A HERMITE TYPE ADAPTIVE SEMI-LAGRANGIAN SCHEME

Michel Mehrenberger [1] and Eric Violard [2]

Abstract. We study a new Hermite type interpolating operator arising in a semi-lagrangian scheme for solving the Vlasov equation. Numerical results on uniform and adaptive grid are shown and compared with biquadratic interpolation introduced in [CamMe04] in the case of a rotating Gaussian.

1. Introduction

Adaptive semi-lagrangian schemes for solving the Vlasov equation in the phase space have recently been developped: wavelet techniques [Gut-al04, Gut-al05], moving mesh method [Son-al04], hierarchical finite element decomposition [CamMe04, CamMe05]. One main advantage of the latter method, is that the underlying dyadic partition of cells allows an efficient parallelization. It has been implemented with a biquadratic Lagrange interpolation. But the use of higher order methods is not straightforward in that context. The same problem in fact occurs in the case of semi-Lagrangian schemes on unstructured grids. One solution was there to use an Hermite type interpolation. We propose here to do the same in the adaptive context. Thanks to a well chosen Hermite interpolation recently found [HonSc04], we thus obtain a more accurate scheme.

The paper is organized as follows. Section 2 presents the two interpolating operators that we designed for our numerical scheme. First, we recall the Lagrange operator and then we present the Hermite new one. Section 3 briefly recalls our uniform and adaptive semi-lagrangian schemes. Section 4 focusses on the crucial point of the computational cost of the two operators and we give efficient algorithms to compute the interpolated value as a sequence of assignments. Section 5 completes the definition of our adaptive scheme. It gives for each operators

Key words and phrases. Adaptive method, Vlasov equation, Numerical simulation, Hermite operator.

[1]Corresponding author: INRIA Rocquencourt Domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex, France, email: Michel.Mehrenberger@inria.fr

[2]Laboratoire ICPS - LSIIT (CNRS UMR-7005), Université Louis Pasteur, Pôle API Boulevard Sébastien Brant, 67400 Illkirch, France, email: violard@icps.u-strasbg.fr

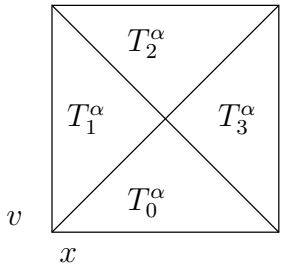

FIGURE 1. The 4 triangles of a cell α

a criteria for compressing cells of the dyadic mesh. Last, section 6 shows our experimental results before concluding.

2. Local interpolating operators

Notations. We use the square $\Omega = [0,1[^2$ as a computational domain. It is decomposed into a partition $\mathcal{M}$ of square shaped cells $\alpha = [k2^{-j},(k+1)2^{-j}[\times[\ell 2^{-j},(\ell+1)2^{-j}[$, where k,ℓ and j are integers, and j denotes the level of the cell.

For a point $(x,v) \in \Omega$, we can thus define a unique cell $\alpha_{x,v} \in \mathcal{M}$ such that $(x,v) \in \alpha_{x,v}$.

Given a cell α, we denote by $(0,0)_\alpha$, $(1,0)_\alpha$, $(1,1)_\alpha$, $(0,1)_\alpha$ its four corners, and more generally, $(\lambda,\mu)_\alpha$ will be the point whose local coordinates in α are $\lambda,\mu \in [0,1]^2$. Let T_k^α, $k = 0,\dots,3$ be the 4 triangles obtained by subdivising the cell α with the diagonals (Fig.1).

For $d \in \mathbb{N}$, we classically define

$$Q_d = \{\sum a_{i,j}x^i v^j, \quad i,j \le d\}, \quad P_d = \{\sum a_{i,j}x^i v^j, \quad i+j \le d\}.$$

Now, let f be a function defined on Ω prolongated on $\mathbb{R}^2$ by zero (we can similarly prolongate it by periodicity), and $(x,v) \in \Omega$.

Biquadratic Lagrange interpolation. $P_L f(x,v)$ is defined as the unique element of Q_2 on the cell $\alpha = \alpha_{x,v}$ such that it coincides with f on the 9 equireparted nodes:

$$P_L f((p,q)_\alpha) = f((p,q)_\alpha) \quad p,q = 0,1/2,1.$$

Hermite interpolation. Assuming that f is derivable on each cell α. $P_H f(x,v)$ is then the unique C^1 spline on the cell $\alpha = \alpha_{x,v}$, P_3 on the triangles T_k^α, $k = 0,\dots 3$ such that there is coincidence on the 12 degrees of freedom at the corners:

$$f(a), \partial_x f(a), \partial_v f(a), \quad a = (p,q)_\alpha \quad p = 0,1, \quad q = 0,1.$$

and also on the 4 normal derivatives on the edges i.e.:

$$\partial_v f((1/2,0)_\alpha), \partial_x f((1,1/2)_\alpha), \partial_v f((1/2,1)_\alpha), \partial_x f((0,1/2)_\alpha).$$

3. A general iterative scheme

A semi-Lagrangian scheme takes the form of a succession of interpolation and transport steps.

We define classically the characteristic curves:

$$Z(t;s,x,v) = (X(t;s,x,v), V(t;s,x,v)),$$

satisfying $Z(s;s,x,v) = (x,v)$, and we consider a function $f = f(t,x,v)$ that we want to approximate, which is constant along these characteristics:

$$f(t,x,v) = f(t, Z(t;t,x,v)) = f(s, Z(s;t,x,v)) = f_0(Z(0;t,x,v)),$$

where f_0 is a given initial condition defined in Ω and completed by zero outside of Ω.

Let $\Delta t > 0$ be the time step, and $\mathcal{T}^n$ be the exact backward transport operator at iteration step n which is defined by:

$$\mathcal{T}^n(x,v) = Z((n-1)\Delta t; n\Delta t, x, v)$$

Since f is constant along the characteristics, we have

$$f((n+1)\Delta t, x, v) = f(n\Delta t, T^{n+1}(x,v)) = f_0(Z(0;(n+1)\Delta t, x, v)).$$

We focus here on the errors produced by the interpolating process and we will consider that the exact transport operator (and thus also the exact solution) is known. In the general case, we should use an approximation of the exact transport operator.

The iterative scheme consists then in finding the degrees of freedom at each iteration step n, which gives a representation f^n, completed by zero outside of Ω. We fix a resolution level $J \in \mathbb{N}^*$.

Uniform scheme. We consider the uniform grid $\mathcal{M}_J$ of 2^{2J} cells.

- (Iteration step $n = 0$) We compute the degrees of freedom from f_0, on the corresponding grid which gives a representation f^0 at iteration step 0.
- (Iteration step $n+1$) For each point (x,v) corresponding to a degree of freedom, we compute the backward advected point $a = \mathcal{T}^{n+1}(x,v)$. The new value is thus $f^n(a)$ (or $\frac{\mathrm{d}}{\mathrm{dz}} f^n(a)$, with $z = x$ or v). We thus have a representation f^{n+1} at iteration step $n+1$.

Adaptive scheme. We will use a compression step $(\tilde{f}, \tilde{\mathcal{M}}) = \mathcal{C}(f, \mathcal{M})$. From a representation of a function f on a mesh $\mathcal{M}$, we will derive a new coarser representation $\tilde{f}$ on a mesh $\tilde{\mathcal{M}}$. This can be done locally, by comparing the representation of the current function on 4 "daughters" cells, with the interpolated function on the "mother" cell, whose 4 "daughters" cells form a partition. If the two representation are not far, we will keep the coarser representation on the mother cell. Specific

tests, for the biquadratic Lagrange interpolation, and for the Hermite interpolation will be specified.

We also define a prediction step $\tilde{\mathcal{M}}_n = \mathcal{T}^{n+1}(\mathcal{M}_n)$, from a mesh $\mathcal{M}_n$ and the backward advection operator $\mathcal{T}^{n+1}$, we compute a new mesh $\tilde{\mathcal{M}}_n$. This is performed by beginning with a coarse mesh and recursively refining each cell of level $j \leq J$, if the backward advected center of the cell falls on a cell of $\mathcal{M}^n$ whose level is strictly smaller than j.

The algorithm then reads:

- (Iteration step $n = 0$) From f_u^0 obtained by the uniform algorithm, we compress the solution and obtain $(f^0, \mathcal{M}^0) = \mathcal{C}(\mathcal{M}_J, f_u^0)$.
- (Iteration step $n + 1$) We predict a first grid $\mathcal{T}^{n+1}(\mathcal{M}^n)$ from $\mathcal{M}^n$, compute f_1^{n+1} as in the uniform algorithm (replacing $\mathcal{M}_J$ with $\mathcal{T}^{n+1}(\mathcal{M}^n)$) and next compute the representation of f on $\mathcal{M}^{n+1}$, $(f^{n+1}, \mathcal{M}^{n+1}) = \mathcal{C}(\mathcal{T}^{n+1}(\mathcal{M}^n), f_1^{n+1})$.

4. Fast formulae

Our formulae are defined on any square cell $\alpha = [a, a+h] \times [b, b+h]$, i.e., for $a \leq x \leq a + h$ and $b \leq v \leq b + h$.

Biquadratic Lagrange interpolation. Given 9 numbers $(g_{i,j})_{i,j=0}^2$, the function $g(x, v)$ of degree ≤ 2 that satisfies

$$g(a + ih/2, b + jh/2) = g_{i,j}, \quad \text{for } i, j = 0, 1, 2,$$

is uniquely determined. We can compute it as follows. We first set $N = 1/h$ (which can be precomputed) and $t_0 = (x-a)N, \quad t_1 = 2t_0-1$, which gives

$$\begin{aligned} h_0 &= g_{1,j} + t_1(g_{1,j} - g_{0,j}) \\ h_1 &= g_{1,j} + t_1(g_{2,j} - g_{1,j}) \\ g_j &= h_0 + t_0(h_1 - h_0), \quad \text{for } j = 0, 1, 2. \end{aligned}$$

We next set $t_0 = (v - b)N, \quad t_1 = 2t_0 - 1$, so that we have

$$\begin{aligned} h_0 &= g_1 + t_1(g_1 - g_0) \\ h_1 &= g_1 + t_1(g_2 - g_1) \\ g(x, v) &= h_0 + t_0(h_1 - h_0). \end{aligned}$$

This procedure thus needs only 10 assignments, 16 multiplications and 28 additions (or substractions).

Hermite interpolation. Given 16 numbers $g_{i,j}, g_{i,j}^x, g_{i,j}^v, i, j = 0, 1$, g_0^x, g_1^x, g_0^v and g_1^v, the C^1cubic spline, P_3 on the 4 triangles $T_0^\alpha, T_1^\alpha, T_2^\alpha$

and T_3^α, satisfying

$$\begin{aligned}
&g(a+ih, b+jh) = g_{i,j},\\
&\partial_x g(a+ih, b+jh) = g^x_{i,j},\\
&\partial_v g(a+ih, b+jh) = g^v_{i,j}, \qquad \text{for } i,j = 0,1 \quad \text{and}\\
&\partial_x g(a+ih, b) = g^x_i,\\
&\partial_v g(a, b+ih) = g^v_i, \qquad \text{for } i = 0,1,
\end{aligned}$$

is uniquely determined.

We can compute it as follows on the triangle T_0^α (formulae on the other triangles can be similarly derived). By setting $u = (x-a)N$ and $y = (v-b)N$, we obtain

$$\begin{aligned}
g(x,v) &= g_{00} + (2u^3 - 3u^2)(g_{00} - g_{10}) + (y^3 - 3y^2 + 3uy^2)(g_{00} - g_{01})\\
&+ (y^3 - 3uy^2)(g_{10} - g_{11}) + h((2/3y^3 - 2u^2 - 3/2y^2 + u + 3/2uy^2 + u^3)g^u_{00}\\
&+(-1/2y^2+uy^2+y-3uy+1/6y^3+2u^2y)g^y_{00}+(2/3y^3-2y^2+4(-u^2+u)y)g^y_0\\
&+(-2/3y^3 - u^2 + 3/2uy^2 + u^3)g^u_{10} + (1/2y^2 - uy^2 - uy + 1/6y^3 + 2u^2y)g^y_{10}\\
&+(4/3y^3 - 2uy^2)g^u_1 + (-2/3y^3 + 1/2uy^2)g^u_{11} + (5/6y^3 - uy^2)g^y_{11} - 2/3y^3 g^y_1\\
&+ (2/3y^3 + 1/2(-y^2 + uy^2))g^u_{01} + (5/6y^3 - y^2 + uy^2)g^y_{01}\\
&+ (-4/3y^3 + 2(y^2 - uy^2))g^u_0)
\end{aligned}$$

We used the 'optimize' function of the 'codegen' package of Maple to lower the cost of these interpolation operators and implement them in our code. We applied these optimizations in the practical case where $a, b = 0$ and h is the size of a mesh cell.

For example, in our code, the cost of the computation of $g(x,v)$ on triangle T_0^α by the Hermite operator is 18 assignments, 49 multiplications and 53 additions. It is obtained by introducing some auxilary variables as follows:

$$\begin{aligned}
&x_1 = u^2, x_2 = uu_1, v_1 = y^2, v_2 = 3v_1, v_3 = uv_2,\\
&v_4 = yv_1, v_5 = 2/3v_4, v_6 = 4/3v_4, v_7 = 5/6v_4, v_8 = uv_1,\\
&x_3 = ux_2, x_4 = x_3 + 3/2v_8, v_9 = 1/6v_4 + 2x_1y,
\end{aligned}$$

Moreover, for the Hermite operator, instead of computing $g(x,v)$, $\partial_x g(x,v)$ and $\partial_v g(x,v)$ separately, we compute them together which lowers the number of required elementary operations and reduces the computation cost of a minimum of 10% for most processor architectures.

5. Compression formulae

Hermite compression. In the case of the Hermite interpolation, the compression test used for the 4 daughter cells of a given cell α of

length h is:

$$d_0^x + d_1^x + d_0^v + d_1^v \leq \varepsilon,$$

with $d_k^x = \left| f((1/2,k)_\alpha) - \tilde{f}_{1/2k} \right|$, where

$$\tilde{f}_{1/2k} = ((f((0,k)_\alpha) + f((1,k)_\alpha))/2 + h/8(\partial_x f((0,k)_\alpha) + \partial_x f((1,k)_\alpha)),$$

and $d_k^v = \left| f((k,1/2)_\alpha - \tilde{f}_{k1/2}) \right|$, where

$$\tilde{f}_{1/2k} = ((f((k,0)_\alpha) + f((k,1)_\alpha))/2 + h/8(\partial_v f((k,0)_\alpha) + \partial_v f((k,1)_\alpha)),$$

for $k = 0, 1$. Note that the value f is the reconstructed value at the middle of the edge, with the 1D Hermite interpolation operator.

Biquadratic Lagrange compression. For a given mother cell α, the compression test used is

$$\sum_{p,q=0}^{4} \left| f((p/4,q/4)_\alpha) - \tilde{f}_{pq} \right| \leq \varepsilon,$$

where $\tilde{f}_{pq}$ is the value obtained by interpolation on the cell α, at the point $(p/4, q/4)_\alpha$.

6. Numerical results

We take for initial data

$$f_0(x,v) = \exp(-0.07((40(x-0.5)+4.8)^2 + (40(v-0.5)+4.8)^2)),$$

and $\Delta t = 0.19635$. The transport operator is here given by a rotation of angle Δt around the center $(0.5, 0.5)$.

We implemented our schemes in C++ and carried out our code on a Pentium4 processor cadenced at 3.06 GHz with 512Mo of RAM. We considered our adaptive scheme for $\epsilon = 10^{-4}$, 10^{-5}, 10^{-6}, 10^{-7} and 10^{-8}. The error is computed on a uniform grid of 256×256 points.

Figure 2 shows the average absolute error (norm L1) as a function of the number of mesh cells for the different schemes after 99 iteration steps. We observe that for any given error threshold, any ϵ and any interpolation operator, the number of cells is always lower with the adaptive scheme than with the uniform one. But that does not necessarly mean that the execution time is always lower with the adaptive scheme than with the uniform one as illustrated on Fig. 3.

Figure 3 plots the average absolute error as a function of the average time to compute one step (in seconds). We notice that, for the Hermite operator, the performance of the adaptive scheme is always better than the performance of the uniform scheme. But this is not the case for the Lagrange operator.

Figure 4 shows the error as a function of the number of steps for each of the two operators. We observe that the Hermite operator for

$J = 9$ and $\varepsilon = 10^{-8}$ has a little better accuracy than the uniform Lagrange operator for $J = 10$. Moreover we notice that the amplitude of the error oscillation is lower for the Hermite operator than for the Lagrange operator. The executive time is also better: 0.419 seconds per iteration step with 86092 cells for the Hermite operator and 0.646 seconds with 1048576 cells for the uniform Lagrange operator.

7. Conclusion

In this paper, we defined an Hermite interpolating operator which can be used in both our uniform and adaptive schemes for solving the Vlasov equation. The experimental results show that this operator presents great advantages in comparison with the Lagrange operator which was originally used.

Our first goal was to demonstrate the feasibility of developing a Vlasov solver based on the Hermite operator. Therefore we defined it for a 2D phase space and considered a test case for which the analytic solution was known. Our next goal is to run more realistic test cases and extend our approach to 4D.

In a physical test case, the backward operator is computed at each step from the electric field. Since this computation cost is proportional to the number of cells, it means that it is even more advantageous to use the Hermite operator instead of the Lagrange one as the results in this paper show it.

Further work includes parallelizing our sequential code. We plan to reuse some parallelization techniques which resulted in good speed-up onto distributed memory parallel machines for a similar numerical scheme but Lagrange operator [Hoe-al04]. In particular, we designed a specific data structure [HoeVi06] which is suitable to exploit data locality coming from the local nature of these schemes and which can be advantageously reused to design code for shared memory parallel machines using OpenMP directives.

References

[CamMe04] M. Campos Pinto and M. Mehrenberger (2005) : *Adaptive numerical resolution of the Vlasov equation,* Numerical Methods for Hyperbolic and Kinetic Problems, CEMRACS 2003/ IRMA Lectures in Mathematics and Theoretical Physics 7, pp. 43–58.

[CamMe05] M. Campos Pinto and Michel Mehrenberger (2005) : Convergence of an adaptive scheme for the one-dimensional Vlasov- Poisson system, Technical report, INRIA Lorraine, No RR-5519.

[Gut-al04] M. Gutnic, M. Haefele, I. Paun, E. Sonnendrücker (2004) : *Vlasov simulations on an adaptive phase-space grid*, Comput. Phys Commun. Vol.164, pp. 214–219.

[Gut-al05] M. Gutnic, M. Haefele and G. Latu (2005) : *A Parallel Vlasov solver using a Wavelet based Adaptive Mesh Refinement*, in: 2005 International Conference on Parallel Processing (ICPP'2005), 7th Workshop on High Perf. Scientific and Engineering Computing, IEEE Computer Society Press, pp. 181–188.

[Hoe-al04] O. Hoenen, M. Mehrenberger and E. Violard (2004) : *Parallelization of an Adaptive Vlasov Solver*, 11th European PVM/MPI Users' Group Conference (EuroPVM/MPI 2004), ParSim Session, Lecture Notes in Computer Science, vol. 3241, Springer-Verlag, Budapest, Hungary, September 2004, pp. 430–435.

[HoeVi05] O. Hoenen and E. Violard (2005) : *Parallélisation d'un solveur adaptatif de l'équation de Vlasov*, in: 16èmes Rencontres Francophones du Parallélisme (RenPar 2005), Le Croisic (France), April 2005, pp. 249–258.

[HoeVi06] O. Hoenen and E. Violard (2006) : *An Efficient Data Structure for an Adaptive Vlasov Solver*, Research Report RR 06-02, ICPS - LSIIT laboratory (CNRS UMR-7005).

[HonSc04] D. Hong, L. L. Schumaker (2004) ; *Surface compression using a space of C^1 cubic splines with a hierarchical basis*, Geometric modelling Computing Vol.72, No.1-2, pp. 79–92.

[Meh-al06] . Mehrenberger, E. Violard, O. Hoenen, M. Campos Pinto and E. Sonnendrücker (2006) : *A Parallel Adaptive Vlasov Solver Based on Hierarchical Finite Element Interpolation,* Proceedings ICAP2004 St-Petersburg, Nuclear Inst. and Methods in Physics Research, A 558, pp. 188–191.

[Son-al04] E. Sonnendrücker, F. Filbet, A. Friedman, E. Oudet, J.L. Vay (2004) : *Vlasov simulation of beams on a moving phase-space grid*, Comput. Phys. Commun, 2004, Vol.164, pp. 390–395.

[Tor05] V. Torri (2005) : *Numerical Scheme for the One-Dimensional Vlasov-Poisson Equation Using Bi-Orthogonal Spline Wavelets,* Technical report, INRIA Lorraine, No RR-5757.

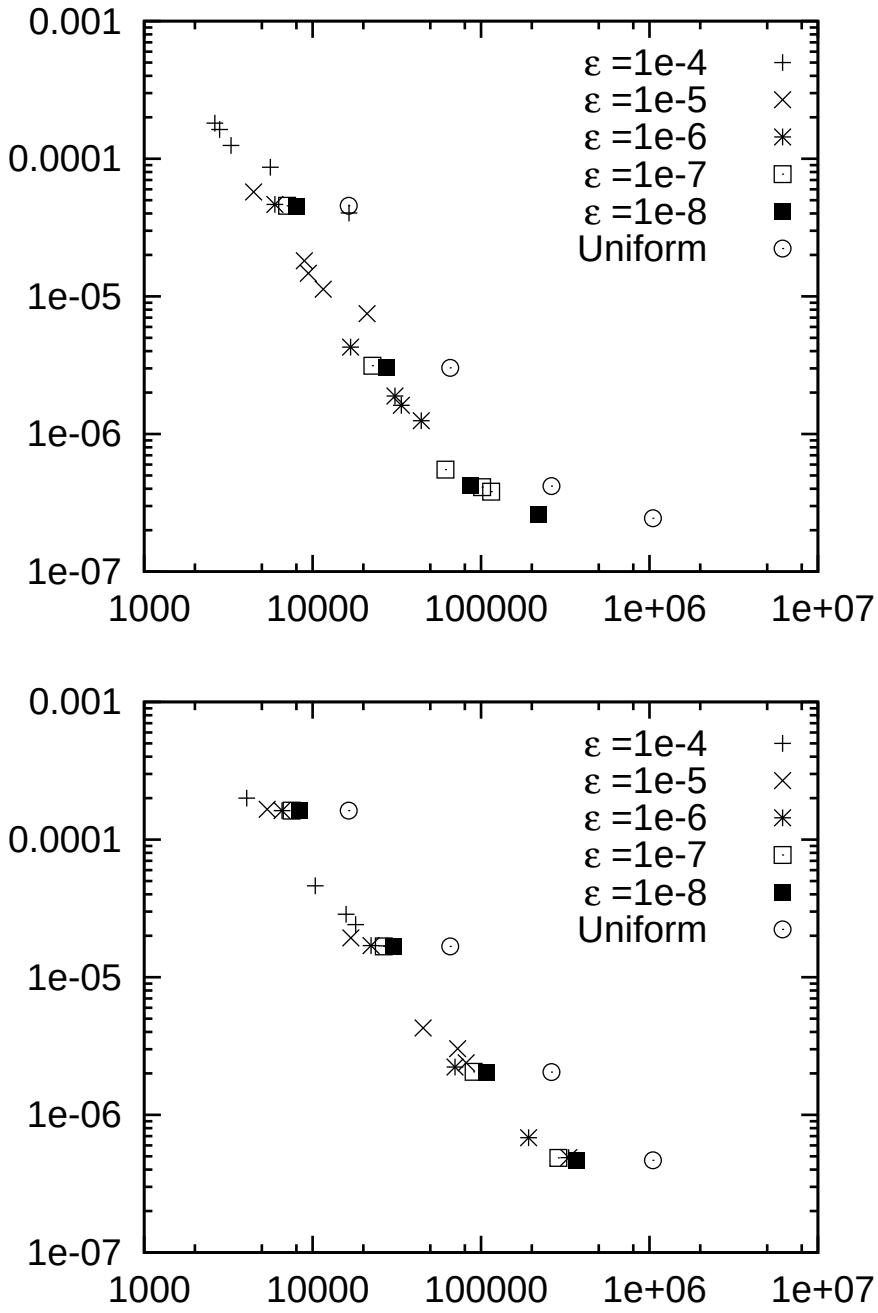

FIGURE 2. Error/number of cells for Hermite (on the top) and Lagrange (on the bottom). Each point on the figure corresponds to a value of J between 7 and 11.

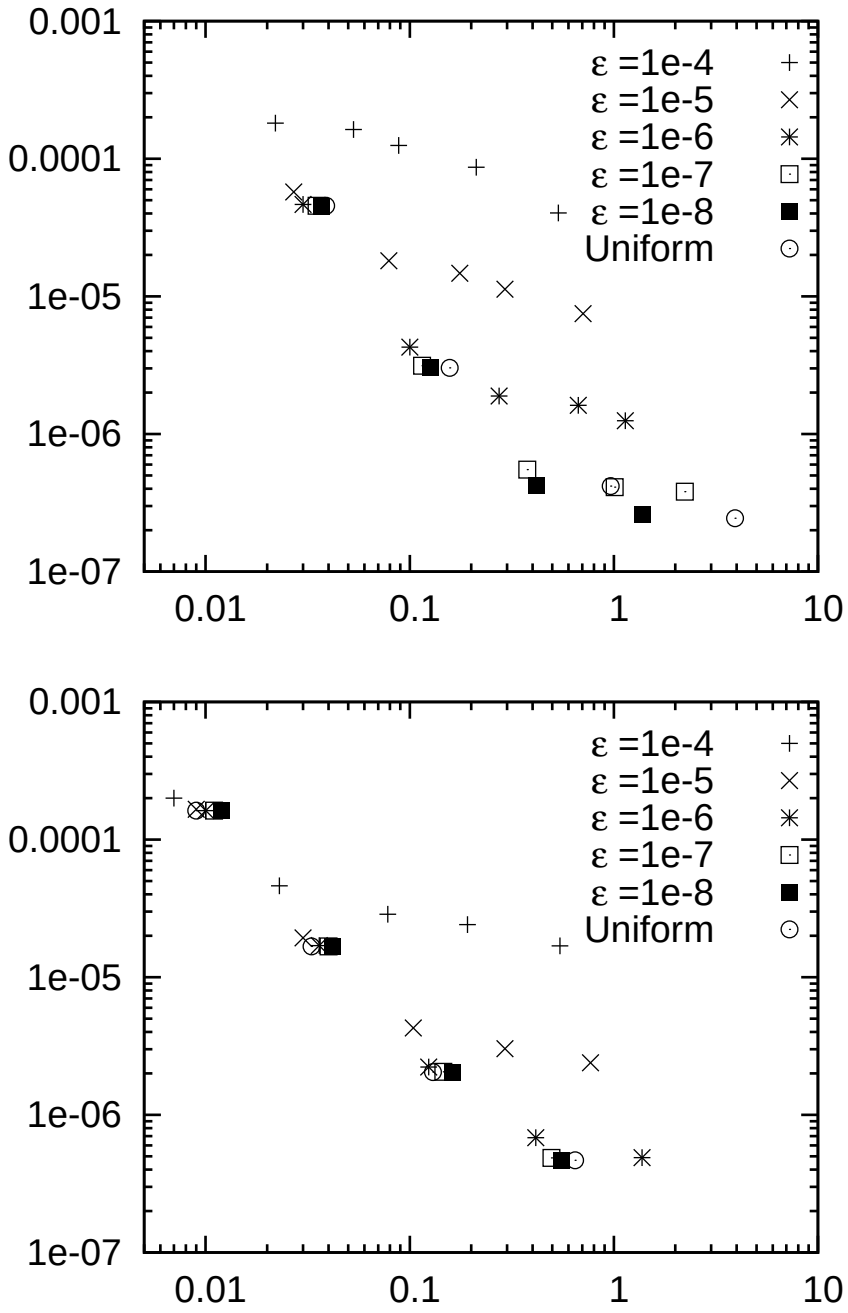

FIGURE 3. Error/average time to compute one iteration step (s) for Hermite (on the top) and Lagrange (on the bottom). Each point on the figure corresponds to a value of J between 7 and 11.

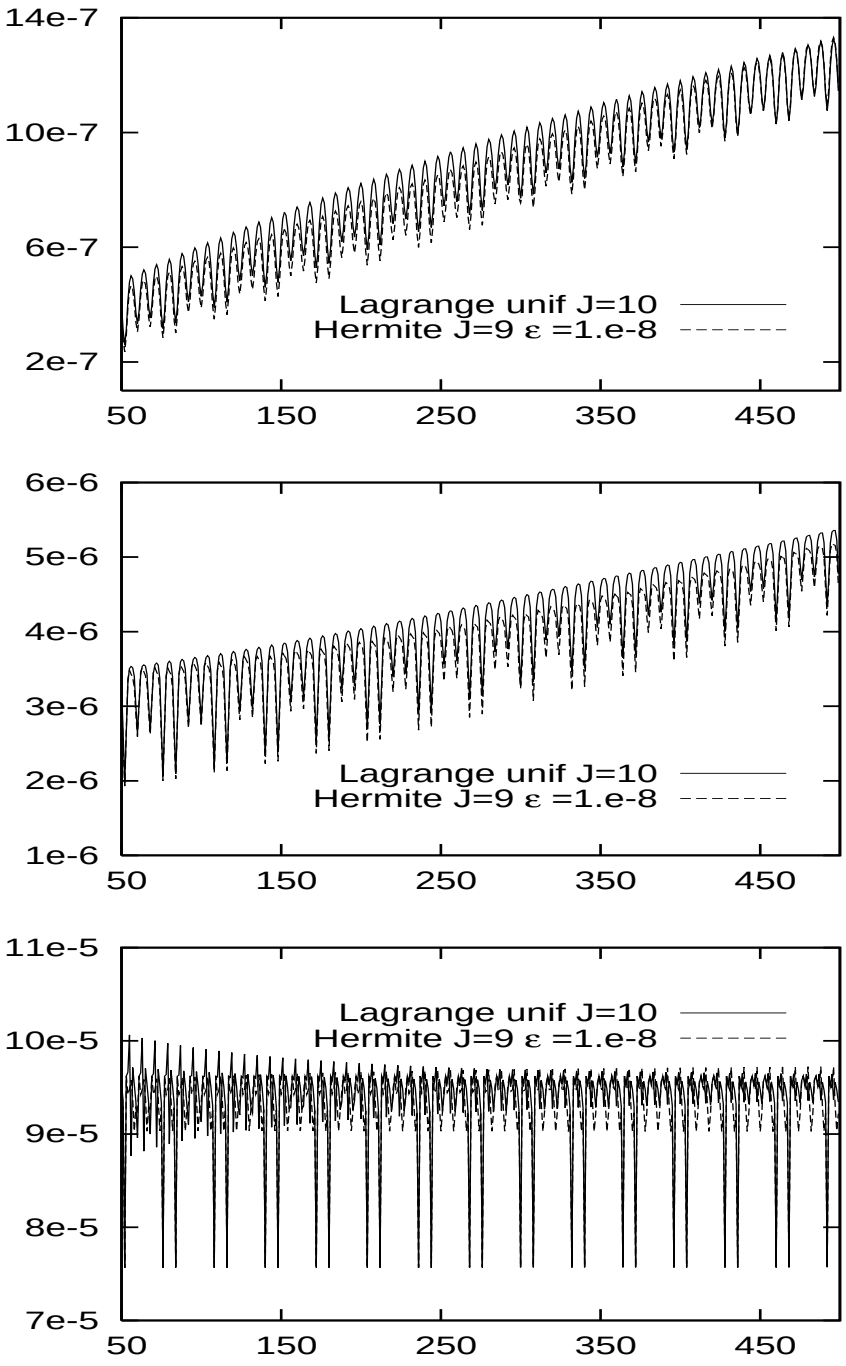

FIGURE 4. Error in norm L1 (on the top), L2 (on the middle), L∞ (on the bottom) for Hermite with J=9, Lagrange with J=10, $\epsilon = 10^{-8}$ and from 50 to 500 iteration steps.

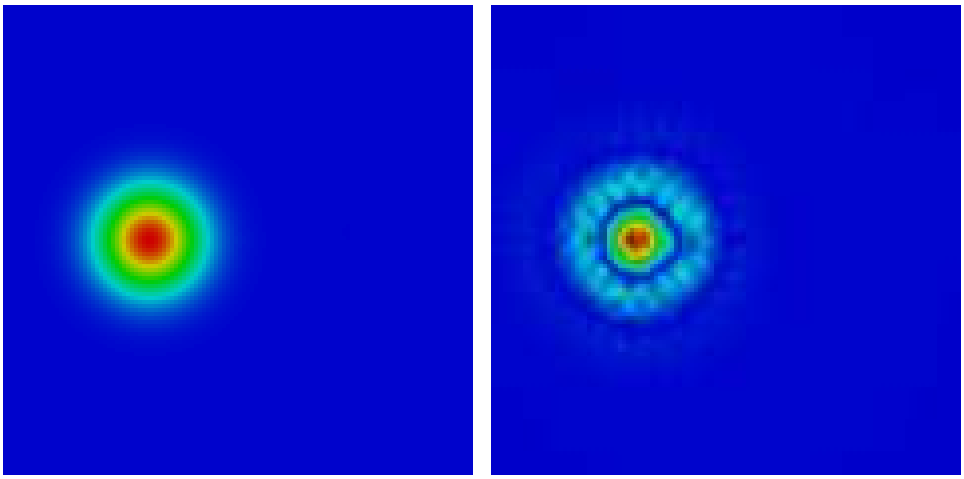

FIGURE 5. Solution and error for $\varepsilon = 10^{-6}$ and $J = 7$ after 900 steps in the Hermite case (the maximal error is 0.0223 in red).

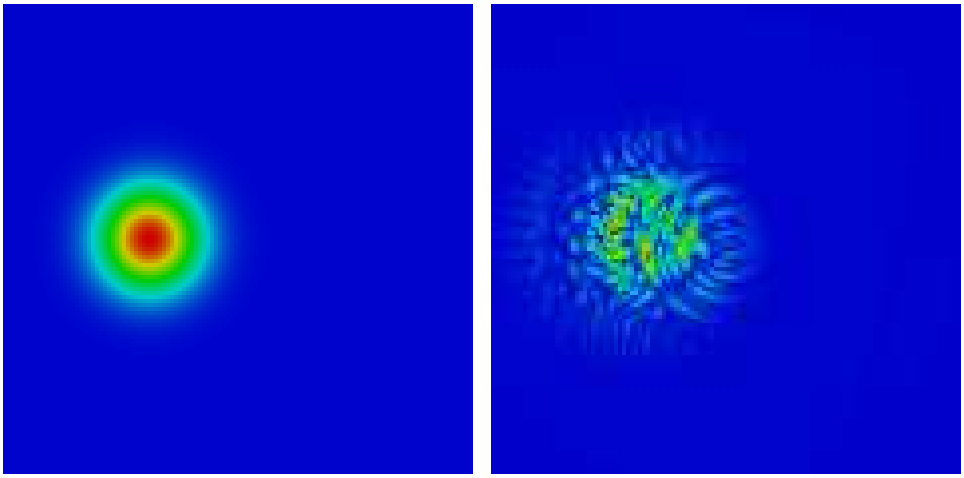

FIGURE 6. Solution and error for $\varepsilon = 10^{-6}$ and $J = 7$ after 900 steps in the Lagrange case (the maximal error is 0.0055 in red).