Commun. Comput. Phys.
doi: 10.4208/cicp.260514.101214a

Vol. **18**, No. 5, pp. 1211-1233
November 2015

High Order Finite Difference Discretization for Composite Grid Hierarchy and Its Applications

Qun Gu[1], Weiguo Gao[2,*] and Carlos J. García-Cervera[3]

[1] *School of Mathematical Sciences, Fudan University, Shanghai 200433, P.R. China.*
[2] *MOE Key Laboratory of Computational Physical Sciences and School of Mathematical Sciences, Fudan University, Shanghai 200433, P.R. China.*
[3] *Department of Mathematics, University of California, Santa Barbara, CA 93106, USA.*

Received 26 May 2014; Accepted (in revised version) 10 December 2014

Abstract. We introduce efficient approaches to construct high order finite difference discretizations for solving partial differential equations, based on a composite grid hierarchy. We introduce a modification of the traditional point clustering algorithm, obtained by adding restrictive parameters that control the minimal patch length and the size of the buffer zone. As a result, a reduction in the number of interfacial cells is observed. Based on a reasonable geometric grid setting, we discuss a general approach for the construction of stencils in a composite grid environment. The straightforward approach leads to an ill-posed problem. In our approach we regularize this problem, and transform it into solving a symmetric system of linear of equations. Finally, a stencil repository has been designed to further reduce computational overhead. The effectiveness of the discretizations is illustrated by numerical experiments on second order elliptic differential equations.

AMS subject classifications: 35J57, 47A52, 65M50, 65N06

Key words: High order, point clustering algorithm, finite difference method, composite grid, stencil, regularization.

1 Introduction and preliminaries

Adaptive mesh refinement (AMR) is a simple and popular framework to reduce the computational overhead in dealing with large scale modern scientific computational problems. It is commonly used in a number of research areas, and there are several software packages available that implement AMR to solve problems on different scales. For

*Corresponding author. *Email addresses:* 081018018@fudan.edu.cn (Q. Gu), wggao@fudan.edu.cn (W. Gao), cgarcia@math.ucsb.edu (C. J. García-Cervera)

example, APBS [2, 16] uses multilevel adaptive finite elements for solving the Poisson-Boltzmann equation in order to study the chemical properties of complex molecules or polymers and their microscopic behavior with implicit solvents. Macroscopic applications can be found in ENZO [7, 23], designed for multi-physics simulations in astrophysics and cosmology. General purpose packages such as CHOMBO [10], developed by a team at Lawrence Berkeley National Laboratory (LBNL), is designed for the solution of multidimensional elliptic equations and time-dependent problems. Packages such as PARAMESH [21] and AGRIF [12] are similar tools that have been developed for solving partial differential equations (PDEs) in a parallel environment.

Based on an adaptive mesh hierarchy, there are several approaches for transforming the partial differential equations into their discrete counterpart. Attempts that incorporate classical discretization methods like finite difference method (FDM), the finite element method (FEM), or the finite volume method (FVM) within the AMR framework have been very successful. It was introduced by Berger and Oliger for the study of hyperbolic partial differential equations using an adaptive finite difference method (AFDM) [5]. This same discretization framework was used by Berger and Colella to carry out simulations of hydrodynamic shocks [3]. More recently, this approach has been used for the study of the phase separation process using the Cahn-Hilliard equations [8], for the study of multiphase incompressible flows [9], and for simulating solid tumor growth [27], to name just a few examples. This approach also has a long history within the finite element community. It is a standard tool in elasticity theory computations, specially in the study of solid fracture [25]. In recent years it has also proved to be a very effective tool in electronic structure computations, within the orbital-free [15] or the Kohn-Sham [29] approaches to density functional theory. The finite volume method could also be formulated within an adaptive mesh refinement framework. In [19], the incompressible heat flow problem was studied using an adaptive finite volume method (AFVM); in [17] the authors investigated the multi-phase flow and transport in porous media based on the same framework; and in [28, 31] the authors developed an adaptive coupled level-set/volume-of-fluid (ACLSVOF) method for interfacial flow simulations on unstructured triangular grids.

High order numerical schemes are also a trend for development of high performance solvers and are necessary to simulate complex and large systems. There is also a large body of literature on the subject. For instance, in [20] the authors reported a parallel implementation of a solver for the Poisson-Boltzmann equation with periodic boundary conditions using a sixth order finite difference scheme. The three dimensional sixth order scheme is basically a Cartesian product of three one dimensional sixth order stencils. An alternative approach to implement and achieve high order accuracy can be found in compact schemes. For compact finite difference methods, explicit formulas have been presented in earlier works. A compact fourth order scheme can be found in [30], while a sixth order compact scheme is applied in [26]. Note that the sixth order compact discretization relies on derivatives of the right hand side. It is well known that compact schemes higher than sixth order on uniform grids do not exist, unless derivatives of the

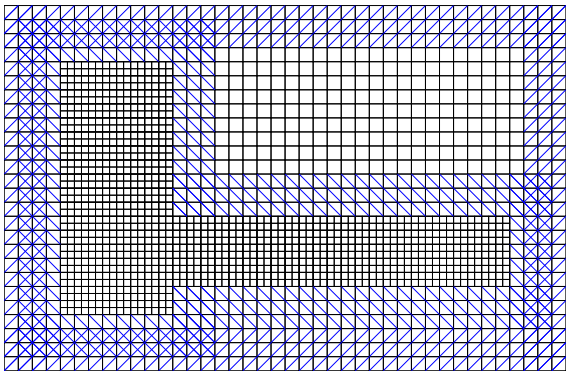


Figure 1: A two-level example of composite grid in two dimension. This figure is also used to define the classifications of cells. Imagine that every cell has a label, then interior region is labeled by nothing, the outer interface cells are labeled by '/', inter interface cells by '\', the overlapping one '×', and the redundant one '+'.

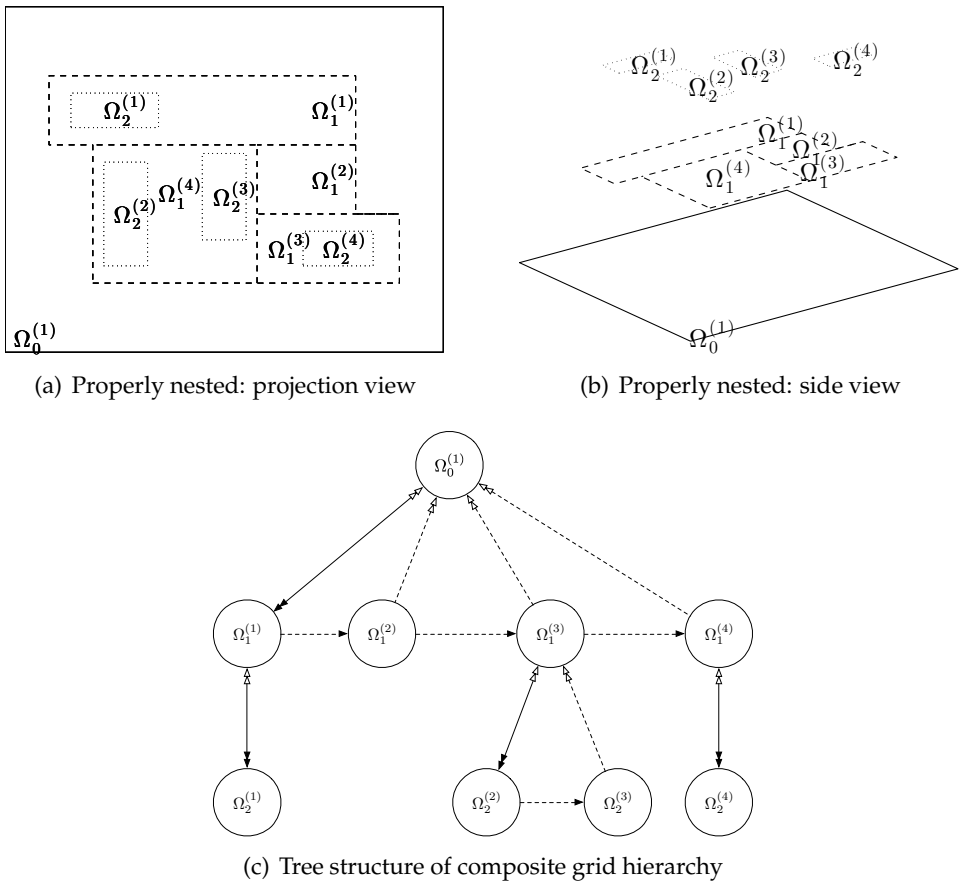
right hand side are available[†]. General purpose high order finite difference stencil creators are available, but mainly in one dimension [18]. In [14] Fornberg proposed a finite difference scheme creator that could approximate arbitrary orders of derivatives. These formulas are derived for an arbitrary spaced grid in one dimension in a recursive way. Other general purpose high order discretizations usually appear in the framework of the finite element method. In this paper, we will present a general finite difference discretization framework for some local differential operators that can be utilized on an composite grid hierarchy, up to any order of accuracy.

To motivate our discussion, we start with a description of some basic concepts in the composite grid framework. Throughout our discussion, we will use two dimensional examples to illustrate the setting. However, the reader should keep in mind that the three dimensional case is just a straightforward generalization. We utilized the block-structured composite grid, first introduced by Berger and Oliger for the solution of hyperbolic PDEs [5]. Note also that for convenience in the indexing we employ a cell-centered scheme [4]. An example of a composite grid is shown in Fig. 1. In this figure, we see two levels of grids. In general, we deal with $\ell_{\max}$ levels of grids. Level $\ell=0$ is always the coarsest level, which covers the entire domain Ω. Finer grids are built upon the coarsest one in a tree structure. For simplicity, we restrict ourselves to the case of equal step length. Denote the finest grid step length as h. Then, *the ℓ-th level of grids* Ω^ℓ is defined by the union of all the sub-grids that have step length $2^{\ell_{\max}-\ell}\cdot h$. We follow the conventional assumptions [22] that the grids are *properly nested*, that is we have

$$\mathcal{R}(\mathcal{P}(\Omega^\ell))=\Omega^\ell,$$
$$\mathcal{P}(\Omega^{\ell+1})\subset\Omega^\ell,$$

where $\mathcal{P}$ is the projector from level ℓ to level $\ell-1$, and $\mathcal{R}$ is the restriction operator that maps level ℓ to level $\ell+1$. Figs. 2(a) and 2(b) illustrate the properly nested property. In

[†]Here, compact means we only allow nearest grid points (or cells)



(a) Properly nested: projection view

(b) Properly nested: side view

(c) Tree structure of composite grid hierarchy

Figure 2: Properly nested property and tree structure of composite grid hierarchy.

the next section, we will describe how to perform local refinements of the grids. This composite grid hierarchy is organized in a tree structure, which is shown in Fig. 2(c). For further details, we refer to [32].

The remainder of the paper is organized as follows: In Section 2, we describe a modified point clustering algorithm that is suitable for high order finite difference discretization on a composite grid hierarchy. Section 3 is devoted to solving the stencil problems for the high order finite difference schemes, including the introduction of the pre-selection area, the existence and uniqueness of the resulting linear system, and the coefficients calculator. We illustrate our approach with several numerical experiments in Section 4, and finish with some concluding remarks in Section 5.

2 Modified point clustering algorithm

Our starting point is the point clustering algorithm proposed by Berger and Rigoutsos in [6]. As we will see in the next section, constructing the special (high order) finite differ-

ence schemes for interface points requires considerable overhead, so we hope to reduce the number of cells in the interfacial region. To begin with, we discuss the classification of grid points. In our implementation, we classify the cells into five classes:

1. *Interior region*. In this region, we ensure that standard finite difference (long) schemes are directly applicable and there is no *search* or *solve* computational cost here.

2. *Inner interface*. For the cells at the inner interface, we need to construct finite difference stencils that are (at least) related to cell(s) on a finer level. Note that the inner interface could overlap with the outer interface, as defined below. We save the overlapping information at overlapping interfaces.

3. *Outer interface*. In this region, finite difference stencils need to be constructed according to the exterior environment of this grid. It could happen that the stencil will touch the physical domain boundary or touch another grid (either coarser or finer or the same level, both situations could happen) or itself, if the boundary condition is periodic. Note, again, that we also subtract the overlapping region.

4. *Overlapping interface*. This is the overlapping region. Special stencils need to be constructed for each cell.

5. *Redundant region*. These points do not belong to the set of degrees of freedom (DOF) on the composite grid. Hence, there is no need to construct any scheme here.

One observation is that using the original algorithm in [6], this division process can result in extremely small sub-grids, which will bring complexity for high order discretizations. As we will see in Section 3, the design for these special stencils is a non-trivial and time-consuming process. Thus, we hope to reduce the number of interface cells.

In our current implementation, two natural ways could be helpful to overcome this difficult, that is, to add some restrictions on the clustering algorithm by introducing the concept of *minimal length* of a patch and the *buffer zone*. Algorithm 1 describes our methodology.

The introduction of buffer zone is straightforward. Specifically, consider a grid with some flags (*i.e.*, the refinement labels) defined on it. We simply remove those flags on the outer layer of the grid. The resulting patch will be the initial patch that drives the original clustering algorithm.

For the minimal length restriction, one has to be careful in the splitting-for-signature step in the clustering iteration. A quick return means that for those patches whose length is already smaller than or equal to the minimal length restriction, we have no need to split them again. As in the original algorithm, we prioritize the search for the zero-cut opportunity. If there is no hole in the signature array, we try to find out the cut that has the maximum discrete Laplacian jump (which means between these two points, there is a zero, and the jump amplitude is the deepest). One needs to verify that the resulting sub-patches will still meet the minimal length restriction.

Algorithm 1 Modified Berger-Rigoutsos Point Clustering Algorithm

Require: Buffer zone length, minimal patch length, patch efficiency threshold τ;
Ensure: A series of patches;
1: Set current patch indicator $i \leftarrow 1$ and allocate a queue of patches;
2: Push the *buffer-zone-free* flagged grid into the queue;
3: **while** $i \leq$ length of patch queue **do**
4: **if** efficiency of patch$(i) \geq \tau$ and patch(i) is flag-free **then**
5: $i++$ and **continue**;
6: **end if**
7: Compute signature for each direction, *i.e.*, $\int\int \texttt{flag}(\mathbf{x}) \mathrm{d}x_j \mathrm{d}x_k$;
8: Search a split position *with minimal length take into account*;
9: **if** no where to split **then**
10: $i++$ and **continue**;
11: **end if**
12: Split the patch into two sub-patches and append two sub-patches into the queue;
13: **end while**

In order to illustrate the effectiveness of the minimal length parameter, we consider a two dimensional example defined on a unit square with a 100-by-100 meshing and an "S"-shaped flag setting imposed on it, as is shown in Fig. 3.

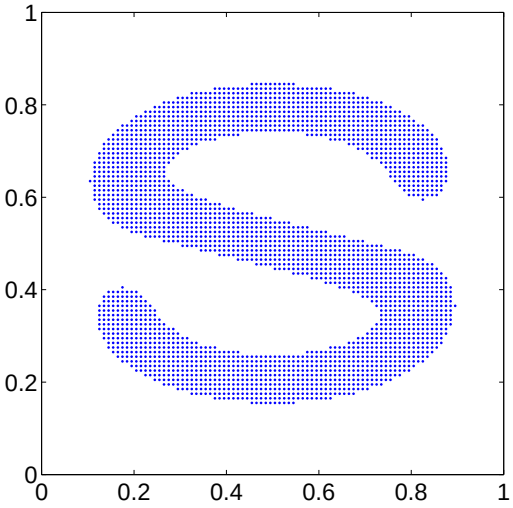
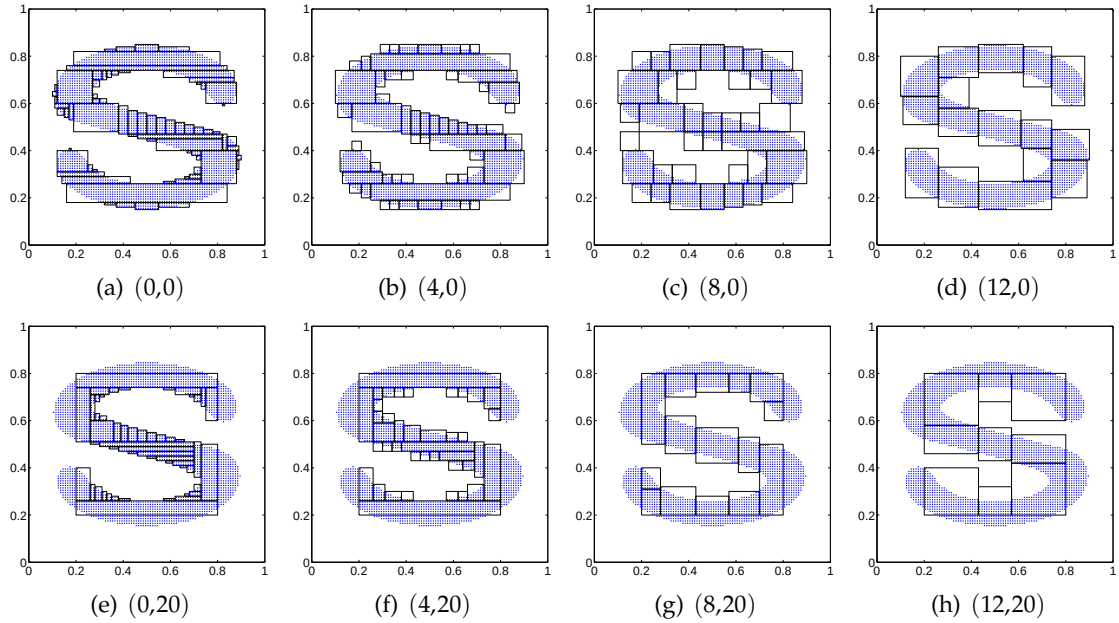


Figure 3: Flag setting.

We test four different minimal length settings, *viz*, $\texttt{minlen} = 0,4,8,12$. The result of each case is shown in Figs. 4(a), 4(b), 4(c), and 4(d), respectively. Also, we tested these four different minimal length settings under the buffer zone $\texttt{bfzone} = 20$, shown in Figs. 4(e), 4(f), 4(g), and 4(h).



(a) (0,0) (b) (4,0) (c) (8,0) (d) (12,0)

(e) (0,20) (f) (4,20) (g) (8,20) (h) (12,20)

Figure 4: Output of the point clustering algorithm with different setting of `minlen` (the first parameter) and `bfzone` (the second one). Figs. 4(a) is the result by the original Berger and Rigoutsos algorithm. Figs. 4(b), 4(c), and 4(d) are three results under the minimal length restriction, Figs. 4(e), 4(f), 4(g), and 4(h) are results with buffer zone length equals to 20.

The statistics for the number of points is summarized in Table 1. As we can see, when `minlen` increases, we are allowed to deal with fewer interface points, which leads to an increase in the number of degrees of freedom. When we impose a thick buffer zone, the number of interface points and of interior cells drop drastically. The risk is that some of the flagged points are not going to be refined, which might downgrade the accuracy of the solver.

Table 1: Statistics for point clustering algorithm with adjustable parameters.

(`minlen`, `bfzone`)	(0,0)	(4,0)	(8,0)	(12,0)
# Interface cells	20999	19821	17408	14913
	N/A	(−5.94%)	(−17.10%)	(−28.98%)
# Interior cells	25799	26655	28519	31102
	N/A	(+3.32%)	(+10.54%)	(+20.56%)
(`minlen`, `bfzone`)	(0,20)	(4,20)	(8,20)	(12,20)
# Interface cells	12172	11304	8696	6756
	(−42.04%)	(−46.17%)	(−58.59%)	(−67.83%)
# Interior cells	10089	10742	13176	14473
	(−60.89%)	(−58.36%)	(−48.93%)	(−43.90%)

We will see in Section 3.1 that the restrictions on minimal patch length and buffer zone length not only gives us practical CPU time savings in setting stencils steps, but also are of theoretical importance in ensuring breakdown in pre-selection area enlargement process.

3 Stencil generation

After running the point clustering algorithm, the geometric setting of the composite grid hierarchy is fixed. The next central problem is to design special computational scheme for each cell *at interfacial region*. To do so, we introduce the concept of stencil is used to in constructing such computational schemes [24].

Definition 3.1. Let $\mathbf{x}_0$ be the center of the cell at level ℓ in the composite grid hierarchy and let $\mathcal{L}u(\mathbf{x}_0)$ be the quantity that is to be approximated by a linear combination $\sum_{j=1}^{N} c_j u(\mathbf{x}_j)$, where $\{\mathbf{x}_j\}_{j=0}^{N}$ are cell centers chosen from composite grid hierarchy. We call $\mathsf{X}=\{\mathbf{x}_j\}_{j=1}^{N}$ the *stencil positions* and $\mathbf{c}=\{c_j\}_{j=1}^{N}$ the *stencil coefficients*. We say that the triplet $(\mathbf{x}_0,\mathsf{X},\mathbf{c})$ is a *stencil at* $\mathbf{x}_0$. The *local truncation error* is defined by

$$\tau=\mathcal{L}u(\mathbf{x}_0)-\sum_{j=1}^{N} c_j u(\mathbf{x}_j).$$

Set the *scaled stencil position* $\boldsymbol{\delta}_j=\frac{1}{h_\ell}(\mathbf{x}_j-\mathbf{x}_0)$. If

$$\tau=\mathcal{O}(h_\ell^m), \tag{3.1}$$

we say that $(\mathbf{x}_0,\mathsf{X},\mathbf{c},m)$ is a *mth-order stencil at* $\mathbf{x}_0$.

For a one dimensional problem, finite difference formulas on arbitrarily spaced grids can be found in [14]. A stencil adaptive algorithm is proposed by [13] for incompressible viscous flows in two dimensional space with a refinement ratio $\sqrt{2}$. However, there is no literature found for arbitrary dimensional stencil construction in a composite grid setting.

For simplicity, we mainly focus on the generation of stencils to approximate the Laplace operator. For Laplacian is a second order differential operator, we denote the *scaled stencil coefficients* as $\boldsymbol{\omega}\equiv\{\omega_i\}_{i=1}^{N}=\mathbf{c}h^2$. Note that our approach can also be applied to more general differential operators. Note also that, without loss of generality, we set $\mathbf{x}_0$ as $\mathbf{0}$. We build up the mathematical formulation of the stencil generation problem.

Applying a Taylor expansion to $(m+1)$-nd order on (3.1) with remainder expressed in the Peano form gives us

$$\Delta u(\mathbf{0})-\sum_{i=1}^{N}\sum_{|\boldsymbol{\alpha}|=0}^{m+1}\frac{\boldsymbol{\delta}_i^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\partial^{\boldsymbol{\alpha}}u(\mathbf{0})\omega_i h^{|\boldsymbol{\alpha}|-2}=\mathcal{O}(h^m),$$

where the standard multi-index notation is applied in $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_d)$ and the standard multi-index operations are also assumed. In matrix form, we have

$$\begin{bmatrix} \delta_i^{\boldsymbol{\alpha}}/\boldsymbol{\alpha}! \\ \hline \text{H.O.T.} \end{bmatrix} \begin{bmatrix} \\ \boldsymbol{\omega} \\ \\ \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \sum_{j=1}^{d}\mathbf{e}_{i_j} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \hline \mathcal{O}(h^m) \end{bmatrix} \begin{array}{ll} \to \text{coeff. of } u^{(\boldsymbol{\alpha})}h^{-2}, & |\boldsymbol{\alpha}|=0, \\ \to \text{coeff. of } u^{(\boldsymbol{\alpha})}h^{-1}, & |\boldsymbol{\alpha}|=1, \\ \to \text{coeff. of } u^{(\boldsymbol{\alpha})}h^{0}, & |\boldsymbol{\alpha}|=2, \\ \to \text{coeff. of } u^{(\boldsymbol{\alpha})}h^{1}, & |\boldsymbol{\alpha}|=3, \\ \vdots & \\ \to \text{coeff. of } u^{(\boldsymbol{\alpha})}h^{m-1}, & |\boldsymbol{\alpha}|=m+1, \\ \to \text{H.O.T.} & \end{array} \tag{3.2}$$

where i_j stands for the rows that correspond to the derivatives of $u^{(2\mathbf{e}_j)}$. Dropping the higher order $\mathcal{O}(h^m)$ in (3.2) and writing in compact form, we arrive at the governing equation for stencil generation

$$\mathsf{A}\boldsymbol{\omega}=\mathbf{b}, \tag{3.3}$$

where

$$\mathsf{A}\in\mathbb{R}^{M\times N}, \quad \boldsymbol{\omega}\in\mathbb{R}^{N\times 1}, \quad \mathbf{b}\in\mathbb{R}^{M\times 1},$$

with

$$M=\sum_{|\boldsymbol{\alpha}|=0}^{m+1}\binom{|\boldsymbol{\alpha}|+d-1}{d-1}=\binom{m+d+1}{d}.$$

The central problem is to specify the set of scaled stencil position $\{\delta_j\}$ and their corresponding coefficients $\{\omega_j\}$ so that (3.3) holds.

It is obvious that the smaller the stencil, the less computational cost required. Hence, our stencil generator resorts to the following steps.

Step 1. Generate a "pre-selection area" of "selection radius" r on the composite grid and compute $\{\delta_j\}$ according to the pre-selection area.

Step 2. Compute $\{\omega_j\}$.

Step 3. Check accuracy requirement. If not satisfied, go to Step 1 with $r=r+1$.

We discuss the details of pre-selection area in Section 3.1. Once the positions is specified, in Section 3.2, we investigate the existence and uniqueness of the solution of (3.3). It turns out that this problem is generally ill-posed. We resort to regularization method to establish an effective coefficient calculator in Section 3.3.

3.1 Constructing the shape of the stencil on the composite grid by pre-selection area

It is a seemingly easy yet non-trivial task to find a set of candidate cells located at $\{\delta_j\}$, that is *suitable* to approximate the Laplacian at a certain position, especially for high order approximation in the composite grid environment. This point selection process for a lower (e.g. second) order discretization is relatively easy, for the number of possible stencil configurations is dramatically smaller than it is the high order case. Incrementing the dimensionality greatly increases the complexity as well. An additional difficulty comes from the possible complexity caused by the structure of the grid. In fact, there is no general principle for this cell selection process.

Suppose we are constructing the stencil at $\mathbf{x}_0$ which is a level ℓ cell. First, we generate a *pre-selection area of selection radius* r, which is defined by the union of a set of d-dimensional hypercubes whose centers are located at $\{\mathbf{x}_0+\mathbf{w}_i h_\ell\}$, where $\mathbf{w}_i$ runs over the set

$$\left\{\mathbf{w}:\|\mathbf{w}\|_1\leq r, w_j\in\mathbb{Z}\right\},$$

and whose edge lengths are of h_ℓ.

The key step is to find out the corresponding cells on the composite grid hierarchy for each hypercube in the pre-selection area. Results are saved in an output list. Note that special attentions should be paid according to boundary conditions. For each hypercube, the possible situations are listed below.

Case 1. The hypercube is covered by level ℓ or a coarser level. In this case, we only need to append the hypercube or coarser cell into the output list.

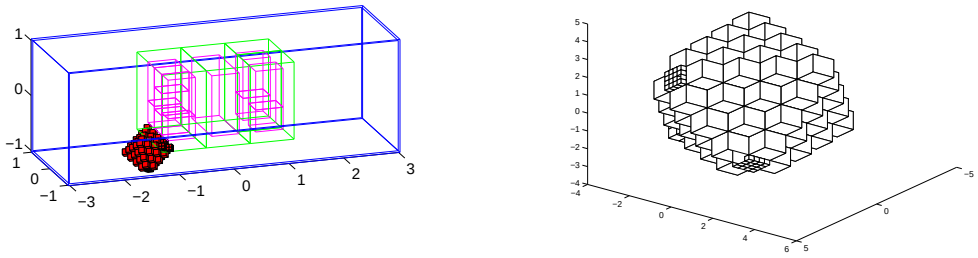
Case 2. A finer level of grid is covering the hypercube. We use a recursive approach to append information to the output list by sub-dividing the cell into 2^d patches.

Case 3. If the hypercube is outside the DOF region, there could be two possibilities.

(a) The hypercube is contains the cells that could be defined by Dirichlet boundary condition: we need to append the cell into the list (and discard the redundant ones if necessary).

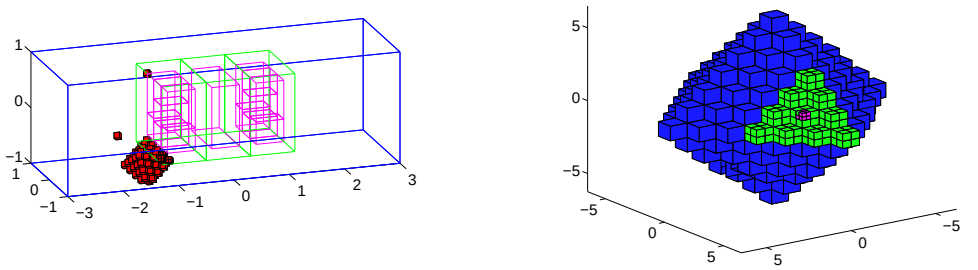
(b) The hypercube is outside the DOF region, but the periodic boundary condition will pull the hypercube back into the DOF region. In this case, Case 1 and Case 2 are applied.

We present some examples of the pre-selection area with differential boundary conditions. Figs. 5(a) and 5(b) are for Dirichlet boundary condition, while Figs. 6(a), and 6(b) are for periodic ones. The left figures show the pre-selection area in geometrically and the right ones, theoretically.



(a) Geometrical plot (b) Theoretical plot

Figure 5: Sample pre-selection result. Radius is set as 4 and the boundary condition is set as Dirichlet.



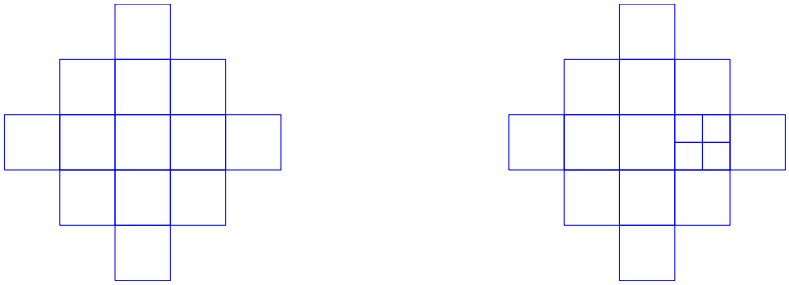
(a) Geometrical plot (b) Theoretical plot

Figure 6: Sample pre-selection result. Radius is set as 4 and the boundary condition is set as Dirichlet.

3.2 Existence and uniqueness of (3.3)

In the last subsection, we discuss the strategy to choose cells from a composite grid by pre-selection area. However, one important issue is that it is not obvious that enlargement of selection radius r will ensure the existence of a solution $\boldsymbol{\omega}$ that satisfies (3.3), especially under complex composite grid setting.

The existence of the stencil is related with the selection radius. Obviously, the larger the radius, the more degrees of freedom we have in the construction of a stencil. However, it is not entirely determined by the selection radius r, it could also depend on the specific structure of the local grid(s). Consider a two dimensional fourth order stencil construction with pre-selection radius equals to 2, but with two different grid settings shown in Fig. 7. In the first setting shown in Fig. 7(a) which is a two dimensional fourth order stencil construction problem. There exists a unique stencil, for the reason that $\text{rank}(\mathsf{A}) = \text{rank}\left(\left[\mathsf{A} \quad \mathbf{b}\right]\right) = 13$. However, under Setting 2 in Fig. 7(b), we have, the rank of coefficient matrix A is 16 and the rank of augmented matrix is 17. Hence there is no possible solution.



(a) Setting 1 (b) Setting 2

Figure 7: Different settings of initial candidate cells selected by a pre-selection area of radius 2.

The following theorem ensures that the increment of r will sure to breakdown under a mild restrictions on the structure of the composite grid hierarchy.

Theorem 3.1. *For any accuracy requirement m and any d-dimensional cell-centered composite grid hierarchy, generated by the modified point clustering algorithm with minimal length larger than $m+2$ and buffer zone length larger than 1, and given a cell is on level ℓ. Then, for selection radius*

$$r \geq 2^{\ell} \cdot (m+2),$$

there exists a $\boldsymbol{\omega}$ that satisfies (3.3).

Proof. It is enough to prove that, for any dimension $i \in \{1,\cdots,d\}$, there exists an index $\mathcal{J} \subset \{1,\cdots,N\}$ such that

$$\frac{1}{h^2}\sum_{j\in\mathcal{J}} \omega_j u(\mathbf{x}_j) = -\frac{\partial^2}{\partial x_i^2} u(\mathbf{x}_0) + \mathcal{O}(h^{m+1}). \tag{3.4}$$

By the minimal length restriction, the existence of buffer zone length and the theory of polynomial interpolation, there exists a index set $\mathcal{J}_k \subset \{1,\cdots,N\}$, such that for each $\mathbf{x}_0 + kh_0\mathbf{e}_i$, we have

$$u(\mathbf{x}_0 + kh_0\mathbf{e}_i) = \sum_{j\in\mathcal{J}_k} p_j^{(k)} u(\mathbf{x}_j) + \mathcal{O}(h^{m+1}), \tag{3.5}$$

where $\{p_j^{(k)}\}$ are the coefficients of interpolation. Furthermore, it is well known that for $m+2$ distinct points, there exists a m-th order finite difference scheme that approximates the Laplace operator. The worst case is that $m+2$ points are all located at one size of $\mathbf{x}_0$. Hence for $r \geq 2^{\ell}\cdot(m+2)$, the stencil could be constructed by polynomial interpolation. □

Remark 3.1. Theorem 3.1 ensures that there exists $\boldsymbol{\omega}$ such that when given large enough pre-selection area, Eq. (3.3) holds.

Uniqueness of (3.3) is generally not guaranteed. In fact, we have the following proposition.

Proposition 3.1. Matrix A in (3.3) may not necessarily have full column rank, even if A is a tall or square matrix, that is $M \leq N$. Hence the stencil generation problem could be an ill-posed problem.

Proof. The proof is constructive. Consider a $d=2$ dimensional stencil generation problem with the standard long central difference points, *i.e.*, $\{\mathbf{x}_i\}=\{(n_1,n_2): n_1 \in [-2,2]\cap\mathbb{Z}, n_2 = 0\}\cup\{(n_1,n_2): n_2 \in [-2,2]\cap\mathbb{Z}, n_1 = 0\}$ and the accuracy requirement is 2. The matrix A takes the form

$$\mathsf{A}=\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & -1 & 0 & 2 & 0 & -2 \\
0 & 1 & 0 & -1 & 0 & 2 & 0 & -2 & 0 \\
0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 2 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 2 & 0 & 2 & 0 \\
0 & 0 & \frac{1}{6} & 0 & -\frac{1}{6} & 0 & \frac{4}{3} & 0 & -\frac{4}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{6} & 0 & -\frac{1}{6} & 0 & \frac{4}{3} & 0 & -\frac{4}{3} & 0
\end{bmatrix}_{10\times 9}$$

with $\mathrm{rank}(\mathsf{A})=7$. □

Remark 3.2. Proposition 3.1 implies that uniqueness of (3.3) is generally not guaranteed. This could also be understood intuitively, since if we are given too many candidate points, it is possible to construct different stencils while preserving the required accuracy. In order to solve this problem, we rely on some regularization techniques in the next subsection.

3.3 Effective coefficients via regularization

As is discussed in the last subsection, when pre-selection area is large enough, the problem becomes under-determined. A common treatment for such ill-posed problems is utilizing the regularization techniques. However, the choice of optimization function is not unique. Hence, the first trial is the standard Tikhonov regularization.

($\widetilde{\mathbf{P1}}$) *Tikhonov regularized stencil generation problem*

$$\min_{\boldsymbol{\omega}} \|\mathsf{T}\boldsymbol{\omega}\|_2, \qquad \text{s.t. } \mathsf{A}\boldsymbol{\omega}=\mathbf{b}, \tag{3.6}$$

where T is the Tikhonov matrix, and a simple usual choice is the identity matrix. Eq. (3.6) can be directly transformed into a linear system of equations by introducing a Lagrange multiplier.

(**P1**) *Explicit solution to Tikhonov regularized stencil generation problem*

$$\begin{bmatrix} 2\mathsf{T}^{\mathrm{T}}\mathsf{T} & \mathsf{A}^{\mathrm{T}} \\ \mathsf{A} & \mathsf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}. \tag{3.7}$$

This symmetric linear system can be solved directly by using a `DSYSV` routine in LAPACK [1]. In this way, we are able to construct the stencil for any cell on the composite grid for any order, provided enough grid points are available, by solving (**P1**) or ($\widetilde{\mathbf{P1}}$). Note that in the process of solving, we essentially find a solution with the minimal $\mathsf{T}^{\mathrm{T}}\mathsf{T}$-norm.

Remark 3.3. This regularization technique could be applied to the uniform grid as well. We observe that in 2D and 3D case, for radius of pre-selection area small r (from 1 to 10), solving this problem will result in a $2r$ order standard long finite difference scheme. However, this is not generally true for arbitrary r.

To motivate our implementation, we use examples to illustrate our modifications on this problem. The first observation is that if $\mathsf{T}=\mathsf{I}$, the enlarged linear system may be ill-conditioned. For example, if we present a walk list on a uniform grid with walk radius 3, the condition number in (3.7) equals to $6.0577\times10^{+34}$, which indicates a severally ill-conditioned problem. The solution of (3.7) results in a stencil with absolutely no accuracy. One way to overcome this difficulty is to change to another Tikhonov regularization matrix. In our implementation, we choose $\widetilde{\mathsf{T}}=\mathsf{I}-\mathsf{E}_{11}$. Then we arrive at the second version of the regularized problem:

($\widetilde{\mathbf{P2}}$) *Second version of regularized problem*

$$\min_{\boldsymbol{\omega}} \|\widetilde{\mathsf{T}}\boldsymbol{\omega}\|_2, \qquad \text{s.t. } \mathsf{A}\boldsymbol{\omega}=\mathbf{b}, \tag{3.8}$$

where $\mathsf{E}_{11}=\mathbf{e}_1\mathbf{e}_1^{\mathrm{T}}$ is the elementary matrix with a one entry on $(1,1)$-position. The motivation for this choice is that the solution of ($\widetilde{\mathbf{P2}}$) is equivalent to minimizing the off-diagonal part of energy in the stencils. Again, Eq. (3.8) could be transformed into a linear system of equations

(**P2**) *Explicit solution to second version of stencil generation problem*

$$\begin{bmatrix} 2\widetilde{\mathsf{T}}^{\mathrm{T}}\widetilde{\mathsf{T}} & \mathsf{A}^{\mathrm{T}} \\ \mathsf{A} & \mathsf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}. \tag{3.9}$$

After solving (3.9), we recover three dimensional long scheme, that is a direct sum of three one dimensional sixth order stencils in three directions as expected.

The second observation is that we could end up with a lot of different stencils for a single cell (or grid point) with the same order of local truncation error. Suppose we are given a stencil of accuracy order m, then the local truncation error is of $\mathcal{O}(h^{m+1})$, or more specifically, $C_{m+1}h^{m+1}$, and there is no constrains for this constant C_{m+1}.

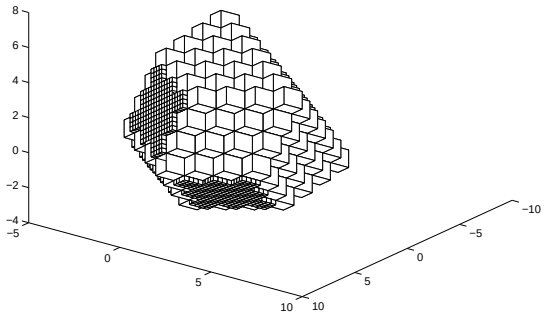


Figure 8: To show how to reduce the constant before the local truncation error, we use these positions as a test problem. This is a commonly used stencil that could be used at the edge.

We compare two special cases that will be frequently used in the composite grid decomposition. The local truncation error for a standard sixth order long scheme central difference scheme in three dimensions has the following form:

$$\begin{aligned} R_6 &= \Delta u(\mathbf{0}) - \frac{1}{h^2}\left(-\frac{1}{90}\sum_{i=1}^{3} u(\pm 3\mathbf{e}_i) + \frac{3}{20}\sum_{i=1}^{3} u(\pm 2\mathbf{e}_i) - \frac{3}{2}\sum_{i=1}^{3} u(\pm \mathbf{e}_i) + \frac{49}{18}u(\mathbf{0}) \right) \\ &= \frac{1}{560}\sum_{i=1}^{3} u^{(8\mathbf{e}_i)}(\mathbf{0})h^6 + \mathcal{O}(h^8). \end{aligned} \tag{3.10}$$

If we pipe the sixth order terms into a vector, the l^2-norm of this vector is $\sqrt{3}/560 \approx 0.0031$. Another point stencil is shown in Fig. 8. If we solve (**P0**) with accuracy requirement set to 5 and do the same pipe-vector procedure, the resulting norm is 19.2773, which is 6,219 times that of the sixth order one. The norm for (**P2**) is 0.8321, which is 268 times larger. Hence, we propose a third strategy by adding a penalty term on the local truncation error in the optimization problem.

($\widetilde{\mathbf{P3}}$) *Penalized stencil generation problem*

$$\min_{\boldsymbol{\omega}} \|\widetilde{\mathsf{T}}\boldsymbol{\omega}\|_2^2 + \|\mathsf{A}_{m+1}\boldsymbol{\omega}\|_2^2, \qquad \text{s.t. } \mathsf{A}\boldsymbol{\omega} = \mathbf{b}, \tag{3.11}$$

where m is the order of accuracy requirement. The last term is exactly the squared pipe-vector norm, since $\mathbf{b}_{m+1} = \mathbf{0}$. Again, we present the Euler-Lagrange version of (3.11).

(**P3**) *Penalized stencil generation problem*

$$\begin{bmatrix} 2\widetilde{\mathsf{T}}^{\mathrm{T}}\widetilde{\mathsf{T}} + 2\mathsf{A}_{m+1}^{\mathrm{T}}\mathsf{A}_{m+1} & \mathsf{A}^{\mathrm{T}} \\ \mathsf{A} & \mathsf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}.$$

After solving (**P3**), we have that the vector norm is 0.0543, which is about 15 times smaller than the one in the case of (**P2**).

3.4 Repository

In the previous subsection, we were able to compute the stencil by an automatic procedure with satisfactory results. However, the procedure of solving saddle point problems for each cell at grid interfaces could produce considerable overhead, for one may have to solve an optimization problem or a linear system of equations for each cell on the interface. In the sixth order setting, the usual size of each stencil problem is in the order of a few hundreds. To overcome this situation, we proposed to establish a pre-defined *stencil repository* before entering the solver. Intuitively speaking, if the stencil is found in the repository, then the coefficients are directly assigned to the grid points without solving any optimization problem.

Since the number of interface points is usually an order of magnitude lower than the number of regular points, it is enough to impose fifth order stencils at the interface. In our implementation, we preset 5701 stencils with fifth order accuracy and 1 stencil that is of sixth order. The idea of the construction of the preset stencil repository is to enumerate some of the two-level situations, *i.e.*, all grid points at the same level, with coarser cells or with finer cells. Also, with the aid of "direction of the stencil" for the latter two situations, better matching stencil performance is acquired. Once the locations are specified, one needs to carry out the calculation of the coefficients as discussed in Section 3.3. We will discuss the effectiveness of solving the different optimization problems.

To close this subsection, we summarize an illustration in Table 2 to show the effectiveness of our current version of the stencil repository. Our test platform is a Mac OS X 10.9 system with Intel(R) Core(TM) i7-2720QM CPU @ 2.20GHz. As we can see in Table 2, the speed of search repository will actually double the performance for the whole process.

Table 2: Effectiveness of stencil repository in Example 4.1.

Strategy	Search	Compute	Total
Compute only (s)	0.00	419.49	419.49
Compute only (pts.)	0	28,476	28,476
Speed (pts./s)	N/A	67.88	67.88
Search + Compute (s)	13.92	187.20	201.12
Search + Compute (pts.)	18,320	10,156	28,476
Speed (pts./s)	1,316.09	54.25	141.22

4 Numerical experiments

In this section, we report some numerical experiments to illustrate the effectiveness and efficiency of the proposed methods.

Example 4.1 (One Gaussian at center of domain)**.** In our first test problem, we consider

the problem

$$\begin{cases} -\Delta u(\mathbf{x}) + p(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = u_b(\mathbf{x}), & \mathbf{x} \in \partial\Omega. \end{cases} \tag{4.1}$$

which is motivated by the linearized Poisson-Boltzmann equation (LPBE) after the change-of-variable technique in [11]. One can check that the analytical functions

$$\begin{aligned} u_a(\mathbf{x}) &= \exp(-\|\mathbf{x}\|_2^2), \\ p(\mathbf{x}) &= \|\mathbf{x}\|_2^2 + 1, \\ f(\mathbf{x}) &= (7 - 3\|\mathbf{x}\|_2^2)\exp(-\|\mathbf{x}\|_2^2), \end{aligned}$$

satisfy (4.1). We solve the problem on $\Omega = [-1,1]^3$ and impose Dirichlet boundary conditions. Naturally we define $u_b(\mathbf{x}) = u_a(\mathbf{x})$ on the boundary. We want to solve this equation to sixth order accuracy.

To check the scaling behavior, we use different mesh lengths to test the discretization strategies. Notice that our machine accuracy is $\epsilon = \mathcal{O}(10^{-16})$ and we hope $\mathcal{O}(h^6)$ could be accurately expressed. Hence, the finest mesh lengths are restricted to the range 0.05 to 0.025, that is the finest mesh of grid points is chosen using 40^3, 50^3, 60^3, and 70^3 grid points for a domain length of 2 in this case. We name the cases Test 1.1 to 1.4. We fix the buffer zone length to 3. We keep track of the following quantities: The error in the discrete L^2 norm to measure the accuracy, the CPU time required to set the interfaces, the CPU time required to solve using a direct method to measure the performance, and the number of non-zeros in the discrete operators to measure the memory requirement. Table 3 shows the resulting discretization details.

Table 3: Summary of discretization for Example 4.1.

Items	Test 1.1	Test 1.2	Test 1.3	Test 1.4
Finest grid size	40^3	50^3	60^3	70^3
h_{Finest}	0.051282	0.040816	0.033898	0.028986
# of DOF	22238	54648	109558	192968
# of DOF (uniform)	54872	110592	195112	314432
# of G_1 interface cells	4662	7992	12222	17352
# of G_2 interface cells	9576	19656	33336	50616
# of G_1 interior cells	0	0	0	0
# of G_2 interior cells	8000	27000	64000	125000
% of interface cells	64.03 %	50.59 %	41.58 %	35.22 %
% of interior cells	35.97 %	49.41 %	58.42 %	64.78 %

We compared the following four strategies for determining the required stencils:

a. Solving (**P2**).

b. Solving (**P3**).

c. Searching the pre-defined stencil repository, coupled with the solution of problem (**P2**).

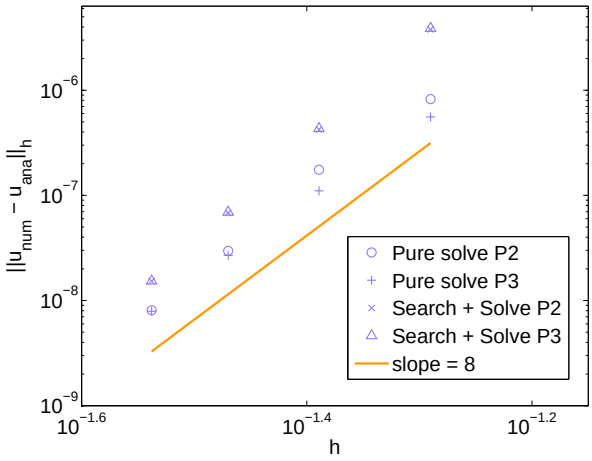
d. Searching the pre-defined stencil repository, coupled with the solution of problem (**P3**).

The resulting data for each one of these strategies are shown in Table 4.

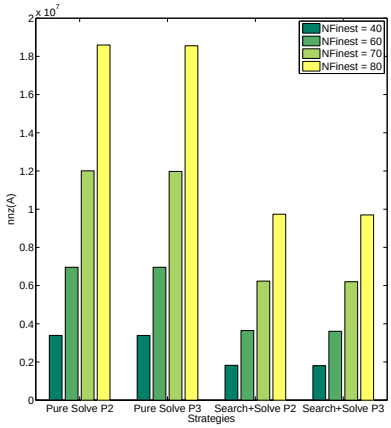
Table 4: Data of Example 4.1

Items	Test 1.1a	Test 1.2a	Test 1.3a	Test 1.4a
# of non-zeros of A	3391524	6959379	12007475	18593900
$\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}$	8.23×10^{-7}	1.75×10^{-7}	2.97×10^{-8}	8.08×10^{-9}
CPU time for G_1 Interface (s)	164.61	294.82	476.00	665.05
CPU time for G_2 Interface (s)	350.75	690.81	1167.69	1773.56
$C_5(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-5})$	2.32	1.55	0.66	0.39
$C_6(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-6})$	45.23	37.90	19.57	13.62
CPU time for solving $Ae=r$ (s)	23.29	172.66	546.87	2388.96
Items	Test 1.1b	Test 1.2b	Test 1.3b	Test 1.4b
# of non-zeros of A	3389526	6958396	11975966	18556896
$\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}$	5.57×10^{-7}	1.10×10^{-7}	2.69×10^{-8}	7.91×10^{-9}
CPU time for G_1 Interface (s)	266.52	472.26	752.59	1064.21
CPU time for G_2 Interface (s)	534.09	1056.34	1806.82	2676.70
$C_5(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-5})$	1.57	0.98	0.60	0.39
$C_6(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-6})$	30.63	23.90	17.73	13.34
CPU time for solving $Ae=r$ (s)	22.22	148.32	552.05	2055.83
Items	Test 1.1c	Test 1.2c	Test 1.3c	Test 1.4c
# of non-zeros of A	1825934	3642397	6234235	9737620
$\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}$	3.79×10^{-6}	4.23×10^{-7}	6.81×10^{-8}	1.53×10^{-8}
CPU time for G_1 Interface (s)	167.25	299.82	477.53	675.26
CPU time for G_2 Interface (s)	71.62	101.63	132.38	165.44
$C_5(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-5})$	10.68	3.74	1.52	0.75
$C_6(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-6})$	208.16	91.58	44.87	25.78
CPU time for solving $Ae=r$ (s)	57.39	306.97	621.25	1710.70
Items	Test 1.1d	Test 1.2d	Test 1.3d	Test 1.4d
# of non-zeros of A	1803570	3607000	6200330	9698220
$\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}$	3.88×10^{-6}	4.32×10^{-7}	6.92×10^{-8}	1.54×10^{-8}
CPU time for G_1 Interface (s)	265.98	480.20	753.80	1093.06
CPU time for G_2 Interface (s)	106.85	153.05	199.59	248.24
$C_5(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-5})$	10.94	3.81	1.55	0.75
$C_6(=\|\mathbf{u}_a-\mathbf{u}_e\|_{\mathbf{h}}\times h_{\text{Finest}}^{-6})$	213.35	93.46	45.63	25.90
CPU time for solving $Ae=r$ (s)	21.69	132.60	555.57	1725.21

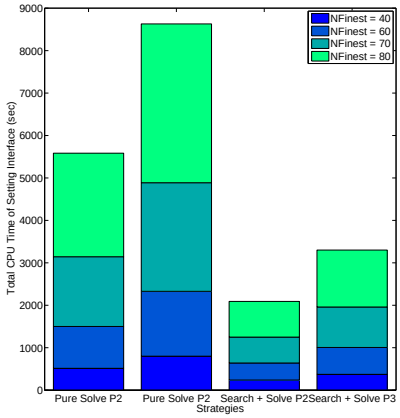
Remark 4.1. In terms of accuracy, solely solving either optimization problem (**P2**) or (**P3**) seems to be the optimal choice. Accuracies are almost the same, although (**P3**) slightly outperforms (**P2**). All these strategies result in a good order of accuracy, with a numerical scaling around 8-th order. (see Fig. 9(a)). However, as expected, these two strategies are more time consuming. An initial repository search step speeds up the stencil creation process (see Fig. 9(c)). It will also reduce the number of nonzero terms in the discrete operator (see Fig. 9(b)), at the cost of a small sacrifice in the accuracy (again, see Fig. 9(a)). This, however, only seems to affect the pre-factor in the error, and not the scaling.



(a) Absolute error in discrete L^2 norm for analytic setting 1. Scaling is approaching 8.



(b) Nonzero number in discrete Laplace operator.



(c) CPU time for Example 4.1 with difference strategies.

Figure 9: Performance plot for Example 4.1.

Example 4.2 (Ten Gaussians)**.** We consider now a more complex example (see Fig. 10):

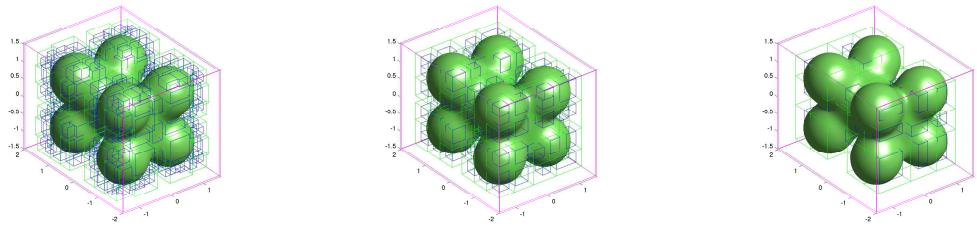
$$\begin{aligned}
u(\mathbf{x}) =& G\left(\frac{1}{2}\mathbf{e}_2,\frac{\mathbf{e}}{3},4\right)+G\left(-\frac{1}{2}\mathbf{e}_2,\frac{\mathbf{e}}{3},4\right)\\
&+G\left(d_2(\mathbf{e}_1+\mathbf{e}_3)+d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right)+G\left(d_2(\mathbf{e}_1-\mathbf{e}_3)+d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right),\\
&+G\left(d_2(\mathbf{e}_3-\mathbf{e}_1)+d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right)+G\left(d_2(-\mathbf{e}_1-\mathbf{e}_3)+d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right),\\
&+G\left(d_2(\mathbf{e}_1+\mathbf{e}_3)-d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right)+G\left(d_2(\mathbf{e}_1-\mathbf{e}_3)-d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right),\\
&+G\left(d_2(\mathbf{e}_3-\mathbf{e}_1)-d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right)+G\left(d_2(-\mathbf{e}_1-\mathbf{e}_3)-d_1\mathbf{e}_2,\frac{\mathbf{e}}{3},2\right),\\
p(\mathbf{x}) =& 81r^2+81,
\end{aligned}$$

where $d_1=\frac{1}{4}+\frac{1}{\sqrt{2}}$, $d_2=\frac{1}{\sqrt{2}}$ and $s:=x_1+x_2+x_3$. We assume the convention that $\mathbf{e}:=(1,1,1)$. Function G is the three dimensional Gaussian-like function defined by

$$\begin{aligned}
G(\mathbf{x},\sigma,A) &:= A\cdot\prod_{i=1}^{3} g(x_i,\sigma_i,1),\\
g(x_0,\sigma,A) &:= A\cdot\exp\left(-\frac{(x-x_0)^2}{2\sigma^2}\right).
\end{aligned}$$

Problem is defined on a cuboid domain $\Omega:=\left[-\frac{121}{80},-\frac{161}{80},-\frac{121}{80}\right]\times\left[\frac{121}{80},\frac{161}{80},\frac{121}{80}\right]$ with finest mesh $122\times162\times122$. The contour plot of the right hand side can be seen in Fig. 10. A full discretization requires the solution of a linear system of order $120\times160\times120=2{,}304{,}000$. We plot the CPU time corresponding to Example 4.1 in Fig. 11 and extrapolate the CPU time for the case where a full finest grid is imposed on the whole domain. We observe that for each case, the predicted CPU time is about the same. The minimal predicted CPU time is about 1.44660×10^5 seconds $=1.67$ days.

Next, we evaluate the performance of the discretization using a composite mesh. We tested the effect of the buffer zone length in three situations by setting this length in the



(a) (1,4) (b) (2,5 (c) (3,6)

Figure 10: Three dimensional contour plot $f(\mathbf{x})=0.3$ and composite grids for Example 4.2.

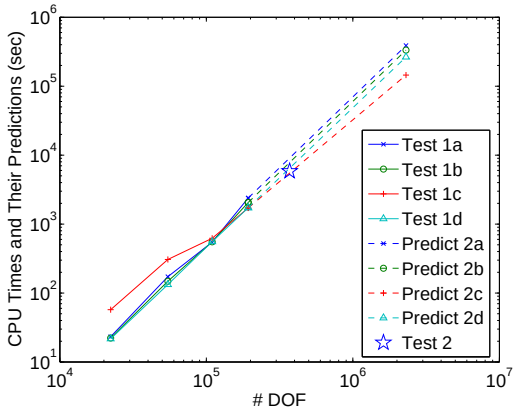


Figure 11: CPU time for Example 4.1 and its prediction for uniform finest problem in Example 4.2.

range $1-3$; we also ranged the minimal patch length from 4 to 6. The resulting composite grid is shown in Fig. 10 and their statistics are shown in Table 5. Our test is based on Discretization 3. The resulting discrete L^2-norm of error is $\|\mathbf{u}-\mathbf{u}_a\|_{\mathbf{h}} = 2.733384\times 10^{-5}$. And the CPU time cost is about 5.02120×10^3 seconds.

Table 5: Statistics for Example 4.2.

n_{bfzone}	$n_{\texttt{minlen}}$	$\ell_{\max}$	n_{grids}	$n_{\text{interface}}$	CPU Time (sec)	Speed (sec/cell)
1	4	3	390	599,936	29,996.8	0.05
2	5	3	222	397,661	19,883.1	0.05
3	6	3	90	289,354	15,081.8	0.052

5 Concluding remarks

We discussed the problems for high order finite difference discretization on composite grids. A modification on the traditional point clustering algorithm with adjustable parameters was introduced. We proved the effectiveness of these parameters in reducing CPU time by numerical experiments.

We also explained the difficulties in selecting the DOFs in a stencil. To compute the coefficients in the stencil, a special Tikhonov regularization technique was applied, transforming the optimization problem into a saddle point problem. Moreover, a stencil repository was designed to speed up the discretization process.

To evaluate the performance of this discretization, we tested the algorithm on a second order elliptic problem. Both order of accuracy and CPU time agreed with our expectations.

Acknowledgments

The first author gratefully acknowledge financial support by China Scholarship Council (No. 2011610055) and would like to express thanks to Prof. Wenbin Chen and Dr. Jingrun Chen for their patient guidance and enlightening discussions. The work of W. Gao is supported by the National Natural Science Foundation of China under grants 11071047 and 91330202. Carlos J. García-Cervera would also like to acknowledge support from NSF CAREER award DMS-0645766.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

[2] N. A. Baker, M. J. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson–Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput. Chem.*, 21 (15): 1343–1352, 2000.

[3] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82 (1): 64–84, 1989.

[4] M. J. Berger and A. Jameson. Automatic adaptive grid refinement for the Euler equations. *AIAA J.*, 23 (4): 561–568, 1985.

[5] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53 (3): 484–512, 1984.

[6] M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE T. Syst. Man. Cy.*, 21 (5): 1278–1286, 1991.

[7] G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *arXiv preprint arXiv:1307.2265*, 2013.

[8] H. D. Ceniceros and A. M. Roma. A nonstiff, adaptive, mesh refinement-based method for the Cahn-Hilliard equation, *J. Comput. Phys.*, 225 (2): 1849–1862, 2007.

[9] H. D. Ceniceros, R. L. Nos, and A. M. Roma. Three-dimensional, fully adaptive simulations of phase field fluid models, *J. Comput. Phys.*, 229 (17): 6135–6155, 2010.

[10] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo software package for amr applications-design document, 2000.

[11] P. Concus and G. H. Golub. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. *SIAM J. Numer. Anal.*, 10 (6): 1103–1120, 1973.

[12] L. Debreu, C. Vouland, and E. Blayo. AGRIF: Adaptive grid refinement in Fortran. *Comput. Geosci.*, 34 (1): 8–13, 2008.

[13] H. Ding and C.-W. Shu. A stencil adaptive algorithm for finite difference solution of incompressible viscous flows. *J. Comput. Phys.*, 214 (1): 397–420, 2006.

[14] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput.*, 51 (184): 699–706, 1988.

[15] R. L. Hayes, M. Fago, M. Ortiz, and E. A. Carter. Prediction of dislocation nucleation during nanoindentation by the orbital-free density functional theory local quasi-continuum method. *Multiscale Model. Sim.*, 4 (2): 359–389, 2006.

[16] M. J. Holst, N. A. Baker, and F. Wang. Adaptive multilevel finite element solution of the Poisson–Boltzmann equation I. Algorithms and examples. *J. Comput. Chem.*, 21 (15): 1319–1342, 2000.
[17] P. Jenny, S. H. Lee, and H. A. Tchelepi. Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Model. Sim.*, 3 (1): 50–64, 2005.
[18] W. D. Lakin. Differentiating matrices for arbitrarily spaced grid points. *Int. J. Numer. Meth. Eng.*, 23 (2): 209–218, 1986.
[19] C.-W. Lan, C.-C. Liu, and C.-M. Hsu. An adaptive finite volume method for incompressible heat flow problems in solidification. *J. Comput. Phys.*, 178 (2): 464–497, 2002.
[20] Y.-F. Li, Z.-P. Liu, L.-L. Liu, and W.-G. Gao. Mechanism and activity of photocatalytic oxygen evolution on titania anatase in aqueous surroundings. *J. Am. Chem. Soc.*, 132 (37): 13008–13015, 2010.
[21] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *Comput. Phys. Commun.*, 126 (3): 330–354, 2000.
[22] D. F. Martin and K. L. Cartwright. *Solving Poisson's equation using adaptive mesh refinement*. Electronics Research Laboratory, College of Engineering, University of California, 1996.
[23] B. W. O'Shea, G. Bryan, J. Bordner, M. L. Norman, T. Abel, R. Harkness, and A. Kritsuk. Introducing Enzo, an AMR cosmology application. *arXiv preprint astro-ph/0403044*, 2004.
[24] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
[25] V. Shenoy, R. Miller, E. Tadmor, D. Rodney, R. Phillips, and M. Ortiz. An adaptive finite element approach to atomic-scale mechanics – the quasicontinuum method. *J. Mech. Phys. Solid*, 47: 611–642, 1999.
[26] Y. Wang and J. Zhang. Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation. *J. Comput. Phys.*, 228 (1): 137–146, 2009.
[27] S. M. Wise, J. S. Lowengrub, and V. Cristini. An adaptive multigrid algorithm for simulating solid tumor growth using mixture models, *Math. Comput. Modelling*, 53: 1–20, 2011.
[28] X. Yang, A. J. James, J. S. Lowengrub, X. Zheng, and V. Cristini. An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids, *J. Comput. Phys.*, 217: 364–394, 2006.
[29] D.-E. Zhang, L.-H. Shen, A.-H. Zhou, and X.-G. Gong. Finite element method for solving Kohn–Sham equations based on self-adaptive tetrahedral mesh. *Phys. Lett. A*, 372 (30): 5071–5076, 2008.
[30] J. Zhang. An explicit fourth-order compact finite difference scheme for three-dimensional convection-diffusion equation. *Commun. Numer. Meth. Eng.*, 14 (3): 209–218, 1998.
[31] X. Zheng, J. Lowengrub, A. Anderson, and V. Cristini. Adaptive unstructured volume remeshing II: Applications to two- and three-dimensional level set simulations of multiphase flow, *J. Comp. Phys.*, 208: 626–650, 2005.
[32] X.-L. Zhu and J. Oliger. *Data Structures and Implementations of Composite Adaptive Grid Methods*. Citeseer, 1995.