Noname manuscript No.
(will be inserted by the editor)

High-order Adaptive Mesh Refinement Multigrid Poisson Solver in any Dimension

Erwan Deriaz

Received: date / Accepted: date

Abstract A numerical method to solve the d-dimensional Poisson equation with $2p$th-order accuracy for arbitrary d and p integers is proposed in the Adaptive Mesh Refinement framework. This method relies on vertex-centered mesh refinement and interpolation, on compact finite difference schemes and on multigrid algorithm. It is compared to other existing methods. And, in extensive numerical experiments, a sixth-order version of it in dimensions two to six and a tenth-order version in dimension three are tested.

Keywords Poisson solver · Adaptive Mesh Refinement · Multigrid · Compact Finite Difference · High order scheme

Mathematics Subject Classification (2000) 65N50 · 65N55 · 65Y02

1 Introduction

Solving the Poisson Equation in dimension d:

$$\Delta u = v, \quad \text{with} \quad \Delta = \sum_{i=1}^{d} \partial_i^2 \tag{1.1}$$

plays a fundamental role in the Physics modelization and simulation.

Amongst many of its applications, this ubiquitous equation appears:

E. Deriaz
Institut Jean Lamour, Matériaux-Métallurgie-Nanosciences-Plasmas-Surfaces UMR 7198 - CNRS - Université de Lorraine, Campus Artem, 2 allée André Guinier - BP 50840, 54011 Nancy Cedex
Tel.: +33(0)372742557
E-mail: erwan.deriaz@univ-lorraine.fr

- in the computation of potential forces which derive from gradients of potentials of the kind $\mathbf{a} = \nabla\varphi$ with $\Delta\varphi = \pm\rho$ where ρ represents a density of mass or of charges and φ a gravitational field (plus case) or an electric field (minus case) for instances,
- to numerically solve the Incompressible Navier-Stokes Equation where the pressure is obtained through $\Delta P = -\rho \mathrm{div}\,((\mathbf{u}\cdot\nabla)\mathbf{u})$,
- to model dispersive effects as in the Heat Equation $\partial_t u = \Delta u$ which needs to be solved with implicit numerical methods of the kind $(Id - \nu\Delta)u_{n+1} = u_n$ with $\nu > 0$, related to Poisson Equation.

Some of the solutions to these equations develop singularities in space and/or in time. In this case the Adaptive Mesh Refinement (AMR) methods whose principle consists in automatically adapting the grid to the numerical solution, give access to all the scales, from the largest to the smallest with discrepancies of few orders of magnitudes as in the cosmological simulations [30]. Compared to the uniform grid methods, these methods cut down the memory needs of the simulations, and cut down also their computational times when their compression rates drop below 10% [8,11]. They allow to treat boundary layers [27], sudden disruptions [9] and to reduce the cost of large simulations in cosmology [30], in *ab initio* simulations [15] and in geological simulations [22,23].

Nevertheless these new possibilities offered by the AMRs come at the cost of more complexity in the implementation because of the versatility of the data structure, and more numerical difficulties because of the non-uniformity of the grid. In particular, as the discontinuities of the space step trigger discontinuities in the differentials of the error functions, composing discrete differential operators decreases the final order of accuracy –this effect does not appear in uniform grids where the final order of accuracy of the composed differential operator is usually equal to the minimum order of its differential components. Hence, in AMRs, the need for composed discrete operators [27] or high-order schemes [3,9]. The parallelization of AMRs, in particular for distributed memory, constitutes a real challenge overcome only by a few number of codes [30,5,23].

In uniform Cartesian grids with periodic boundary conditions the Fast Fourier Transform (FFT) with its complexity in $O(N_{pt}\log(N_{pt}))$ –N_{pt} denotes the number of points– and its spectral accuracy beats all concurrent numerical methods. In the 80's, the multigrid methods [18] with their complexity in $O(N_{pt}\log(N_{pt}))$ (the $\log(N_{pt})$ factor stands for the number of iterations necessary to reach the accuracy corresponding to the increase of the number of points N_{pt}) opened the door to efficient numerical methods suited to non periodic boundaries and immersed boundaries. Their principle (to separate scales to apply Gauss Seidel iterations) inspired the preconditioning of powerful Linear Solvers establishing the algebraic multigrid methods. These are blind to the underlying grid structure and can be used in the adaptive grids as do [22,14] for instance. In the 90's, the Fast Multipole Method [25,17,4] based on the integral solution of the Poisson Equation and on the properties of its Green kernel, appeared as a concurrent method efficiently handling the adaptive context and the presence of boundaries. This research area continues to develop actively nowadays [19,2].

In the following, we present and investigate an original geometric multigrid method in the AMR framework to solve the Poisson Equation in any dimension at a high order of accuracy. This multigrid algorithm is implemented inside a fully-threaded tree structure programmed in C, but it should also work in block-structured AMR implementations like [3,26]. Compared to other state-of-the-art Poisson solvers [16], geometric multigrid Poisson solvers yield very good results regarding the efficiency and the accuracy, similar to Fast Multipole Methods and better than algebraic multigrid methods. The method presented here relies on finite-difference compact schemes introduced by Collatz in [7] and more precisely on the original high-order schemes presented in [10]. It generalizes to any order and any dimension a fourth-order Poisson solver used to run a two-dimensional magnetic reconnection problem [9].

The course of the present paper goes as follows: Section 2 underpins the difference between vertex-based and cell-based AMRs, introduces the use of flags to attribute roles to the points, details the interpolation process and the refinement criterion; Section 3 recalls the original high-order compact finite-difference Poisson schemes from [10], unfolds the algorithm of the AMR Poisson solver and studies its stopping criterion, its cost and its convergence; Section 4 is concerned with extended numerical experiments, it tests the convergence of the AMR Poisson solver, its accuracy, its computational efficiency, its OpenMP parallelization, the gain obtained from initial guess and finally it statistically validates the theoretical refinement criterion.

2 Vertex-based AMR

Most of the AMR schemes rely on finite volume discretization. It is the most natural way to subdivide a computation cell [14]. It genuinely conserves the volume and allows to use all the powerful finite volume tools: fluxes, slope limiters, ENO/WENO adaptive stencils.

On the other side, finite difference based AMRs reach high order accuracies more easily [9]. It necessitates more complexity in the implementation but reward comes with its compactness and ability to reach high dimensions [11].

2.1 Differences between cell-based and vertex-based AMRs

In a d-dimensional space, a cell-based AMR refines a cell at level j by cutting it into 2^d finer cells at level $j+1$. Given a cell $C_{j,\mathbf{k}} \subset \mathbb{R}^d$:

$$C_{j,\mathbf{k}} = \prod_{i=1}^{d} \left[2^{-j}k_i, 2^{-j}(k_i+1)\right] \tag{2.1}$$

it splits into

$$\bigcup_{\varepsilon\in\{0,1\}^d} C_{j+1,2\mathbf{k}+\varepsilon} = \bigcup_{\varepsilon\in\{0,1\}^d} \prod_{i=1}^{d} \left[2^{-j-1}(2k_i+\varepsilon_i), 2^{-j-1}(2k_i+\varepsilon_i+1)\right]. \tag{2.2}$$

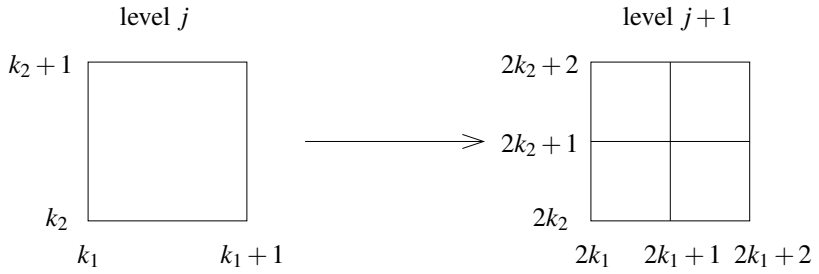


Fig. 2.1 Splitting of a two-dimensional cell into four cells. The numbers indicate their positions in their respective levels.

The two-dimensional case is illustrated by Fig. 2.1.

A vertex-based AMR refines a point at level j by introducing the 2^d-1 middle points located at its right (by convention) at level $j+1$:

$$\left\{2^{-j}\mathbf{k}\right\} = \left(2^{-j}k_i\right)_{1\leq i\leq d} \quad \text{spawns the points} \quad \bigcup_{\varepsilon\in\{0,1\}^d} \left\{2^{-j-1}\left(2\mathbf{k}+\varepsilon\right)\right\}. \tag{2.3}$$

One point at level j becomes 2^d points at level $j+1$ and one of these is exactly at the same place, when $\varepsilon = \mathbf{0}$. Hence this vertex-centered structure eases the interpolation process. Fig. 2.2 compares this spawning process with the cell-based splitting.

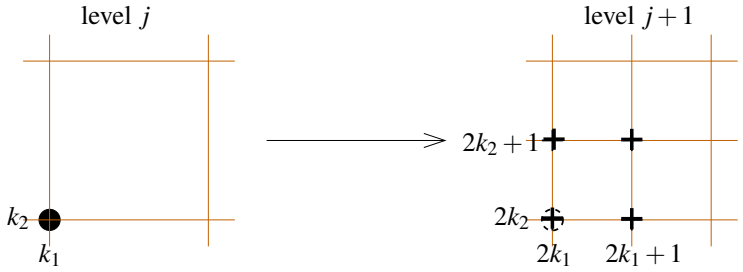
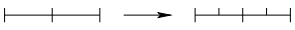
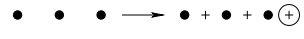


Fig. 2.2 Two-dimensional spawning of the new three vertices from the initial one. The numbers indicate their numbering in their respective levels.

2.2 From interpolability criterion to activation flag

A first difficulty is that the left/right symmetry is broken.

The cell-based splitting  naturally conserves the left/right symmetry, while the vertex-based spawning  breaks it.

A solution to this problem is to symmetrize the new mesh by activating only the middle new vertices which have their two neighboring older vertices. Hence it is necessary to put flags indicating whether a vertex is active or not.

This consideration generalizes to any d-dimensional refinement: at level j, given $\varepsilon \in \{0,1\}^d$ the vertex $2^{-j}(2\mathbf{k}+\varepsilon)$ is activated if and only if the vertices $2^{-j}(2\mathbf{k}+2\varepsilon')$ such that $\varepsilon' \in \{0,1\}^d$ with $\varepsilon'_i \leq \varepsilon_i \; \forall i \in [1,d]$, are activated.

In an approximation perspective, at level j, the older vertices upload their corresponding values at level $j-1$. Then the new vertices are interpolated from their direct neighbors with a 2nd-order accuracy.

Conversely this 2nd-order interpolation process can be seen as an activation criterion on the new points. A vertex is activated if and only if it can be interpolated through this process.

A similar criterion with $2p$th-order interpolability permits to generalize this 2nd-order interpolability criterion and to increase the order of accuracy of the interpolation in the AMR to any $2p$th-order with $p \geq 1$.

Remark 2.1 In the following numerical experiments, we will need as high as 12th-order interpolation to test the 10th-order accurate AMR three-dimensional Poisson solver.

2.3 Vertex typing

Some of the interpolable vertices are located outside any continuous active domain. For instance the fourth order two-dimensional case of Fig. 2.3 allows us to distinguish three kinds of vertices:
- the vertices inside the computational domain,
- the vertices remaining to the boundary,
- and the vertices which are only necessary for the interpolation.

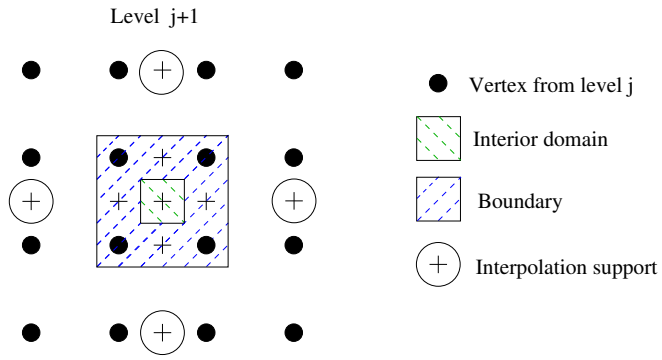


Fig. 2.3 Instance of fourth-order consistent distribution of vertices with their types. Only one among the horizontal couple of interpolatory points or the vertical one is necessary.

Typing these vertices gives us the opportunity to do the right operations at the right places. This typing relies on the fact that for all $\mathbf{k}$ node of the level $j+1$, the least interpolable vertex $\left(2^{-j-1}(2k_i+1)\right)_i$ is active if and only if all its neighbors are active. As a result if it is active (interpolable) then it is located inside the continuous domain of interpolable vertices.

Remark 2.2 The boundary can be made thicker with two points large boundaries, depending on the stencil used for the Poisson solver.

In the C program, all these flags are stored in `unsigned long long` variables (i.e. on $64 = 2^6$ bits) inside nodes, allowing the dimensionality to reach $d = 6$.

2.4 Interpolet

Wavelet theory provides the most comprehensive frame to translate the interpolation process into the basis element of a multiresolution analysis [13,21]. We look for an interpolatory compactly supported function $x \mapsto \varphi_p(x)$ verifying:

$$\varphi_p(0) = 1, \quad \varphi_p(k) = 0 \quad \forall k \in \mathbb{Z}^*$$

that induces a $2p$th-order interpolation of the points in $\frac{1}{2}\mathbb{Z}$ by the points in $\mathbb{Z}$ *i.e.*

$$I_{\frac{h}{2}}(f) = \sum_{k\in\mathbb{Z}} \varphi_p\left(-k+\frac{1}{2}\right) f(kh) = f\left(\frac{h}{2}\right) + O(h^{2p}) \quad \forall f \in C^{2p}.$$

Then the function φ_p satisfies the self-similarity relation:

$$\varphi_p(x) = \varphi_p(2x) + \sum_{\ell\in\mathbb{Z}} c^{(p)}_{2\ell+1}(\varphi_p(2x-(2\ell+1))$$

with $c^{(p)}_{2\ell+1}$ the coefficients associated to a $2p$-th order interpolation at the point 0 by the points $1+2\mathbb{Z}$. The coefficients associated to the shortest compact supports $[-2p+1, 2p-1]$ are displayed in table 2.1. And the corresponding basis functions $\varphi_p(x)$ are plotted in figure 2.4. These functions are called *Interpolets* [13].

According to [21,11] the coefficients $c^{(p)}_{2\ell+1}$ are symmetric $c^{(p)}_{-2\ell-1} = c^{(p)}_{2\ell+1}$, non zero only for $\ell < p$ and given by

$$c^{(p)}_{2\ell+1} = \frac{p}{2^{4p-3}} \binom{2p-1}{p-1} \frac{(-1)^\ell}{2\ell+1} \binom{2p-1}{p+\ell}. \tag{2.4}$$

For ℓ fixed, $c^{(p)}_{2\ell+1}$ is monotonous in p and tends to $\frac{2}{\pi}\frac{(-1)^\ell}{2\ell+1}$ which is the filter associated to the sinc function also called the Shannon wavelet [21,12], hence

$$\forall x \in \mathbb{R} \quad \lim_{p\to\infty} \varphi_p(x) = \mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}. \tag{2.5}$$

The sinc function is associated to the Fourier interpolation.

Table 2.1 Interpolation coefficients $c_{2\ell+1}^{(p)}$ for 2nd to 8th orders.

$2\ell+1$	-7	-5	-3	-1	1	3	5	7
$c_{2\ell+1}^{(1)}$				$\frac{1}{2}$	$\frac{1}{2}$			
$c_{2\ell+1}^{(2)}$			$-\frac{1}{16}$	$\frac{9}{16}$	$\frac{9}{16}$	$-\frac{1}{16}$		
$c_{2\ell+1}^{(3)}$		$\frac{3}{256}$	$-\frac{25}{256}$	$\frac{150}{256}$	$\frac{150}{256}$	$-\frac{25}{256}$	$\frac{3}{256}$	
$c_{2\ell+1}^{(4)}$	$-\frac{5}{2048}$	$\frac{49}{2048}$	$-\frac{245}{2048}$	$\frac{1225}{2048}$	$\frac{1225}{2048}$	$-\frac{245}{2048}$	$\frac{49}{2048}$	$-\frac{5}{2048}$

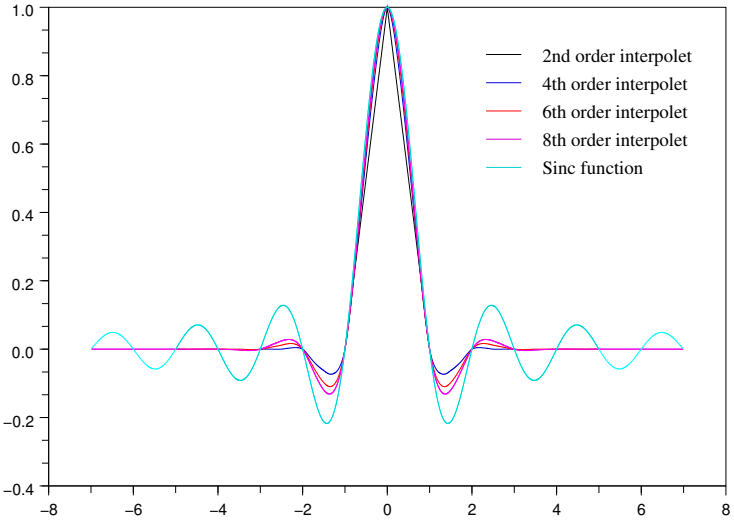


Fig. 2.4 Interpolets of 2nd, 4th, 6th and 8th orders tending to the sinc function.

In dimension d, the multidimensional *interpolet* is obtained by tensor product:

$$\Phi(\mathbf{x}) = \prod_{i=1}^{d} \varphi_p(x_i).$$

In the following, we denote by I_{2p} the $2p$th-order interpolation at mid-point using the coefficients $c_{2\ell+1}^{(p)}$ from Table 2.1.

2.5 Refinement criterion

In order to single out the areas of interest which need grid refinement to approximate a given function f, we expand f into a wavelet basis or a hierarchical basis $(\varphi_\lambda)_\lambda$ with $\lambda = (j, \mathbf{k})$:

$$f(\mathbf{x}) = \sum_{j\geq 0} \sum_{\mathbf{k}\in[0,2^j-1]^d} d_\lambda(f)\varphi(2^j\mathbf{x}-\mathbf{k}) = \sum_\lambda d_\lambda(f)\varphi_\lambda(\mathbf{x}). \tag{2.6}$$

If $|d_\lambda| > \varepsilon_j$ a threshold which depends on the level j, then the node $\mathcal{N}_\lambda$ containing φ_λ is tagged as area of interest. This area extends horizontally to the neighbors of the node $\mathcal{N}_\lambda$, vertically upward to all their ancestors and vertically downward to all its children.

A particular care should be given to the choice of the refinement criterion ε_j since it conditions the efficiency of the Adaptive Refinement. If the numerical scheme consistency *i.e.* the numerical accuracy behaves like $O\left(h^p f^{(p)}\right)$ then a best expansion basis (φ_λ) should satisfy $d_\lambda = O\left(h^p f^{(p)}\right)$ for any smooth f function. The norm equivalence [6]:

$$\|f\|_{B^s_q(L^q)} = O\left(\left\|\left(\|(2^{sj}2^{-\frac{dj}{q}}d_\lambda)_{\mathbf{k}\in[0,2^j-1]^d}\|_{\ell_q}\right)_{j\geq 0}\right\|_{\ell_q}\right) \tag{2.7}$$

points out the exact criterion minimizing the L^q-norm of the sth-derivative of f:

$$|d_\lambda| \geq 2^{\left(-s+\frac{d}{q}\right)j}\varepsilon_0 \tag{2.8}$$

with $\varepsilon_0 > 0$ a constant in agreement with the desired number of discretization points.

For instance, in our case we want to solve the Poisson equation $\Delta u = v$ and to minimize the error on the result u controlling the AMR thanks to the wavelet expansion of $v = \sum_\lambda d_\lambda \varphi_\lambda$. As we can see u as a $s = -2$nd derivative of v we should take $\varepsilon_j = 2^{\left(2+\frac{d}{q}\right)j}\varepsilon_0$ to minimize the L^q error on u.

If we just have a wavelet expansion $v = \sum_\lambda d_\lambda \varphi_\lambda$ such that $d_\lambda = O\left(h^{p'} f^{(p')}\right)$ with $p' \neq p$ we can consider in first approximation that $d_\lambda(f^{(p-p')}) = O\left(2^{(p-p')j} d_\lambda(f)\right)$. Hence as $d_\lambda(f^{(p-p')}) = O\left(h^p f^{(p)}\right)$, the criterion

$$2^{(p-p')j}|d_\lambda(f)| \geq 2^{\left(-s+\frac{d}{q}\right)j}\varepsilon_0 \quad i.e. \quad |d_\lambda| \geq 2^{\left(-s+p'-p+\frac{d}{q}\right)j}\varepsilon_0$$

yields a good result if the function f does not have too many disparities regarding its local scales, *i.e.* $f^{(k)}(x) \sim \xi^k(x) f(x)$ with the local frequency $\xi(x)$ bounded on the whole domain. Otherwise, the criterion should just stick to the solid estimate (2.7) and follow (2.8) or something in between.

The coarse-graining induced by a high-order large compact-support wavelet φ may constitute a serious flaw and make a lower-order smaller compact support wavelet expansion more optimal in practice.

2.6 Summary

To summarize this vertex-based AMR construction, it goes along the following the steps:

1. Spot the areas of interest thanks to a wavelet expansion.
2. Allocate nodes (with 2^d vertices each) in the tree structure so that a minimal active area encompasses the areas of interest.
3. Given these possibly active vertices and depending on the interpolations needed further, activate the vertices.
4. Attribute types to the vertices: inside the computational domain, on the boundary or just necessary for the interpolation process.

Remark 2.3 All the typed vertices are active. The others are deactivated and may be removed from the memory as it may represent a significant load of memory in the high-dimensional cases.

3 High order AMR Poisson solver

In the following, we present in detail a generalization to any dimension and any order of the two-dimensional fourth-order AMR Poisson solver introduced in [9] for the adaptive simulation of the magnetic reconnection. This algorithm necessitates three main ingredients:

- vertex-based AMR presented in the previous section,
- multigrid methods [18,28],
- high-order compact finite difference schemes for Poisson equations [7,29,24,10].

3.1 Compact finite difference schemes for Poisson equation

These elegant methods were introduced by [1,7]. The reference [10] describes a method to construct arbitrary order compact finite difference schemes for Poisson equation in arbitrary dimension. In particular it derives sixth-order any-dimensional schemes and a tenth-order three-dimensional scheme satisfying a *Discrete Maximum Principle* condition which implies the stability of these schemes and the convergence of the associated Gauss-Seidel iterations used in the multigrid methods.

We call *stencil* a discrete function $A : \mathbb{Z}^d \to \mathbb{R}^d, \mathbf{k} \mapsto a_{\mathbf{k}}$. From this function we form an approximation $\mathscr{A}$ of the Laplacian Δ. For any function u sufficiently regular at 0,

$$\lim_{h\to 0} \frac{1}{h^2} \sum_{\mathbf{k}\in\mathbb{Z}^d} a_{\mathbf{k}} u(\mathbf{k}h) = \Delta u(0) = \sum_{i=1}^{d} \partial_i^2 u(0).$$

By extension the A stencil may also denote the discrete operator $\mathscr{A}$

$$\mathscr{A} : \left(\mathbb{R}^d\right)^{\left(\mathbb{Z}^d\right)} \to \left(\mathbb{R}^d\right)^{\left(\mathbb{Z}^d\right)}, (u_{\mathbf{k}})_{\mathbf{k}\in\mathbb{Z}^d} \mapsto \left(\frac{1}{h^2} \sum_{\mathbf{k}'\in\mathbb{Z}^d} a_{\mathbf{k}'} u_{\mathbf{k}+\mathbf{k}'} \right)_{\mathbf{k}\in\mathbb{Z}^d}.$$

To define a compact finite difference scheme we need two stencils A and B with A an approximation of the Laplace operator and B an approximation of the Identity. The compact scheme is $2p$th-order if for all discretized function $u_h = (u(\mathbf{k}h))_{\mathbf{k}}$ with u C^{2p+2}-regular,

$$\frac{1}{h^2} A u_h = B(\Delta u)_h + O(h^{2p}). \tag{3.1}$$

If $a_{\mathbf{0}} < 0$ and $a_{\mathbf{k}} \geq 0$ for all $\mathbf{k} \neq \mathbf{0}$, then the compact scheme (A,B) verifies the *Discrete Maximum Principle*. This property helps constructing accurate and stable schemes. Relying on the one dimensional case, it is possible to show that for any dimension there exist such schemes at arbitrary orders. With the growing order of the compact scheme, the radii r_A and r_B of the stencils A and B, defined as $r_A = \max_{a_{\mathbf{k}} \neq 0} \|\mathbf{k}\|_\infty$ become larger. Nevertheless these radii remain much smaller than in the non compact finite difference case. For instance, the 6th-order compact scheme has radius $r_A = 1$ while we have $r_A = 3$ for the non compact scheme.

3.2 Multigrid Poisson algorithm

Let us consider embedded continuous subsets of $[0,1]^d$

$$\Omega_{J_{\max}} \subset \cdots \subset \Omega_{j+1} \subset \Omega_j \subset \cdots \subset \Omega_0 = [0,1]^d$$

with periodic boundary condition. The set Ω_j defines a discrete set of grid points

$$\omega_j = \Omega_j \cap 2^{-j}\mathbb{Z}^d.$$

We denote ω_j^{c} the complementary of ω_j inside $[0,1]^d \cap 2^{-j}\mathbb{Z}^d$. Then, inside ω_j we distinguish the interior $\mathring{\omega}_j$ of ω_j and its border $\partial\omega_j$ such that

$$\mathrm{dist}(\mathring{\omega}_j, \omega_j^{\mathrm{c}}) = \max(r_A, r_B) \quad \text{and} \quad \partial\omega_j = \omega_j \setminus \mathring{\omega}_j.$$

Doing so, we obtain boundaries $\partial\omega_j$ thick enough to provide for the boundary conditions of the compact scheme (3.1) on each level j. We remind that at level j we have the space step $h = 2^{-j}$. We also stress that in the algorithm, from the computer memory point of view, $x_\lambda \neq x_{\lambda'}$ if $\lambda \neq \lambda'$ even if $2^{-j}\mathbf{k} = 2^{-j'}\mathbf{k}'$.

With all these elements established, the AMR multigrid algorithm at arbitrary order and dimension presented and tested in this manuscript to solve $\Delta u = v$ works as follows:

1. The initial **arguments** given to the routine are:
   (a) a typed adaptive grid $(\omega_j)_{j\in[0,J_{\max}]}$,
   (b) the right hand side discrete function v,
   (c) an initial guess u_0, taken equal to zero if there is no prior estimate of the solution u
   (d) and a relative error tolerance $tol > 0$.

2. On the **initialization** step we make:
   (a) $u = u_0$,
   (b) $v = v - \langle v \rangle$, so that $\int v = 0$,
   (c) interpolation of the boundaries:
   for $j = 1$ to $J_{\max}$, for $x_\lambda \in \partial\omega_j$ do

$$v(x_\lambda) = I_{2p}(\omega_{j-1})v$$

$$u(x_\lambda) = I_{2p+2}(\omega_{j-1})u.$$

   (d) We form the right hand member Bv:
   for $j = 1$ to $J_{\max}$, for $x_\lambda \in \mathring{\omega}_j$ do

$$f(x_\lambda) = B\,v(x_\lambda) = \sum_{\mathbf{k}'} b_{\mathbf{k}'} v(x_{j,\mathbf{k}+\mathbf{k}'}).$$

   (e) We start with the residue $res = tol \times \|v\|_\infty$.

3. Beginning of the **loop**[1]
   while $\|res\|_\infty \geq tol \times \|v\|_\infty$ do
   (a) for $j = J_{\max}$ to 1 do
      i. for $x_\lambda \in \mathring{\omega}_j$ do

$$res(x_\lambda) = f(x_\lambda) - \frac{1}{h^2} A\,u(x_\lambda)$$

$$tmp(x_\lambda) = \frac{h^2}{a_{\mathbf{0}}} res(x_\lambda).$$

      ii. Execute twice the *Gauss-Seidel iteration*:
      for $x_\lambda \in \mathring{\omega}_j$ do

$$tmp(x_\lambda) = tmp(x_\lambda) + \frac{h^2}{a_{\mathbf{0}}} \left(res(x_\lambda) - \frac{1}{h^2} A\ tmp(x_\lambda) \right).$$

      iii. Update *res* and u at the level j
      for $x_\lambda \in \mathring{\omega}_j$ do

$$res(x_\lambda) = res(x_\lambda) - \frac{1}{h^2} A\ tmp(x_\lambda)$$

$$u(x_\lambda) = u(x_\lambda) + tmp(x_\lambda).$$

[1] The algorithm uses two intermediate variables: *res* which stores the residual and *tmp* which stores the increments of u provided by the Gauss-Seidel algorithm.

iv. Pass the residual and u to the level $j-1$ with the tensorial stencil $\left[\frac{1}{4}\ \frac{1}{2}\ \frac{1}{4}\right]^d$ for $x_{j-1,\mathbf{k}} \in \mathring{\omega}_{j-1}$ such that $x_{j,2\mathbf{k}} \in \mathring{\omega}_j$ do

$$res(x_{j-1,\mathbf{k}}) = \begin{bmatrix}\frac{1}{4} & \frac{1}{2} & \frac{1}{4}\end{bmatrix}^d res(x_{j,2\mathbf{k}})$$

$$u(x_{j-1,\mathbf{k}}) = \begin{bmatrix}\frac{1}{4} & \frac{1}{2} & \frac{1}{4}\end{bmatrix}^d u(x_{j,2\mathbf{k}}).$$

(b) $S_0 = res(x_{0,\mathbf{0}})$
for $j = 0$ to $J_{\max}$, for $x_\lambda \in \omega_j$ do

$$f(x_\lambda) = f(x_\lambda) - S_0.$$

(c) Prediction-correction step
for $j = 1$ to $J_{\max}$, for $x_\lambda \in \mathring{\omega}_j$ do

$$u(x_\lambda) = u(x_\lambda) + I_{2p+2}(\omega_{j-1})tmp(x_\lambda)$$

$$tmp(x_\lambda) = tmp(x_\lambda) + I_{2p+2}(\omega_{j-1})tmp(x_\lambda).$$

4. At the **end** of the iterative process, the result is available in u.

3.3 Computational cost

As one can see, this special case of geometric multigrid method naturally extends the original multigrid algorithm [18] to a vertex-based AMR. In particular considering points instead of volume cells as most AMR papers do [14,27,30] makes it easier to stay close to the original finite-difference multigrid design. The computational costs are similar to the uniform grid case although it includes the overhead induced by the more complex AMR data structure, *cf* Fig. 4.6.

The observed minimum tolerance after which the residual stops decreasing lies in between ε and $10^4\varepsilon$ where $\varepsilon = 10^{-16}$ stands for the computer error in double precision, *cf* Fig. 4.5, depending on the accuracy of the approximation (*i.e.* $\sim h_{\min}$). The greater is the accuracy, the larger will be the minimum bound since a good approximation of it is given by

$$2d\frac{\|u\|_\infty}{\|v\|_\infty}h_{\min}^{-2}\varepsilon \quad \text{or more precisely} \quad \sup_{\mathbf{x}\in[0,1]^d}\left(2d\frac{|u(\mathbf{x})|}{\|v\|_\infty}h(\mathbf{x})^{-2}\varepsilon\right).$$

Depending on the number of dimensions and on the compact scheme used, this tolerance is reached in about fifteen iterations. The most costly operations are the application of A four times for each iteration, once for computing the residual (step 3(a)i.), twice in the Gauss-Seidel method (step 3(a)ii.) and once when updating the residual (step 3(a)iii.), and the computation of the right hand member f *i.e.* applying B once

(step 2(d)). The cost of applying a stencil depends on the number of its non-zero elements. Hence the total cost C of the Poisson solver depends on the cardinality of the stencil A mainly and of the stencil B in a lesser extent and can be written:

$$C = (\#B + N_{it}(4\#A + 2\times 3 + 2(2p+2)))N_{pt} \tag{3.2}$$

where $\#B$ and $\#A$ stand for the numbers of non zero elements of the stencils B and A, N_{it} the number of iterations (of the loop 3.), N_{pt} the number of points and $2p$ the order of the method. We neglected the small additional costs dependent on the size of the boundaries including them in the fuzzy definition of N_{pt}. The cardinalities $\#A$ and $\#B$ directly depend on the order p and on the dimensionality d. They should roughly evolve as $O(p^{d-1})$ for $\#A$ and $O(p^d)$ for $\#B$ although in the non compact case they have the values $\#A = 2dp+1$ and $\#B = 1$.

For instance, the 3-dimensional 6th-order HOC stencil [29] verifies $\#A = 27$ (or 16 if we take advantage of the tensorial structure of the stencil), $\#B = 25$ and only needs 12 iterations to converge to the computer rounding error so its cost is given by $C = 1585N_{pt}$. Of course it is possible to decrease this cost by fixing a larger error tolerance and taking fewer iterations. For instance converging to 10^{-6} only takes 5 iterations then the cost is given by $C = 675N_{pt}$.

This compares advantageously to the costs given for the pth-order Fast Multipole Method [4] whose optimal implementation in 3 dimensions yields

$$C = 200N_{pt}p + 3.5N_{pt}p^2.$$

Which means $C = 1326N_{pt}$ for the 6th-order $p = 6$ case. This small computation confirms the findings from [16] which states that the two methods have equivalent costs.

3.4 Convergence of the multigrid iterations

As this AMR multigrid algorithm works very similarly to the original one [18], it should be possible to prove its convergence similarly. If we adopted the technique presented in [30] which solves the Poisson equation recursively on the ω_j domains for j decreasing then the proof of convergence is exactly the same as in the uniform case. Here there is an additional correction from the finer levels to the coarser ones because the residual is locally computed on the finest level. This influences the rate of convergence.

For instance if $r_A = 2$ and the first boundary goes through the pivotal points $(2k_i)_i$ which straightly get their values from level j, and not through the points in the middle of the cells $(2k_i+1)_i$ which are the most dependent on interpolation (see Fig. 3.1 to visualize the situation) then the convergence rate shifts from $1/2$ to $1/8$ which represents a non-negligible gain of efficiency.

In order to reduce the number of iterations N_{it} in (3.2), the initial guess u_0 chosen as close as possible to u in the H^2-norm allows a dramatic cut of the computational costs. In the simulations with time evolution it is possible to first extrapolate the solution u from the previous time steps in order to provide an accurate initial guess.

The H^2-norm criterion comes from the computation of the residual as $res = v - \Delta u_0$ which is the pivotal operation of the Gauss Seidel iterative algorithm. Hence a noisy initial guess, even close to the solution in the L^∞-norm, has little interest.

This specificity of the geometric multigrid methods is a strong incitement to use them in simulations. Let us notice that the algebraic multigrid methods can benefit from similar initial guess when they rely on iterative solvers [22].

4 Numerical experiments

4.1 High order compact schemes (A,B) tested

The finite difference compact schemes used to test the present high order AMR Poisson solver were obtained in [10]. All of them satisfy a condition implying the *Discrete Maximum Principle* and favorable to the fast convergence of the Gauss-Seidel iterations. This condition states: $a_{\mathbf{0}} < 0$ as small as possible and all the other $a_{\mathbf{k}}$ coefficients positive if $\mathbf{k} \neq \mathbf{0}$.

4.1.1 Tenth-order three-dimensional scheme

This scheme was designed to be tenth-order and to have few points in the stencil A although it does not realizes the minimum.

The A stencil is composed of 59 points: the central point $\mathbf{0}$, six symmetric points of type $(1,0,0)$, twelve of type $(1,1,0)$, eight of type $(1,1,1)$, 24 of type $(2,1,1)$ and eight of type $(2,2,2)$, so it has a radius $r_A = 2$. The B stencil has 113 points and a radius $r_B = 4$. All their localizations and their values are indicated in [10].

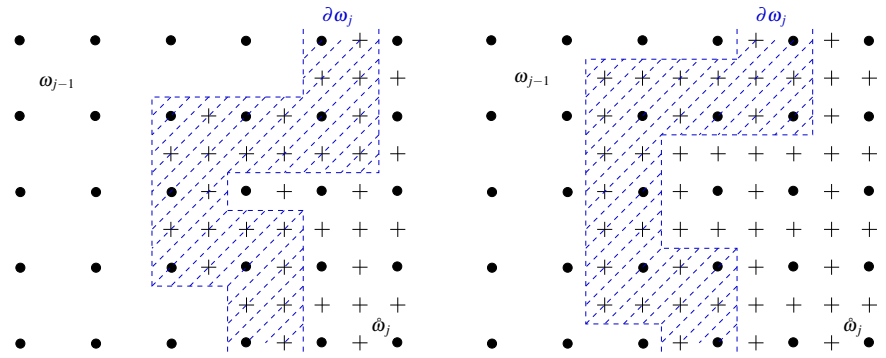


Fig. 3.1 Two ways of fixing the boundary with respect to the interior domain $\mathring{\omega}_j$: setting the limit just after the middle points (on the left) or just after the pivotal points (on the right). The first case is more efficient with respect to memory management when r_A is even, the second case converges much faster.

4.1.2 Sixth-order any-dimensional schemes

This group of schemes generalizes the *HOC* (High Order Compact) stencil [1,29] to any dimensions. Their stencils have tensorial structures so their complexities only increase linearly with the number of dimensions d.

For α, p, β, γ real numbers, the A stencil is of the form:

$$A = \alpha[p \;\; 1 \;\; p]^d + \begin{array}{c} \gamma \\ | \\ \gamma - \beta - \gamma \\ | \\ \gamma \end{array} \tag{4.1}$$

namely

$$a_{\mathbf{0}} = \alpha + \beta, \quad a_{(1,0,\ldots,0)} = \alpha\, p + \gamma, \quad \text{and} \quad a_{(\underbrace{1,\ldots,1}_{k \text{ times}},0,\ldots,0)} = \alpha\, p^k \quad \text{for} \quad k \geq 2.$$

To reach the sixth-order accuracy, computations presented in [10] lead to:

$$p = \frac{1}{3}, \quad \alpha = \frac{3}{2}\left(\frac{3}{5}\right)^{d-2}, \quad \beta = -\frac{25+2d}{6} \quad \text{and} \quad \gamma = \frac{1}{6}.$$

The B stencil is built similarly:

$$B = \omega[q \;\; 1 \;\; q]^d + \begin{array}{c} b_{01} \\ | \\ 0 \\ | \\ b_{01} - 0 - \lambda - 0 - b_{01} \\ | \\ 0 \\ | \\ b_{01} \end{array}$$

namely

$$b_{\mathbf{0}} = \omega + \lambda, \quad b_{(\underbrace{1,\ldots,1}_{k \text{ times}},0,\ldots,0)} = \omega\, q^k \quad \text{for} \quad k \geq 1, \quad \text{and} \quad b_{(2,0,\ldots,0)} = b_{01},$$

with

$$q = \frac{1}{7}, \quad \omega = \frac{9}{10}\left(\frac{7}{9}\right)^d, \quad \lambda = \frac{1}{10} + \frac{d}{120} \quad \text{and} \quad b_{01} = -\frac{1}{240}.$$

The radii of A and B are $r_A = 1$ and $r_B = 2$. Applying the stencil A has a $1+5d$ complexity instead of the usual $O(d^3)$ minimum complexity of compact none-tensorial type stencils, and of the $1+6d$ complexity of the non-compact stencil.

4.2 Gaussian test case

In the periodic domains $[-0.5,0.5]^d$ we consider the Gaussian function:

$$u(\mathbf{x}) = F(r) = \exp\left(-\frac{r^2}{\sigma^2}\right) + b \tag{4.2}$$

with $r = |\mathbf{x}|$ and the parameter σ small enough so that $\exp(-0.5^2/\sigma^2) \ll \varepsilon$ the computer rounding error. For lower dimensions we take $\sigma = 0.005$ as in [14] so $\exp(-0.5^2/\sigma^2) = exp(-100^2) \approx 10^{-4343}$. The number b is chosen such that

$$\int_{[-0.5,0.5]^d} u(\mathbf{x})\, d\mathbf{x} = 0.$$

With periodic boundary conditions, the solution u of the equation $\Delta u = v$ is determined modulo an additive constant. The non uniformity of the grid makes the condition $\int u(\mathbf{x})\, d\mathbf{x} = 0$ difficult to reach in practice. Hence we choose a b minimizing the L^∞ error from the discrete solution to the exact one.

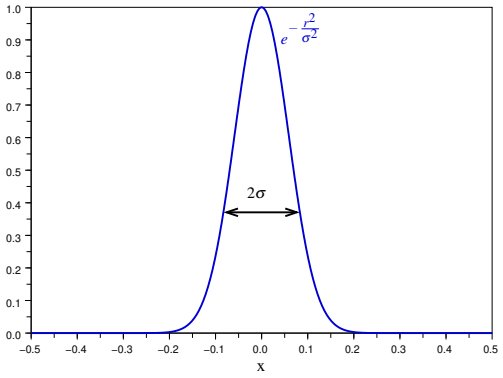


Fig. 4.1 Gaussian with $\sigma = \frac{0.5}{6}$ much larger than $\sigma = 0.005$.

The argument function v, right hand member of the equation $\Delta u = v$, is given by:

$$v(\mathbf{x}) = \left(\frac{4|\mathbf{x}|^2}{\sigma^4} - \frac{2d}{\sigma^2}\right) \exp\left(-\frac{|\mathbf{x}|^2}{\sigma^2}\right) + c$$

with c a free real constant such that $\int v(\mathbf{x})\, d\mathbf{x} = 0$ in the discrete form.

4.2.1 Verification of the order of convergence and of the accuracy

To assert the accuracies of the schemes, we begin with a test where starting from a coarse non uniform grid we half its mesh length everywhere once, then twice, and so on, computing the resulting L^∞, L^2 and L^1 errors. That way the orders of convergence appear clearly in two to six dimensions.

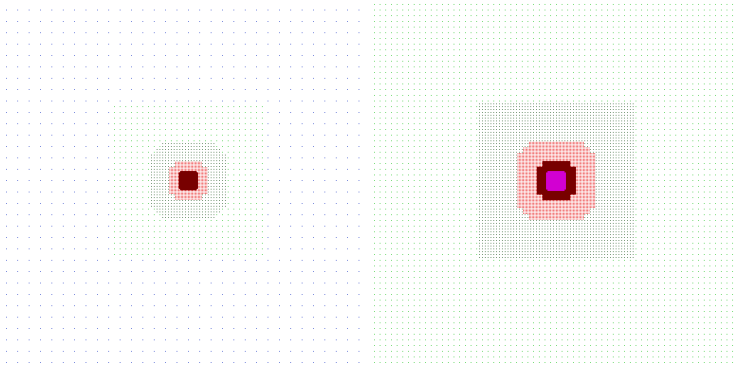


Fig. 4.2 First two instances of the 3-dimensional grid with maximum refinement $J = 9$ corresponding to a 512^3 uniform grid (left) and $J = 10$, a 1024^3 grid (right) used to test the accuracy of the 6th-order scheme. Different colors correspond to different point concentrations.

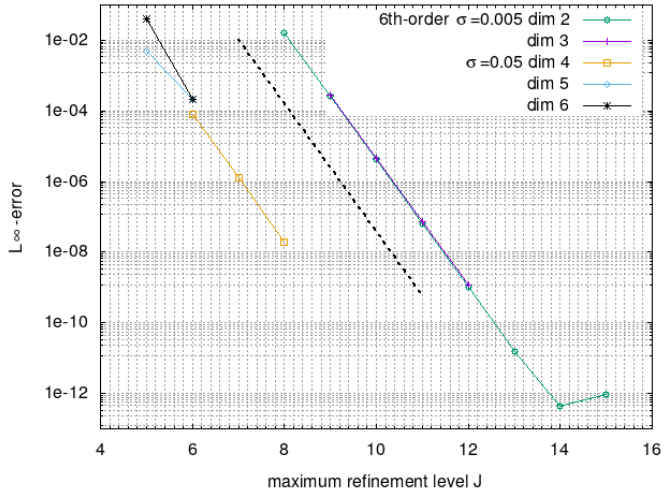


Fig. 4.3 Order of convergence for the 6th-order scheme in various dimensions. The dashed line represents the 6th-order slope.

All the following numerical experiments were run on a 16 cores, 32 threads, 32GB RAM Dell Precision T5610 computer.

Figure 4.2 shows two successive instances of grid. The grid points are represented by dots whose colors change when the mesh length halves. Figures 4.3 and 4.4 show the convergence tests of the 6th-order and 10th-order schemes respectively. These can be compared to the same experiment in [14] where a 2nd-order algebraic multigrid Poisson solver induces a 10^{-6} L^2-error in the finest 3D case. Given the fact that the L^2-norm of the exact solution is approximately 5×10^{-4}, one can measure the improvement of accuracy that the present geometric multigrid method provides.

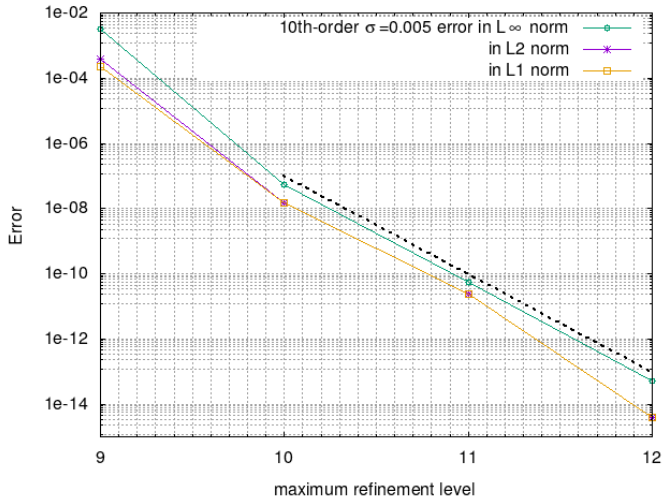


Fig. 4.4 Convergence for the 3-dimensional 10th-order scheme in various norms. The dashed line represents the 10th-order slope.

4.2.2 Computational efficiency

The cost estimate Eq. (3.2) is verified quite accurately knowing that each operation (rawly speaking accessing an element of the grid) costs 10^{-8}s. The OpenMP parallelization of the C code proves useful up to eight threads.

In Fig. 4.5 we visualize the convergence of the iterative process and see the number of necessary loop iterations in the Multigrid Poisson algorithm, paragraph 3.2. For instance, no more than 10 iterations are necessary to reach a residual equal $10^{-11}\|\Delta u\|_\infty$ criterion ending the iterative process, with the 6th-order 2D scheme.

As expected theoretically due to the tensorial structure of the stencil, and as shown in Fig. 4.6, the cost depends bilinearly on the number of points and on the number of dimensions. Above four dimensions, the shape of the adaptive grid plays a prominent role as illustrated by the two six-dimensional cases: in the most costly one, the interior points only represent 25% of the total number of points, while in the other one they represent 95% of it. Regarding the 10th-order 3-dimensional scheme, its computer times amount only to one and a half those of the 6th-order scheme.

For this experiment Fig. 4.6, the minimum number of interior points was taken equal to 15e+3 in 2D, 3e+5 in 3D, 1.36e+6 in 4D, 2.78e+6 in 5D and 17e+6 in 6D. These figures are multiplied by a number approximately equal to 2^d to plot the second curve, except for the 5D and 6D cases where the factors were taken equal to 10 and 3.3 respectively.

From now on we skip the 6-dimensional experiments because of their enormous memory needs.

Based on three runs of the Poisson algorithm with the residual inferior to $10^{-10}\|\Delta u\|_\infty$ ending criterion, Fig. 4.7 represents the average time of execution per interior point when applying an OpenMP task parallelization. Increasing the number of threads un-

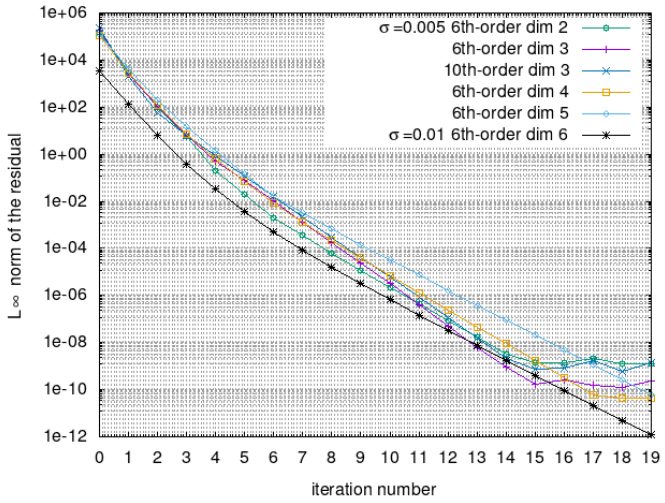


Fig. 4.5 Convergence process of the algorithm 3.2. The first iterations achieve the largest reductions of the residual.

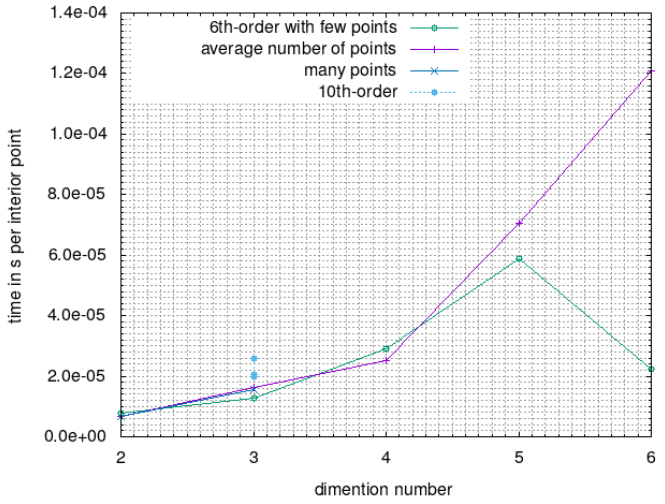


Fig. 4.6 Time per interior point to perform ten iterations of the Multigrid Poisson algorithm when varying the number of points and the dimension.

til 8 to 11 allows to decrease the time of execution. The tenth-order scheme scales better than the tensorial sixth-order scheme because at each Gauss-Seidel iteration this latter needs to visit a node three times with lighter operations when the first one only necessitates one visit with heavier operations. So memory access is more concentrated in the tenth-order case.

The observed cost from Fig. 4.7 for the tenth order scheme, $3.210^{-5} \times 10^8 = 3,200$ operations, is in agreement with the theoretical cost $C = 3,305$ obtained from Eq. (3.2) with $\#B = 113$, $\#A = 59$, $N_{it} = 12$ and $p = 10$.

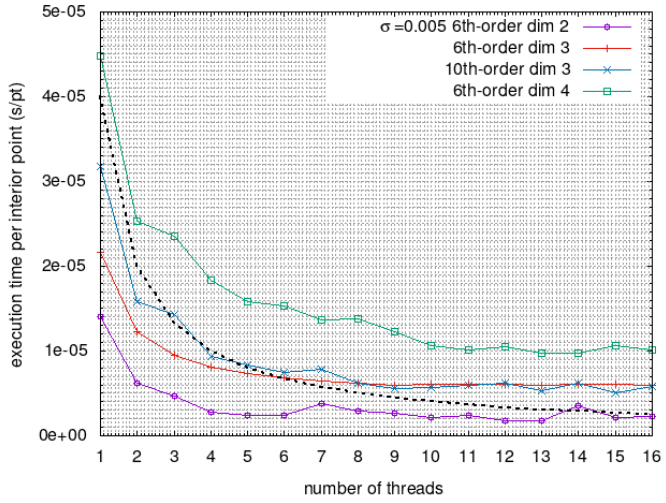


Fig. 4.7 Time per interior point to converge the Multigrid Poisson algorithm to $\|\Delta u_n - \Delta u\| < 10^{-10}\|\Delta u\|$ when varying the number of threads. This tests the strong scaling. The dashed line represents an ideal scaling.

Compared to `p4est` [23], these relatively poor parallelization results show that there is room for improvement.

4.2.3 Initial guess

Starting from an initial guess u_G close to solution u, in the sense $\|\Delta u_G - \Delta u\|$ small compared to $\|\Delta u\|$, allows to decrease the number of iterations necessary to reach convergence. To test this concept and prove its efficiency, we take

$$u_G(\mathbf{x}) = u(\mathbf{x}) + \varepsilon \cos(\xi x_1) \quad \text{so} \quad \Delta u_G(\mathbf{x}) = \Delta u(\mathbf{x}) - \xi^2 \varepsilon \cos(\xi x_1), \tag{4.3}$$

and we look at the number of iterations needed to reach a residual equal to $10^{-10}\|\Delta u\|_\infty$. Starting without any initial guess *i.e.* $u_G = 0$, the algorithm needs exactly 10 iterations with the 6th-order and 10th-order stencils to reach this ending condition.

In order to discard the approximation error, we replace in the definition of u_G Eq. (4.3) the exact solution u by the converged numerical solution of the discrete problem.

For the experiment Fig. 4.8 we used the 3-dimensional Gaussian with $\sigma = 0.005$ as u function, and for both cases, the sixth-order and the tenth-order schemes, the same adaptive grid with $1,139,121$ interior points distributed on scales going from 32^3 to 4096^3 uniform grids equivalent. The computational time comes out approximately the same (approximately 6.5s for 10 iterations with 8 threads) for both cases,

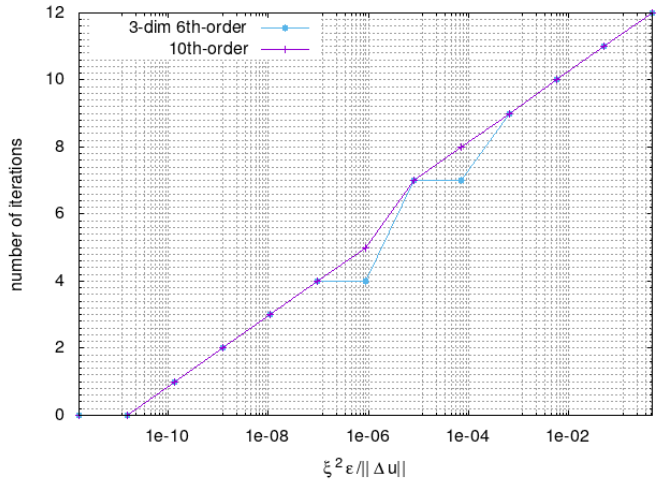


Fig. 4.8 Number of iterations needed to reach a residual equal to $10^{-10}\|\Delta u\|_\infty$ ($\|\Delta u\|_\infty = 2.4\times 10^5$) and stop the iterative process, starting from the initial guess Eq. (4.3). We generate a new point by multiplying ξ by 3. At $\xi^2\varepsilon = 10^{-5}$ we shift ξ from $2187\times 2\pi$ back to 2π and ε from 10^{-8} to 4.8×10^{-2}.

in accordance with Fig. 4.7. The sixth-order scheme ends up with a 3.2×10^{-9} L^∞-error and a 1.1×10^{-9} L^2-error while the tenth-order one ends up with a 9×10^{-12} L^∞-error and a 3.7×10^{-13} L^2-error.

4.3 Oscillating Gaussian test case

The Gaussian test case contains a limited oscillating behavior which does not allow to explore the possibilities of the adaptive refinement. Hence we introduce the *Oscillating Gaussian* test case.

In dimension $2\leq d\leq 6$ we consider the following radial function:

$$F_\varepsilon(r) = e^{-Ar^2}\cos\left(\left(\frac{B}{r^2+\varepsilon}\right)^\alpha\right) \tag{4.4}$$

with $A=150$, $\varepsilon>0$ a parameter which creates oscillations at an approximative distance $r=\sqrt{\varepsilon}$ from $\mathbf{0}$. For $N\in\mathbb{N}^*$ a given number we take $\alpha = -\frac{\log 4N}{\log 4\varepsilon}$ and $B=\frac{\pi^{1/\alpha}}{4\cdot 2^{1/\alpha}}$ so that the function

$$F_\varepsilon(r) = e^{-Ar^2}\cos\left(X(r)\right)$$

with $X(0)=2N\pi$ and $X(\frac{1}{2}) = \frac{\pi}{2}\left(1+4\varepsilon\right)^\alpha \sim \frac{\pi}{2}$ contains N oscillations from the center of the domain to its boundary.

4.3.1 Function analysis

Function $r\mapsto F_\varepsilon(r)$ is C^∞ on $(-0.5,0.5)$ and it is sufficiently small at 0.5 ($\sim 10^{-20}$) so its C^1 discontinuity at the periodic boundary remains numerically benign.

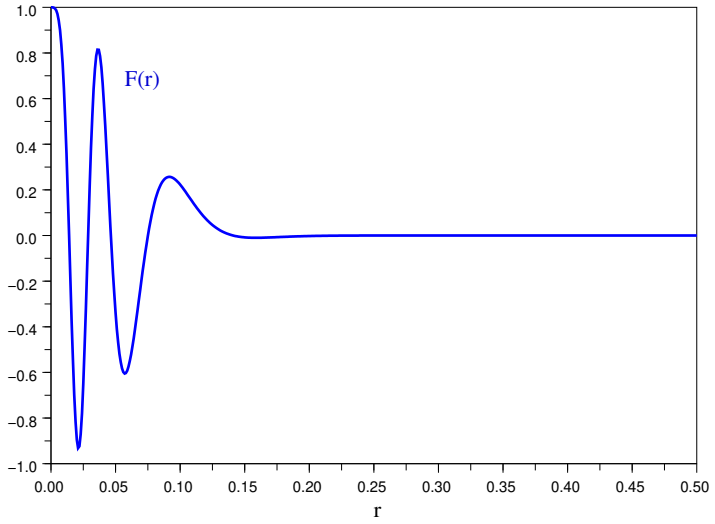


Fig. 4.9 Oscillating Gaussian with $N = 3$ oscillations centered at 0, $(4\varepsilon)^{0.418}/2$ and $(4\varepsilon)^{0.279}/2$.

Suppose $u(\mathbf{x}) = F_\varepsilon(r)$ with $r^2 = \sum_{i=1}^d x_i^2$ then, if $F_\varepsilon(r) = G(r^2)$,

$$v(\mathbf{x}) = \Delta u(\mathbf{x}) = 2d\,G'(r^2) + 4r^2\,G''(r^2).$$

This provides the right hand member for the numerical experiments.

We infer the maximal frequency by studying the variations of the function:

$$g(r) = \left(\frac{B}{r^2+\varepsilon}\right)^\alpha. \tag{4.5}$$

We obtain $g'(r) = -2\alpha r B^\alpha (r^2+\varepsilon)^{-\alpha-1}$ maximal for $r = r_M = \sqrt{\frac{\varepsilon}{2\alpha+1}}$. At this distance to $\mathbf{0}$, $g'(r_M) \sim \frac{2\pi N\alpha}{\sqrt{\varepsilon}}$, the maximal frequency reached by u, provides the smallest scale for the numerical scheme.

4.3.2 Validation of the refinement criterion

The diversity of behaviors displayed by this test function enables to test the refinement criterion. Let us challenge the non-linear approximation theoretical predictions and try to catch the best criteria.

As shown in Fig. 4.3 and 4.4, the present Poisson solver has orders $p = 6$ or $p = 10$ so the L^q-error $\varepsilon_{L^q} = O(h^{-p})$. For an adaptive grid in dimension d the local mesh size $h(\mathbf{x}) = 2^{-j(\mathbf{x})}$ varies as a discontinuous function of the total number of points N_{pt}. It is more or less given by

$$j(\mathbf{x}) = \lfloor \log\left(K_{v,\alpha}(\mathbf{x}) N_{pt}^{-1/d}\right) \rfloor$$

where $K_{v,\alpha}(\mathbf{x})$ is a local constant depending on the right-hand-side function v and on the refinement criterion α.

On the whole domain and in average we would expect a relation of the kind

$$\varepsilon_{L^q} = K_{v,\alpha} N_{pt}^{-p/d}. \tag{4.6}$$

We verify this relation experimentally in Fig. 4.10. The wavelet coefficients d_λ represent the local error of the 4th-order interpolation of the right-hand member v so it is unclear which α parameter should be optimal. The experimental results verify the general trend of Eq. (4.6) although it also shows a slightly chaotic behavior.

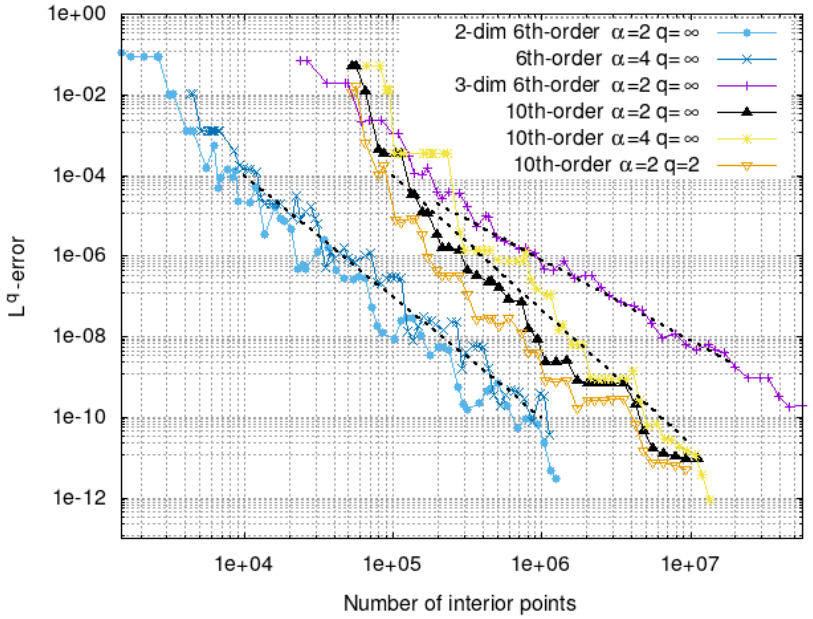


Fig. 4.10 Experimental L^q-error as a function of the number of points for the case $\varepsilon = 10^{-4}$ and $N = 3$ in Eq. (4.4). The theoretical slopes $\varepsilon_{L^q} = K N_{pt}^{-p/d}$ ($\frac{p}{d} = \frac{6}{2}, \frac{10}{3}, \frac{6}{3}$ from left to right) are represented by dashed lines.

In the following we look for an optimal α minimizing the compacity factor $K_{v,\alpha}$ *i.e.* driving to the most efficient non-linear approximation. To assert the study presented in part 2.5, we ran a large number of two-dimensional instances of the sixth-order adaptive Poisson solver with a criterion based on the sixth-order interpolation of the right-hand member v. In Fig. 4.11 we plot the total number of optimal cases when $\alpha = k$ is optimal for various ε values in F, Eq. (4.4), and various numbers of points N_{pt}. It more or less looks like a Gaussian distribution centered on $\alpha = 2$ which is the theoretical value obtained in part 2.5.

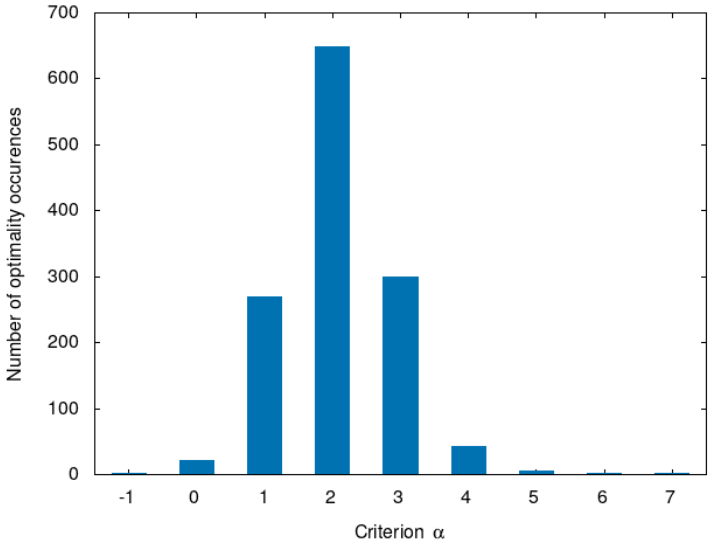


Fig. 4.11 Test of the optimality of the refinement criterion α. Varying ε in F_ε of Eq. (4.4) and the number of points N_{pt} we retained the optimal α providing the smallest value for $K_{v,\alpha}$ in Eq. (4.6). The theoretically optimal value is $\alpha = 2$.

5 Conclusion

The present AMR geometric multigrid Poisson solver proves computationally efficient compared to state-of-the-art modern Poisson solvers and fitted for any order of accuracy and any dimension.

It relies on vertex-centered Cartesian non-uniform grids, although there is no reason why it should not work with cell-based AMR. With cell-based AMR, its rate of convergence which conditions its efficiency should be expected between 0.125, the best vertex-centered case and 0.5, the worse one, *cf* Fig. 3.1 Part 3.4.

Its use of the 10th-order compact scheme developed in [10] demonstrates its ability to increase its order of accuracy. Maybe such an increase could be interesting for delicate simulations such as *ab initio* molecular dynamics [15].

Despite all the experimental efforts, it was possible to demonstrate only a statistical optimal refinement criterion in agreement with the non-linear approximation theory. Thanks to its possibility to start from an initial guess, this geometric solver finds a very efficient use in non stationary equations, such as Navier-Stokes, Schrödinger or Vlasov-Poisson equations. In these time dependent equations an initial guess is provided by temporal extrapolation.

There are quite a few tracks to extend this work:

– the 6th-order scheme combined with a 4th-order immersed boundary method [20] would result in a sixth-order AMR immersed boundary scheme,

- the production of an any-dimensional high-order MPI-parallelized code sounds feasible in a near future,
- there is also the possibility to directly compute the gradient of the potential from the density function as it is done in Fast Multipole Methods [2].

Acknowledgements The author is thankful to Marc Massot and Christian Tenaud for instructive discussions about algebraic and geometric multigrids.

References

1. J. ALBRECHT, *Taylors-Entwicklungen und finite Ausdrücke für Δu und $\Delta\Delta u$*, Z. Angew. Math. Mech. **33**, 48 (1953)
2. T. ASKHAM, A.J. CERFON, *An adaptive fast multipole accelerated Poisson solver for complex geometries*, J. Comput. Phys. **344** 1–22 (2017)
3. M. F. BARAD, P. COLELLA, *A fourth order accurate adaptive mesh refinement method for Poisson's equation*, J. Comput. Phys. **209**(1) (2004)
4. R. BEATSON AND L. GREENGARD, *A short course on fast multipole methods*, *in* Wavelets, Multilevel Methods and Elliptic PDEs, 1–37, Oxford University Press (1997)
5. K. BRIX, S. MELIAN, S. MÜLLER, M. BACHMANN, *Adaptive multi-resolution Methods: Practical issues on Data Structures, Implementation and Parallelization*, ESAIM Proc. **34** 151–183, V. Louvet, M. Massot (eds.) (2011)
6. A. COHEN, R. DEVORE, G. KERKYACHARIAN AND D. PICARD, *Maximal spaces with given rate of convergence for thresholding algorithms*, Appl. Comput. Harmon. Anal. **11** 167–191 (2001)
7. L. COLLATZ, *The numerical treatment of differential equations*, book, 584 pages, Springer-Verlag Berlin (1966)
8. G.-H. COTTET, *Semi-Lagrangian particle methods for high-dimensional Vlasov–Poisson systems*, J. Comput. Phys. **365** 362–375 (2018)
9. D. DEL SARTO, E. DERIAZ, *A multigrid AMR algorithm for the study of magnetic reconnection*, J. Comput. Phys. **351** 511–533 (2017)
10. E. DERIAZ, *Compact finite difference schemes of arbitrary order for the Poisson equation in arbitrary dimensions*, BIT Numer. Math. **60** 199–233 (2020)
11. E. DERIAZ, S. PEIRANI, *Six-Dimensional Adaptive Simulation of the Vlasov Equations Using a Hierarchical Basis*, SIAM Multiscale Model. Simul. **16**(2) 583–614 (2018)
12. E. DERIAZ, V. PERRIER, *Orthogonal Helmholtz decomposition in arbitrary dimension using divergence-free and curl-free wavelets*, Appl. Comput. Harmon. Anal. **26**(2) 249–269 (2009)
13. G. DESLAURIERS, S. DUBUC, *Symmetric iterative interpolation processes*, Constr. Approx. **5**(1) 49–68 (1989)
14. M. DUARTE, Z. BONAVENTURA, M. MASSOT, A. BOURDON, *A numerical strategy to discretize and solve the Poisson equation on dynamically adapted multiresolution grids for time-dependent streamer discharge simulations*, J. Comput. Phys. **289** 129–148 (2015)
15. L. GENOVESE, T. DEUTSCH, S. GOEDECKER, *Efficient and accurate three-dimensional Poisson solver for surface problems*, J. Chem. Phys. **127**, 054704 (2007)
16. A. GHOLAMI, D. MALHOTRA, H. SUNDAR, AND G. BIROS, *FFT, FMM, or Multigrid? A comparative Study of State-Of-the-Art Poisson Solvers for Uniform and Nonuniform Grids in the Unit Cube*, SIAM J. Sci. Comput. **38**(3) C280–C306 (2016)
17. L. GREENGARD, J.-Y. LEE, *A Direct Adaptive Poisson Solver of Arbitrary Order Accuracy*, J. Comput. Phys. **125**(2) 415–424 (1996)
18. W. HACKBUSCH, *Multi-grid methods and applications*, book, 378 pages, Springer-Verlag (1985)
19. M. M. HEJLESEN, J. T. RASMUSSEN, P. CHATELAIN, J. H. WALTHER, *A high order solver for the unbounded Poisson equation*, J. Comput. Phys. **252** 458–467 (2013)
20. S. HOSSEINVERDI, H.F. FASEL, *An efficient, high-order method for solving Poisson equation for immersed boundaries: Combination of compact difference and multiscale multigrid methods*, J. Comput. Phys. **374** 912–940 (2018)
21. J.-P. KAHANE, P.-G. LEMARIÉ-RIEUSSET, *Fourier Series and Wavelets*, book, 368 pages, Gordon & Breach Science Publishers Ltd (1995)

22. M. KRONBICHLER, T. HEISTER AND W. BANGERTH, *High accuracy mantle convection simulation through modern numerical methods*, Geophys. J. Int. **191** 12–29 (2012)
23. T. ISAAC, C. BURSTEDDE, L. C. WILCOX, O. GHATTAS, *Recursive Algorithms for Distributed Forests of Octrees*, SIAM J. Sci. Comput. **37**(5) C497–C531 (2014)
24. A. ISERLES, *A First Course in the Numerical Analysis of Differential Equations*, book, 393 pages, Cambridge University Press (1996)
25. A. MCKENNEY, L. GREENGARD, A. MAYO, *A Fast Poisson Solver for Complex Geometries*, J. Comput. Phys. **118**(2) 348–355 (1995)
26. F. MINIATI, P. COLELLA, *Block structured adaptive mesh and time refinement for hybrid, hyperbolic+N-body systems*, J. Comput. Phys. **227**(1) 400–430 (2007)
27. S. POPINET, *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries*, J. Comput. Phys. **190**(2) 572–600 (2003)
28. S. SCHAFFER, *Higher Order Multi-Grid Methods*, Math. Comput. **43**(167) 89–115 (1984)
29. W. F. SPOTZ AND G. F. CAREY, *High-order compact finite difference methods*, in book, Proceedings of ICOSAM'95, 397–408 (1995)
30. R. TEYSSIER, *Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES*, Astron. Astrophys. **385** 337–364 (2002)