apply an iterative algorithm for computing the exact solution in a stable way. The algorithm can be proved to converge in a finite number of iterations. In most configurations three iterations are sufficient. Let us now describe the principles of this subdomain iterative algorithm. As in Section 2, the main task is the resolution of the transport equation in $\Omega \times [0, \Delta t]$, with initial data:

$$\partial_t f + V \cdot \nabla f = 0, \quad f(X,0) = f^0(X).$$

We assume that Ω is decomposed into a finite number of subdomains Ω_i, $i = 1 \dots n_d$. For the simplicity of the presentation, we assume that Ω is a periodic domain or the whole space, in order to avoid the description of the boundary conditions. However the approach is also valid when $\partial\Omega \neq \emptyset$.

We then denote by f_i the restriction of f to subdomain Ω_i, by $N_i(X)$ the outward normal vector on $\partial\Omega_i$, and by $\partial\Omega_i^-$ the upwind part of the boundary of Ω_i defined by

$$\partial\Omega_i^- = \{X \in \partial\Omega_i, N_i(X) \cdot V < 0\}.$$

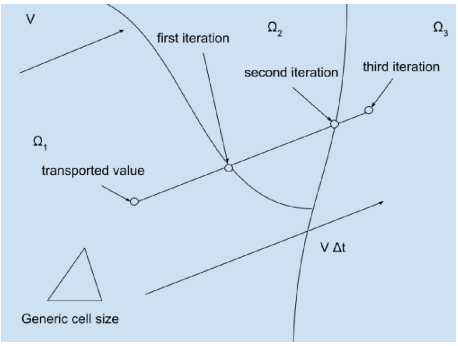

FIGURE 1. Subdomain algorithm, in a generic subdomain decomposition, with corners shared by several subdomains. In this case, the iterative algorithm reaches the exact solution in at most three iterations. First iteration: the boundary value on $\partial\Omega_2^-$ is updated. Second iteration: the boundary value on $\partial\Omega_3^-$ is updated. Third iteration: the correct value is transported.

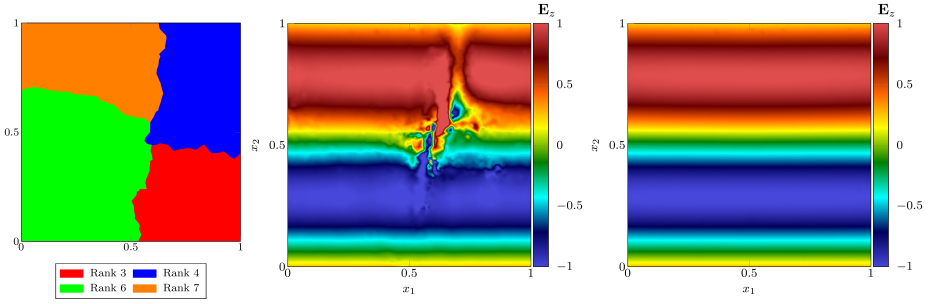

FIGURE 2. Stability of the subdomain iterative algorithm. Left: subdomains structure, Middle: 2 iterations scheme, Right: 3 iterations scheme. We observe that the iterative algorithm is stable, even with large time step, but that three iterations seem to be necessary.

We initialize the algorithm by setting $f_i^0(X,t) = f_i^0(X)$. Thus, the initial iteration does not depend on time. We then consider an iterative algorithm for computing successive time-dependent approximations f_i^p of f_i in subdomain Ω_i for $p \geq 1$. For computing f_i^p from f_i^{p-1} we solve the following time-dependent boundary value problem

$$\partial_t f_i^p + V \cdot \nabla f_i^p = 0, \quad \text{in } \Omega_i, \tag{7}$$

$$f_i^p(X,0) = f_i^0(X), \quad X \in \Omega_i, \tag{8}$$

$$f_i^p(X,t) = f_j^{p-1}(X,t), \quad X \in \partial\Omega_i^- \cap \partial\Omega_j. \tag{9}$$

In other words, the f_i^p in subdomain Ω_i are computed from the f_j^{p-1} in the neighboring subdomains Ω_j. We can prove the following result.

Proposition *let L be the diameter of the smallest subdomain. Under the condition*

$$\Delta t \leq \frac{L}{|V|},$$

in the generic case, the above algorithm (7)-(9) converges to the exact solution in at most three iterations: $f_i^3 = f_i$.

The proof relies on the characteristic method. It is briefly sketched in Figure 1.

3.1. **Stability.** We have implemented the above iterative algorithm. The parallelism within each subdomain is managed, as before, through the work stealing strategy. The communications between the subdomains are managed through calls to the MPI (Message Passing Interface) library. In our first experiments, we have verified the stability properties of the transport solver. They indicate that the number of iterations of the iterative algorithm is indeed important for the stability of the method. For a general domain decomposition obtained with an automatic partitioner, and with large time steps, the algorithm is stable provided that three iterations are done for advancing one time step. An illustration is given in Figure 2.

The objective of the subdomain algorithm was to relax the computational dependencies and to achieve a better parallel (strong) scaling of the method.

We compare the time spent in the iterative algorithm with a varying number of threads and subdomains. We define the efficiency e of the acceleration as the ratio of the elapsed time of the algorithm with the time that we would get with an ideal perfect scaling. The efficiency is perfect if $e = 1$. We observe, for instance, that with a single subdomain the efficiency with 64 threads

MPI nodes	Threads	#CPU	Time (s)	Accel. e
1	2	2	1314	1
1	8	8	346	0.95
1	**64**	**64**	**106**	**0.39**
2	32	64	75	0.55
8	**8**	**64**	**57**	**0.72**

TABLE 1. Multithread and MPI scaling. For a computation done with 64 threads, we observe that it is better to split the domain into 8 subdomains instead of affecting all the threads to one single subdomain.

drops to $e = 0.39$, while with 8 subdomains and 8 threads per subdomain the efficiency is better $e = 0.72$. We have thus validated the efficiency of this approach. Of course, the whole algorithm is impacted by a slowdown factor imposed by the additional iterations. However the weak scaling of the method on a supercomputer for very large computations is now certainly ensured. Indeed, explicit subdomain decomposition methods are known to be well adapted to the architecture of supercomputers. More measurements are given in Table 1.

4. Conclusion

We presented an adaptation of the kinetic DG method introduced in [4]. The method can handle arbitrary conservation laws and complex unstructured meshes. It is explicit in time but CFL-free. The method has good parallelization features, for both shared memory and distributed memory computers. For improving the parallel scaling on distributed memory computers, we have proposed a subdomain decomposition method that relaxes the task dependencies of the kinetic scheme but keeps the possibility to use large time steps.

References

[1] Denise Aregba-Driollet and Roberto Natalini. Discrete kinetic schemes for multidimensional systems of conservation laws. *SIAM Journal on Numerical Analysis*, 37(6):1973–2004, 2000.

[2] François Bouchut. Construction of BGK models with a family of kinetic entropies for a given system of conservation laws. *Journal of Statistical Physics*, 95(1-2):113–170, 1999.

[3] Yann Brenier. Averaged multivalued solutions for scalar conservation laws. *SIAM journal on numerical analysis*, 21(6):1013–1037, 1984.

[4] Pierre Gerhard, Philippe Helluy, and Victor Michel-Dansac. Unconditionally stable and parallel discontinuous galerkin solver. *Computers & Mathematics with Applications*, 112:116–137, 2022.

[1]Université de Strasbourg, [2]IRMA UMR CNRS 7501, [3]Inria Tonus, [4]AxesSim Illkirch