



# A Hermite type adaptive semi-Lagrangian scheme

Michel Mehrenberger, Eric Violard

## ► To cite this version:

Michel Mehrenberger, Eric Violard. A Hermite type adaptive semi-Lagrangian scheme. International Journal of Applied Mathematics and Computer Science, 2006, 10.2478/v10006-007-0027-y . inria-00110865

**HAL Id: inria-00110865**

**<https://inria.hal.science/inria-00110865>**

Submitted on 2 Nov 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A HERMITE TYPE ADAPTIVE SEMI-LAGRANGIAN SCHEME

MICHEL MEHRENBARGER\*, ERIC VIOLARD\*\*

\*IRMA (CNRS UMR-7005), Université Louis Pasteur, 7, rue René Descartes, 67084 Strasbourg, France  
e-mail: mehrenbe@math.u-strasbg.fr

\*\*Laboratoire ICPS - LSIIT (CNRS UMR-7005), Université Louis Pasteur, Pôle API, Boulevard Sébastien Brant, 67400 Illkirch, France  
e-mail: violard@icps.u-strasbg.fr

We study a new Hermite type interpolating operator arising in a semi-Lagrangian scheme for solving the Vlasov equation in the 2D phase space. Numerical results on uniform and adaptive grid are shown and compared with biquadratic Lagrange interpolation introduced in (Campos and Mehrenberger, 2004) in the case of a rotating Gaussian.

**Keywords:** Adaptive method, Vlasov equation, Numerical simulation, Hermite operator

## 1 Introduction

Adaptive semi-Lagrangian schemes for solving the Vlasov equation in the phase space have recently been developed: wavelet techniques (Gutnic et al., 2004; 2005), moving mesh method (Sonnendrücker et al., 2004), hierarchical finite element decomposition (Campos and Mehrenberger, 2004; 2005). 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 (see (Besse and Sonnendrücker, 2003) and also (Nakamura and Yabe, 1999)). We propose here to do the same in the adaptive context. Thanks to a well chosen Hermite interpolation recently found (Hong and Schumaker, 2004), 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 focuses 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 a criteria for compressing cells of the adaptive mesh. Last, section 6 shows our experimental results before concluding.

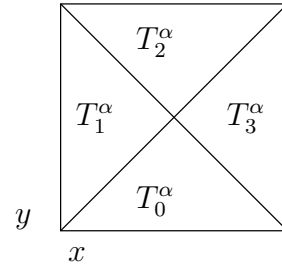


Figure 1. The 4 triangles of a cell  $\alpha$

## 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  $k, \ell$  and  $j$  are integers, and  $j$  denotes the level of the cell.

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

Given a cell  $\alpha$ , 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  $\alpha$  are  $\lambda, \mu \in [0, 1]^2$ . Let  $T_k^\alpha, k = 0, \dots, 3$  be the 4 triangles obtained by subdividing the cell  $\alpha$  with the diagonals (Fig. 1).

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

$$Q_d = \left\{ \sum a_{i,j} x^i y^j, \quad i, j \leq d \right\},$$

and

$$P_d = \left\{ \sum a_{i,j} x^i y^j, \quad i + j \leq d \right\}.$$

Now, let  $f$  be a function defined on  $\Omega$  prolonged on  $\mathbb{R}^2$

by zero (we can similarly prolongate it by periodicity), and  $(x, y) \in \Omega$ .

**Biquadratic Lagrange interpolation**  $P_L f(x, y)$  is defined as the unique element of  $Q_2$  on the cell  $\alpha = \alpha_{x,y}$  such that it coincides with  $f$  on the 9 degree of freedom at the 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  $\alpha$ .  $P_H f(x, y)$  is then the unique  $C^1$  spline on the cell  $\alpha = \alpha_{x,y}$ ,  $P_3$  on the triangles  $T_k^\alpha$ ,  $k = 0, \dots, 3$  such that there is coincidence on the 12 degrees of freedom at the corners:

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

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

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

### 3 A general iterative scheme

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

Let consider the Vlasov equation

$$\partial_t f + y \partial_x f + E(t, x) \partial_y f = 0. \quad (1)$$

The electric field  $E$  is generally computed via the Poisson equation. We will suppose here that it is a known function. We define classically the characteristic curves:

$$Z(t; s, x, y) = (X, Y) = (X(t; s, x, y), Y(t; s, x, y)),$$

such that

$$\frac{d}{dt} X = y, \quad \frac{d}{dt} Y = E$$

together with  $Z(s; s, x, y) = (x, y)$ . The function  $f = f(t, x, y)$  that we want to approximate, satisfying (1) is constant along these characteristics:

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

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

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:

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

Since  $f$  is constant along the characteristics, we have

$$f((n+1)\Delta t, x, y) = f(n\Delta t, \mathcal{T}^{n+1}(x, y)).$$

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  $\Omega$ . We fix a resolution level  $J \in \mathbb{N}^*$ .

**Uniform scheme** The semi-Lagrangian method, with propagation of the gradients has been studied on uniform grids in (Nakamura and Yabe, 1999) and (Besse and Sonendrücker, 2003). 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  $(x, y)$  corresponding to a degree of freedom, we compute the backward advected point  $\mathcal{T}^{n+1}(x, y)$ . The new value of the degree of freedom is thus  $f^n(\mathcal{T}^{n+1}(x, y))$  (or  $\frac{d}{dz} f^n(\mathcal{T}^{n+1}(x, y))$ , with  $z = x$  or  $y$ ). 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  $\tilde{\mathcal{M}}$ . Note that a cell  $\alpha_{k,\ell}^j$  is partitionned into 4 cells  $\alpha_{2k+k', 2\ell+\ell'}^{j+1}$ ,  $k', \ell' = 0, 1$ . We will say that the cell  $\alpha_{k,\ell}^j$  is the mother cell of the 4 daughter cells  $\alpha_{2k+k', 2\ell+\ell'}^{j+1}$ ,  $k', \ell' = 0, 1$ . The compression can then be done locally, by comparing the representation of the current function on the 4 daughters cells, with the interpolated function on the mother cell. If the two representations 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 the representation  $(f^0, \mathcal{M}^0) = \mathcal{C}(\mathcal{M}_J, f_u^0)$ .

- (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  $a \leq x < a + h$  and  $b \leq y < b + h$ .

**Biquadratic Lagrange interpolation** Given 9 numbers  $g_{i,j}$ ,  $i, j = 0, 1, 2$ , the function  $g(x, y)$  of degree  $\leq 2$  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  $t_0 = (x - a)N$ ,  $t_1 = 2t_0 - 1$ , 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  $t_0 = (v - b)N$ ,  $t_1 = 2t_0 - 1$ , 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 subtractions).

**Hermite interpolation** Given 16 numbers  $g_{i,j}$ ,  $g_{i,j}^x$ ,  $g_{i,j}^y$ ,  $i, j = 0, 1$ ,  $g_0^x$ ,  $g_1^x$ ,  $g_0^y$  and  $g_1^y$ , the  $C^1$  cubic spline,  $P_3$  on the 4 triangles  $T_0^\alpha$ ,  $T_1^\alpha$ ,  $T_2^\alpha$  and  $T_3^\alpha$ , satisfying

$$\begin{aligned} g(a + ih, b + jh) &= g_{i,j}, \\ \partial_x g(a + ih, b + jh) &= g_{i,j}^x, \\ \partial_y g(a + ih, b + jh) &= g_{i,j}^y, \quad \text{for } i, j = 0, 1 \quad \text{and} \\ \partial_x g(a + ih, b + h/2) &= g_i^x, \\ \partial_y g(a + h/2, b + ih) &= g_i^y, \quad \text{for } i = 0, 1, \end{aligned}$$

is uniquely determined.

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

$$\begin{aligned} g(x, y) &= g_{00} + (2u^3 - 3u^2)(g_{00} - g_{10}) \\ &+ (v^3 - 3v^2 + 3uv^2)(g_{00} - g_{01}) + (v^3 - 3uv^2)(g_{10} - g_{11}) \\ &+ h((2/3v^3 - 2u^2 - 3/2v^2 + u + 3/2uv^2 + u^3)g_{00}^u \\ &+ (-1/2v^2 + uv^2 + v - 3uv + 1/6v^3 + 2u^2v)g_{00}^v \\ &+ (2/3v^3 - 2v^2 + 4(-u^2 + u)v)g_0^v \\ &+ (-2/3v^3 - u^2 + 3/2uv^2 + u^3)g_{10}^u \\ &+ (1/2v^2 - uv^2 - uv + 1/6v^3 + 2u^2v)g_{10}^v \\ &+ (4/3v^3 - 2uv^2)g_1^u + (-2/3v^3 + 1/2uv^2)g_{11}^u \\ &+ (5/6v^3 - uv^2)g_{11}^v - 2/3v^3g_1^v \\ &+ (2/3v^3 + 1/2(-v^2 + uv^2))g_{01}^u \\ &+ (5/6v^3 - v^2 + uv^2)g_{01}^v \\ &+ (-4/3v^3 + 2(v^2 - uv^2))g_0^u \end{aligned}$$

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  $g(x, y)$  on triangle  $T_0^\alpha$  by the Hermite operator is 18 assignments, 49 multiplications and 53 additions. It is obtained by introducing some auxiliary variables as follows:

$$\begin{aligned} x_1 &= u^2, x_2 = uu_1, v_1 = v^2, v_2 = 3v_1, v_3 = uv_2, \\ v_4 &= vv_1, v_5 = 2/3v_4, v_6 = 4/3v_4, v_7 = 5/6v_4, v_8 = uv_1, \\ x_3 &= ux_2, x_4 = x_3 + 3/2v_8, v_9 = 1/6v_4 + 2x_1v, \end{aligned}$$

Moreover, for the Hermite operator, instead of computing  $g(x, y)$ ,  $\partial_x g(x, y)$  and  $\partial_y g(x, y)$  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  $\alpha$  of length  $h$  is:

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

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

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

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

$$\begin{aligned} \tilde{f}_{k1/2} = & ((f((k, 0)_\alpha) + f((k, 1)_\alpha))/2 \\ & + h/8(\partial_v f((k, 0)_\alpha) + \partial_v f((k, 1)_\alpha)), \end{aligned}$$

for  $k = 0, 1$ .  $\tilde{f}_{k1/2}$  and  $\tilde{f}_{1/2k}$  are the reconstructed values at the middle of the edges, with the 1D Hermite interpolation operator. Note that this compression test is not exactly the same than that introduced in the section 3. However it gives an easy and cheap local criterion of order of accuracy, because it only uses a 1D interpolation, and it has been successfully applied in our testcase. Other criteria may be used and compared.

**Biquadratic Lagrange compression** For a given mother cell  $\alpha$ , the compression test used is

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

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

## 6 Numerical results

We take for initial data

$$f_0(x, y) = e^{-0.07((40(x-0.5)+4.8)^2 + (40(y-0.5)+4.8)^2)},$$

in  $\Omega = [0, 1]^2$  and  $\Delta t = 0.19635$ . The transport operator is here given by a rotation of angle  $\Delta t$  around the center  $(0.5, 0.5)$ , so that the equation that is numerically resolved is (1) with  $E(t, x) = -(x - 0.5)$ .

We implemented our schemes in C++ and carried out our code on a Pentium4 processor cadenced at 3.06GHz 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  $256 \times 256$  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  $\epsilon$  and any interpolation operator, the number of cells is always lower with the adaptive scheme than with the uniform one. But that does not necessarily 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 iteration 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 execution 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 numerical 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 charge density. Since the computation cost of this density 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 (Hoenen et al., 2004). In particular, we designed a specific data structure (Hoenen and Violard, 2006) 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

- Besse N. and Sonnendrücker E. (2003): *Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space*, J. Comput. Phys., 191, (2003), pp. 341–376.
- Campos Pinto M. and Mehrenberger M. (2005): *Adaptive numerical resolution of the Vlasov equation*, Numerical Methods for Hyperbolic and Kinetic Problems, CEM-RACS 2003/ IRMA Lectures in Mathematics and Theoretical Physics 7, pp. 43–58.
- Campos Pinto M. and Mehrenberger M. (2005): *Convergence of an adaptive scheme for the one-dimensional Vlasov-Poisson system*, Technical report, INRIA Lorraine, No RR-5519.
- Gutnic M., Haeefe M., Paun I., Sonnendrücker E. (2004): *Vlasov simulations on an adaptive phase-space grid*, Comput. Phys Commun. Vol.164, pp. 214–219.
- Gutnic M., Haeefe M. and Latu G. (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.
- Hoenen O., Mehrenberger M. and Violard E. (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.
- Hoenen O. and Violard E. (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.
- Hoenen O. and Violard E. (2006): *An Efficient Data Structure for an Adaptive Vlasov Solver*, Research Report RR 06-02, ICPS - LSIIT laboratory (CNRS UMR-7005).
- Hong D., Schumaker L. L. (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.
- Mehrenberger M., Violard E., Hoenen O., Campos Pinto M. and Sonnendrücker E. (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.
- Nakamura T. and Yabe T.: *Cubic interpolated propagation scheme for solving the hyperdimensional Vlasov-Poisson equation in phase space*, Comput. Phys. Comm. 120 (1999), pp. 122–154.
- Sonnendrücker E., Filbet F., Friedman A., Oudet E., Vay J. L. (2004): *Vlasov simulation of beams on a moving phase-space grid*, Comput. Phys. Commun., 2004, Vol.164, pp. 390–395.
- Torri V. (2005): *Numerical Scheme for the One-Dimensional Vlasov-Poisson Equation Using Bi-Orthogonal Spline Wavelets*, Technical report, INRIA Lorraine, No RR-5757.

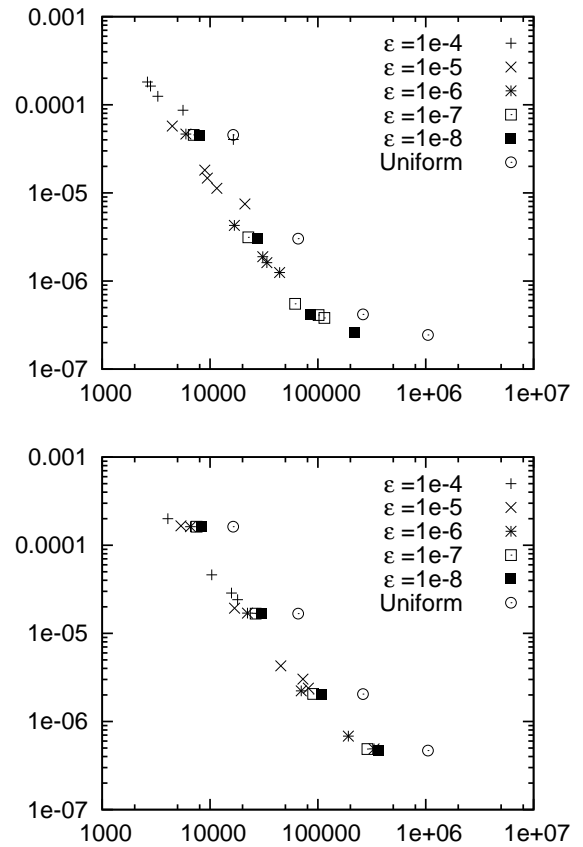


Figure 2. Error/number of cells for Hermite (on the top) and Lagrange (on the bottom). Each point on the figure corresponds to a value of  $J$  between 7 and 11. The three first symbols go from 7 to 11, and the last three symbols go from 7 to 10 (from the left to the right).

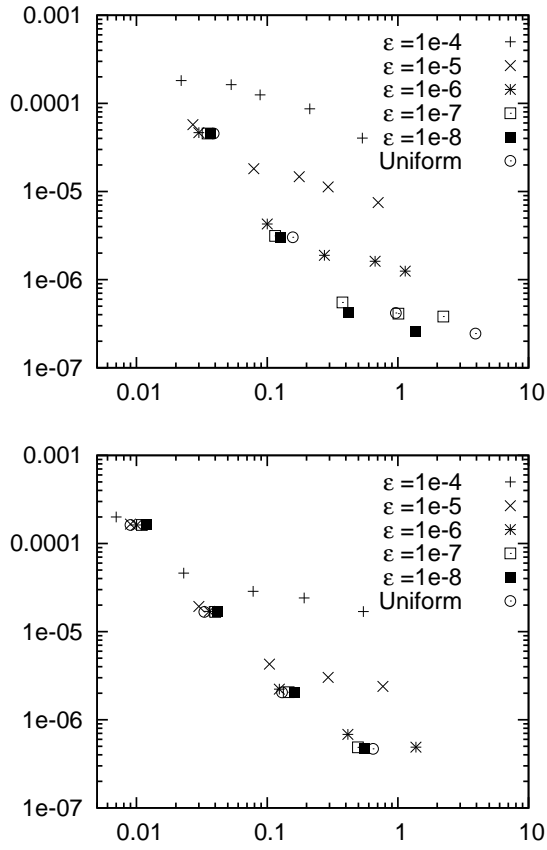


Figure 3. Error/average time to compute one iteration step (s) for Hermite (on the top) and Lagrange (on the bottom). Each point on the figure corresponds to a value of  $J$  between 7 and 11. The three first symbols go from 7 to 11, and the last three symbols go from 7 to 10 (from the left to the right).

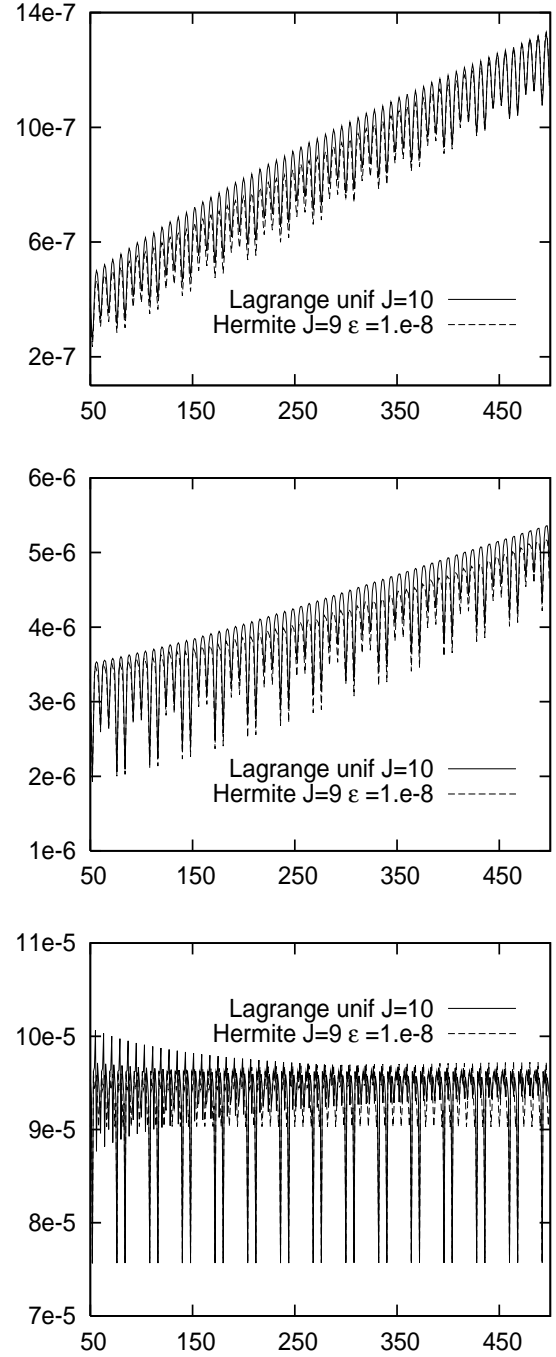


Figure 4. Error in norm L1 (on the top), L2 (on the middle),  $L_\infty$  (on the bottom) for adaptive Hermite scheme with  $J=9$  and  $\epsilon = 10^{-8}$ , uniform Lagrange scheme with  $J=10$  and from 50 to 500 iteration steps.

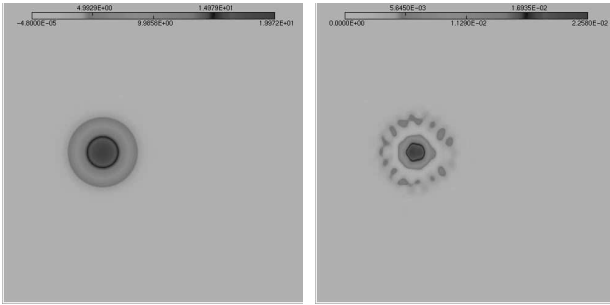


Figure 5. Solution  $f$  and error for  $\varepsilon = 10^{-6}$  and  $J = 7$  after 900 steps in the Hermite case.

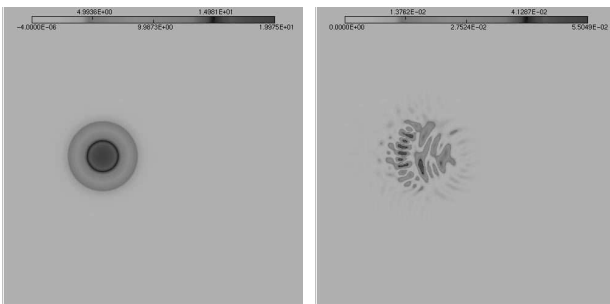


Figure 6. Solution  $f$  and error for  $\varepsilon = 10^{-6}$  and  $J = 7$  after 900 steps in the Lagrange case.