High-Order, Multidimensional, and Conservative Coarse-Fine Interpolation for Adaptive Mesh Refinement

Qinghai Zhang

Applied Numerical Algorithms Group, High Performance Computing Research Department, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Abstract

The author presents a polynomial-based algorithm for high-order multidimensional interpolation at the coarse-fine interface in the context of adaptive mesh refinement on structured Cartesian grids. The proposed algorithm reduces coarse-fine interpolation to matrix-vector products by exploiting the static mesh geometry and a family of nonsingularity-preserving stencil transformations. As such, no linear system is solved at the runtime and the ill-conditioning of Vandermonde matrix is avoided. The algorithm is also generic in that `D`, the dimensionality of the computational domain, and p, the degree of the interpolating polynomial, are both arbitrary positive integers. Stability and accuracy are verified by interpolating simple functions, and by applying the proposed method to adaptively solving Poisson's equation and the convection-diffusion equation. The companion MATLAB® package, AMRCFI, is also freely available for convenience and more implementation details.

Key words: Adaptive mesh refinement; Coarse-fine interpolation; AMRCFI; Principal lattice; Constrained least square; The convection-diffusion equation.

Dedicated in loving memory to my mentor, Madam YIN, Ping

1 Introduction

Interface conditions are an important ingredient of structured adaptive mesh refinement (AMR). This paper reports a high-order, multidimensional, and conservative interpolation method that produces the necessary fine ghost cell values at a coarse-fine interface. We first locate a stencil of coarse grid cells,

Email address: `QHZhang@lbl.gov` (Qinghai Zhang).

Published in Comput. Methods Appl. Mech. Engrg., 200(45-46):3159-3168, 2011. *Last updated : 14 September 2011*

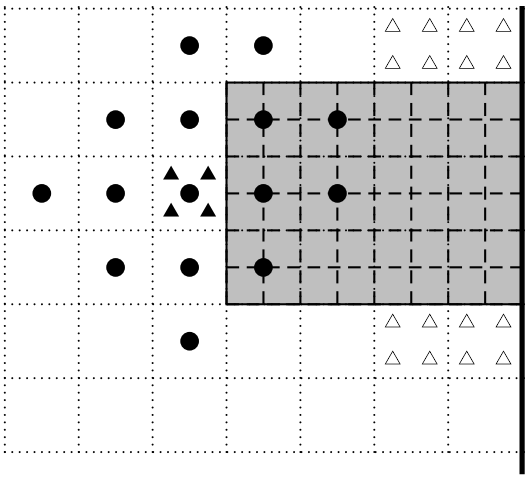


Fig. 1. Coarse-fine interpolation for AMR. The fine level is shaded. '●'s represent a fifth-order interpolation stencil for filling the fine-level ghost cells at '▲'s. The thick and thin solid lines represent a physical boundary and coarse-fine interface, respectively. The interpolation stencil for '△'s is different from that for '▲'s.

then fit a multivariate polynomial to the known coarse cell averages of the stencil, and finally compute the average of this polynomial over smaller ghost cells. This approach is general in terms of dimensionality and the polynomial degree, and is efficient in that it only entails matrix-vector products at the runtime. Its main component, the stencil generation algorithm, is also useful for least square approaches.

As a powerful tool, AMR has been applied to various differential equations and scientific computing problems, cf. [6] and references therein. Most of the previous AMR solvers are second-order accurate and there has been an increasing interest [7,11,12] in further developing them to higher-order accuracies. Indeed, this is the main motivation of this work. As shown in Fig. 1, physical quantities need to be interpolated from the coarse level to the fine level ghost cells for evaluating discrete operators. This poses a number of challenges:

(I) selecting appropriate interpolation stencils from varying source points;
(II) fulfilling high-order accuracy;
(III) achieving the best efficiency.
(IV) preventing ill-conditioning;

Challenge (I) is due to the fact that, along the coarse-fine interface, the interpolation stencil has to adapt itself to the local geometry of available interpolation source. Usually two factors determine which coarse cells are available: the distance from physical domain boundaries and the *proper nesting width*, i.e., the minimum number of layers of coarse cells that surrounds the fine level, see (11) and Fig. 3 for its precise definition and illustration.

Previously, linear least square (LLS) has been used, often with an additional

conservation constraint that the average of the fine ghost values equal that of the covering coarse cell, see [7] for a fourth-order example via fitting multivariate polynomials. The interpolation stencils there are handpicked for the special cases of $p = 3, \mathtt{D} = 2, 3$ and not generic to other values of p and $\mathtt{D}$. In the $(p, \mathtt{D})$-generic case, using more source points than necessary does not guarantee the resulting linear system be nonsingular; blindly adding source points until the related linear system is nonsingular often results in an inefficient interpolation stencil with a large number of cells, especially for high order accuracies and multi-dimensional spaces. One major contribution of this paper is to resolve this difficulty by proposing a $(p, \mathtt{D})$-generic stencil-generation algorithm. Although the new method does not belong to LLS, the generated stencils are of much utility to the LLS approach, since the related linear system is square and nonsingular, and more source points can be added in a *controlled* manner to yield a LLS formulation, see the end of Section 2.3 for more details.

Challenge (III) concerns efficiency. When the positions of the interpolation targets and source points are not known *a priori*, the associated linear system has to be solved 'on-the-fly'; consequently, the efficiency of the interpolation mainly depends on locating the stencil, assembling and solving the linear system, although minor issues such as polynomial evaluation sometimes require additional care. For structured AMR, however, a better efficiency can be achieved by exploiting the following facts on the static mesh geometry:

- the valid interpolation source points are all contained in a rectangular box or unions of boxes, due to the way how tagged cells are organized into new refinement levels [1];
- the normalized distances between the coarse cell center and the fine ghosts centers are uniquely determined by the refinement ratio.

The first fact makes it possible to have a small number [1] of nonsingular stencils for *all* coarse-fine interpolation scenarios of a certain order-of-accuracy. The second fact makes it possible to solve the associated linear system once and for all so that at the runtime coarse-fine interpolation is carried out by matrix-vector products. This also avoids the ill-conditioning of the Vandermonde system and answers challenge (IV).

The rest of the presentation is organized as follows. After a brief introduction to AMR grids, Section 2.1 defines the coarse-fine interpolation problem as two parts: generating stencils and their corresponding linear maps; the former is answered in Section 2.2 while the latter in Section 2.3. Section 3 demonstrates the stability and accuracy of the proposed method by applying it to fourth-order AMR solvers as well as simple interpolation tests. Section 4 concludes this paper with a future research prospect.

[1] this number is $\big(1 + \lfloor p/2 \rfloor\big)^{\mathtt{D}}$, e.g., it is 8 for fourth-order interpolation in 3D.

2 Algorithm

The linear space of all polynomials of degree at most p in $\mathtt{D}$ variables is denoted by $\Pi_p^{\mathtt{D}}$, the dimension of which is $N_p^{\mathtt{D}} = \binom{p+\mathtt{D}}{\mathtt{D}}$ [3, p.29]. The set of monomials

$$\left\{\mathbf{x} \mapsto \mathbf{x}^{\mathbf{q}} : \mathbf{q} \in \mathbb{N}^{\mathtt{D}}, |\mathbf{q}| \leq p\right\} \tag{1}$$

spans $\Pi_p^{\mathtt{D}}$, thus each element of $\Pi_p^{\mathtt{D}}$ can be expressed as

$$f(\mathbf{x}) = \sum_{|\mathbf{q}| \leq p} c_{\mathbf{q}} \mathbf{x}^{\mathbf{q}}, \tag{2}$$

where $|\mathbf{q}| = \sum_{d=1}^{\mathtt{D}} q_d$ and $\mathbf{x}^{\mathbf{q}} = \prod_{d=1}^{\mathtt{D}} x_d^{q_d}$ are the usual multi-index notation. For a multi-index $\boldsymbol{i} \in \mathbb{Z}^{\mathtt{D}}$, $|\boldsymbol{i}| = \sum_{d=1}^{\mathtt{D}} |i_d|$; $\mathrm{abs}(\boldsymbol{i})$ and $\mathrm{sgn}(\boldsymbol{i})$ denote the absolute-value function and the signum function, respectively, applied component-wise to $\boldsymbol{i}$. Similarly, a *partial order* $\boldsymbol{i} < \bar{\boldsymbol{i}}$ holds component-wise, so do '$\leq$', '$>$', and '$\geq$'. A *strict total order* $J : \mathbb{N}^{\mathtt{D}} \to \mathbb{N}$ sorts the monomial set (1) into a *chain* and facilitates iterations over the set of monomials (1); the one employed in implementing this work is the *colexicographical order* [9, p.14]:

$$\boldsymbol{i} <^{\mathrm{colex}} \bar{\boldsymbol{i}} \;\Leftrightarrow\; (\exists m > 0)(\forall d > m)\,(i_m < \bar{i}_m) \wedge (i_d = \bar{i}_d)\,. \tag{3}$$

2.1 Formulation of the Coarse-Fine Interpolation Problem

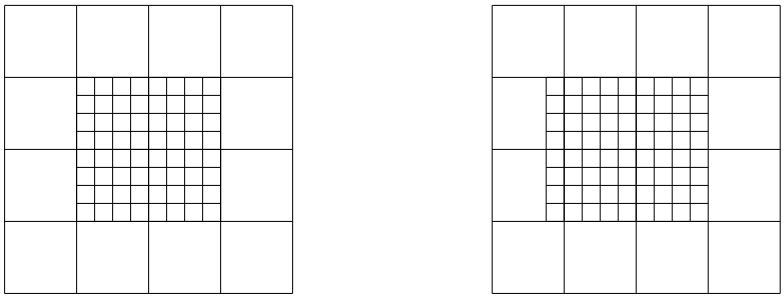
A rectangular mesh discretizes the problem domain Γ into a collection of control volumes $\mathcal{C}(\boldsymbol{i}, h) = \left[\boldsymbol{i} - \frac{\mathbb{1}}{2},\ \boldsymbol{i} + \frac{\mathbb{1}}{2}\right] h$, where $\boldsymbol{i} \in \mathbb{Z}^{\mathtt{D}}$, $\mathbb{1}$ the multi-index whose components are all 1's, the mesh spacing h is assumed to be uniform for ease of exposition. Within a single discretization of the same h, $\mathcal{C}(\boldsymbol{i}, h)$ is represented by $\boldsymbol{i}$ and the cell-average of a scalar function $\phi : \mathbb{R}^{\mathtt{D}} \to \mathbb{R}$ is defined as $\langle\phi\rangle_{\boldsymbol{i}} = \frac{1}{h^{\mathtt{D}}} \int_{\mathcal{C}(\boldsymbol{i},h)} \phi(\mathbf{x})$. A *box* is a rectangular region uniquely determined by two multi-indices:

$$\mathcal{B}(\boldsymbol{j}, \bar{\boldsymbol{j}}) = \left\{\boldsymbol{i} \in \mathbb{Z}^{\mathtt{D}} :\ \min(\boldsymbol{j}, \bar{\boldsymbol{j}}) \leq \boldsymbol{i} \leq \max(\boldsymbol{j}, \bar{\boldsymbol{j}}),\ \right\}, \tag{4}$$

where min and max are also applied component-wise.

In structured AMR, Γ is covered simultaneously by a number of discretizations $\{\Gamma^\ell : \ell = 0, 1, \cdots, l^{\max}\}$, with each Γ^ℓ a multiindex set representing all the control volumes at the ℓth discretization. Inductively, Γ^0 is the coarsest discretization and $\Gamma^{\ell+1} = \mathscr{C}_r^{-1}\Gamma^\ell$, where r is the uniform *refinement ratio* between two successive discretizations, and the coarsening operator $\mathscr{C}_r : \mathbb{Z}^{\mathtt{D}} \to \mathbb{Z}^{\mathtt{D}}$ defined as

$$\mathscr{C}_r(\boldsymbol{i}') = \left(\left\lfloor \frac{i'_1}{r} \right\rfloor, \cdots, \left\lfloor \frac{i'_{\mathtt{D}}}{r} \right\rfloor\right). \tag{5}$$



(a) Proper refinement (b) Improper refinement

Fig. 2. Examples of the proper refinement condition (9).

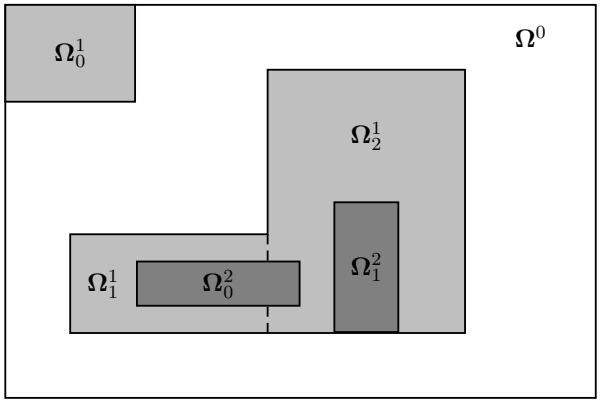


Fig. 3. Examples of the proper nesting condition (11). Ω_0^1: properly nested at physical boundary; Ω_1^1, Ω_2^1: properly nested within coarse level interior; Ω_0^2: properly nested at coarse-fine interface; Ω_1^2: *not* properly nested at coarse-fine interface.

The inverse image of a coarse cell $\boldsymbol{i}$ under $\mathscr{C}_r$, i.e. the set of the r^{D} underlying fine cells, is denoted by

$$\mathcal{U}(\boldsymbol{i}) = \mathscr{C}_r^{-1}(\boldsymbol{i}) = \{\boldsymbol{i}' : \mathscr{C}_r(\boldsymbol{i}') = \boldsymbol{i}\}. \tag{6}$$

The AMR hierarchy Ω consists of a number of *levels*, each of which can be decomposed into a disjoint union of boxes:

$$\Omega = \left\{\Omega^\ell : \ \Omega^0 = \Gamma^0; \ \forall \ell > 0, \Omega^\ell \subset \Gamma^\ell\right\}, \tag{7}$$

$$\Omega^\ell = \left\{\Omega_k^\ell : \ k \neq k' \ \Leftrightarrow \ \Omega_k^\ell \cap \Omega_{k'}^\ell = \emptyset\right\}. \tag{8}$$

As shown in Fig. 2, the *proper refinement condition*

$$\forall \ell > 0, \forall k, \qquad \Omega_k^\ell = \mathscr{C}_r^{-1}\left(\mathscr{C}_r(\Omega_k^\ell)\right); \tag{9}$$

ensures that coarsening or refining does not change the covered region.

Denote the *Minkowski addition* of a point set Ψ and the box $\mathcal{B}(\mathbf{0}, \boldsymbol{i})$ by

$$\mathcal{G}(\Psi, \boldsymbol{i}) = \{\boldsymbol{j} + \boldsymbol{i}_B : \ \boldsymbol{j} \in \Psi; \ \boldsymbol{i}_B \in \mathcal{B}(\mathbf{0}, \boldsymbol{i})\}, \tag{10}$$

the *proper nesting condition* requires

$$\forall \ell > 0, \forall 0 < m \le w_{\text{nest}}, \quad \Omega^{\ell-1} \supseteq \Gamma^{\ell-1} \bigcap \left(\bigcup_{\text{abs}(\boldsymbol{i})=m\mathbb{1}} \mathcal{G}\left(\mathscr{C}_r(\Omega^\ell),\ \boldsymbol{i}\right) \right), \quad (11)$$

where $w_{\text{nest}} \ge 1$ is the proper nesting width, see Fig. 3 for an illustration. The value of w_{nest} depends on the ratio of the maximum width of the finite-difference stencils to the refinement ratio. For fourth-order methods, usually $w_{\text{nest}} = 1$ for $r = 4$; only fine ghosts within the coarse cells abutting the coarse-fine interface need to be filled.

A *stencil* $\mathcal{S}$ is a set of multi-indices, each of which represents a control volume, its cardinality is denoted by $\#\mathcal{S}$. To formalize the coarse-fine interpolation problem, we denote by $\langle \phi \rangle_{\mathcal{S}}$ the average of a scalar function $\phi : \mathbb{R}^{\mathtt{D}} \to \mathbb{R}$ over $\mathcal{S}$ and $\Phi_{\mathcal{S}}$ the value vector of $\langle \phi \rangle_{\mathcal{S}}$. A polynomial f *fits* $\Phi_{\mathcal{S}}$ if $\langle f \rangle_{\mathcal{S}} = \Phi_{\mathcal{S}}$. A stencil $\mathcal{S}$ is *poised* for fitting if, for *any* given $\Phi_{\mathcal{S}}$, there exists a polynomial that fits $\Phi_{\mathcal{S}}$.

Definition 1 (The Coarse-Fine Interpolation (CFI) Problem) *Given a coarse AMR level* Ω^ℓ *satisfying (8), (9), and (11), the* $(p{+}1)$th-order CFI problem *seeks (i) an algorithm of poised stencil generation* $(\boldsymbol{i} \in \Omega^\ell) \mapsto (\mathcal{S}(\boldsymbol{i}) \subseteq \Omega^\ell)$ *with* $\#\mathcal{S} = N_p^{\mathtt{D}} = \dim \Pi_p^{\mathtt{D}}$*, and (ii) an associated linear map* $\mathbf{B} : \Phi_{\mathcal{S}(\boldsymbol{i})} \mapsto \Phi_{\mathcal{U}(\boldsymbol{i})}$*, such that* $\|\Phi_{\mathcal{U}(\boldsymbol{i})} - \mathbf{B}\,\Phi_{\mathcal{S}(\boldsymbol{i})}\| = O(h^{p+1})$ *for any* $\phi \in C^p$ *and any* $\boldsymbol{i} \in \Omega^\ell$*, where* $\mathcal{U}(\boldsymbol{i})$ *is defined in (6) and* h *the grid size of* Ω^ℓ*.*

Note that the condition $\#\mathcal{S} = N_p^{\mathtt{D}} = \dim \Pi_p^{\mathtt{D}}$ is implied by the fitting condition $\langle f \rangle_{\mathcal{S}} = \Phi_{\mathcal{S}}$, from which the *CFI matrix* $\mathbf{B}$ can be deduced as $\langle \Phi \rangle_{\mathcal{U}} = \langle f \rangle_{\mathcal{U}}$. Clearly such a CFI is accurate of order $p + 1$ and satisfies the *conservation constraint*

$$\frac{1}{r^{\mathtt{D}}} \sum_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} \langle \phi \rangle_{\boldsymbol{i}'} = \langle \phi \rangle_{\boldsymbol{i}}\,, \quad (12)$$

because of the finite-volume formulation and the fact that the polynomial determined by $\langle \Phi \rangle_{\mathcal{S}(\boldsymbol{i})}$ and used for evaluating $\langle \Phi \rangle_{\mathcal{U}(\boldsymbol{i})}$ is the same for any given $\boldsymbol{i}$. In a word, Definition 1 implies *automatic conservation*.

In the case of LLS, $\#\mathcal{S} > N_p^{\mathtt{D}}$ and hence $\langle f \rangle_{\mathcal{S}} = \Phi_{\mathcal{S}}$ cannot be satisfied exactly; instead, $\| \langle f \rangle_{\mathcal{S}} - \Phi_{\mathcal{S}} \|$ is minimized with (12) as an additional constraint.

A multi-index $\boldsymbol{j}^*(\boldsymbol{i})$ characterizes the local geometry of available points around a coarse cell $\boldsymbol{i} \in \Omega^\ell$:

$$\forall \boldsymbol{j} \text{ with } \text{abs}(\boldsymbol{j}) = \boldsymbol{j}^*, \qquad \mathcal{B}(\boldsymbol{i}, \boldsymbol{i} + \boldsymbol{j}) \subseteq \Omega^\ell, \quad (13a)$$

$$\forall \boldsymbol{j} \neq \mathbf{0} \text{ and } \text{sgn}(\boldsymbol{j}) = \text{sgn}(\boldsymbol{j}^*), \qquad \mathcal{G}(\mathcal{B}(\boldsymbol{i}, \boldsymbol{i} + \boldsymbol{j}^*), \boldsymbol{j}) \not\subseteq \Omega^\ell. \quad (13b)$$

Intuitively, $\boldsymbol{j}^*(\boldsymbol{i})$ is the offset from $\boldsymbol{i}$ to the closest 'corner' of the coarse box that contains $\boldsymbol{i}$, see Fig. 4 for an illustration.

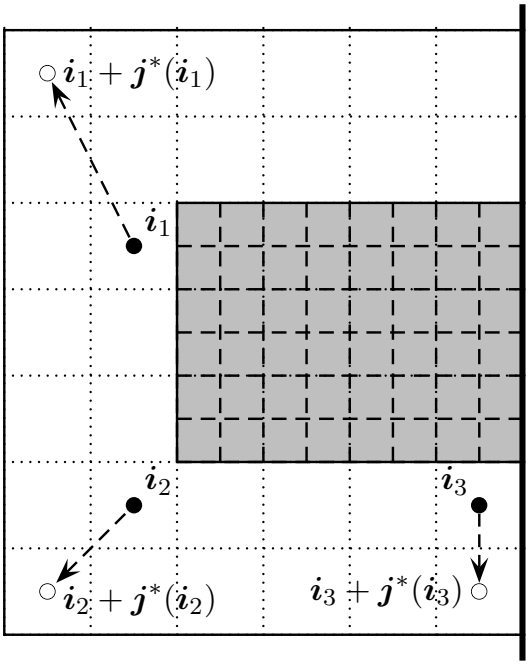


Fig. 4. Characterizing the geometry of available points. The shaded fine level is nested inside a coarse box Ω_k^ℓ with $w_{\text{nest}} = 2$. '●' represents a coarse cell and '○' the closest corner of Ω_k^ℓ; the offset from the former to the latter is $\boldsymbol{j}^*$, shown as a dashed arrow.

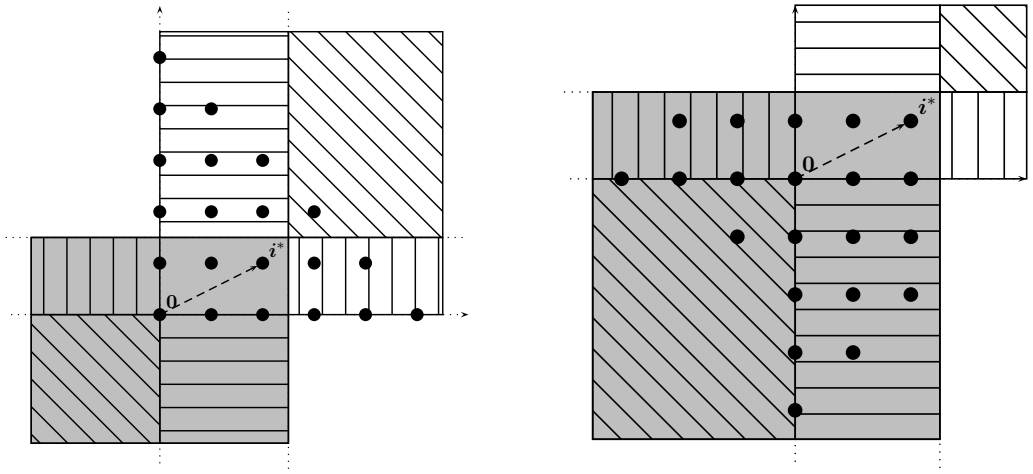
2.2 Poised Stencil Generation: 'Align-Cut-and-Flip'

To generate a stencil, theoretically one can enumerate all the possible choices, compute the condition number for each choice, and select the one with the least condition number. However, rapid growth of the number of possible choices with respect to p and $\mathtt{D}$ makes this approach impractical. For example, to choose a stencil for a polynomial of degree 3 from a 3D 4-by-4-by-4 box, there are more than 10^{17} choices!

The proposed algorithm for generating poised stencils is based on simple transformations of *principal lattice*s. The original concept was defined in $\mathbb{R}^{\mathtt{D}}$ [8] and proved to be poised by a geometric characterization [4]. Restricting this concept to $\mathbb{N}^{\mathtt{D}}$, we define the pth order *principal stencil* as

$$\mathcal{P}_p^{\mathtt{D}} = \left\{ \mathbf{q} \in \mathbb{N}^{\mathtt{D}} : |\mathbf{q}| \leq p \right\}. \tag{14}$$

Note that the word change from 'lattice' to 'stencil' reflects the specialization from $\mathbb{R}^{\mathtt{D}}$ to $\mathbb{N}^{\mathtt{D}}$. Interestingly, this stencil is also the set of multi-index powers of the monomial basis as appeared in (1). Since both the dimension of $\Pi_p^{\mathtt{D}}$ and the cardinality of $\mathcal{P}_p^{\mathtt{D}}$ are $N_p^{\mathtt{D}}$, $\mathcal{P}_p^{\mathtt{D}}$ has the minimum cardinality of all poised stencils for $(p+1)$th-order polynomial fitting.



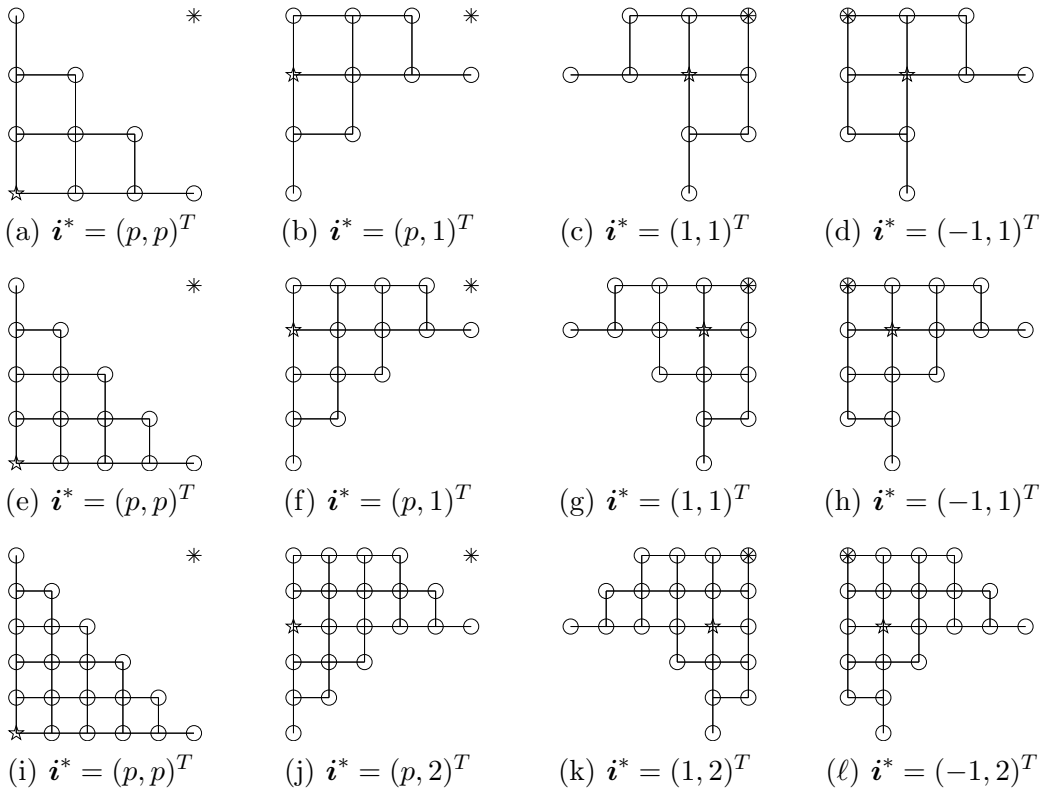
(a) Align and Cut (b) Flip

Fig. 5. principal stencil transformation. The principal stencil is aligned with the coarse cell $\mathbf{0}$, whose fine ghosts need interpolation; $\boldsymbol{i}^*$ then cuts the $\mathtt{D}$-dimensional space into $2^{\mathtt{D}}$ hyperoctants. The multi-indices in the all-negative $\boldsymbol{i}^*$-hyperoctants (shaded area) remains unchanged while those in all other $\boldsymbol{i}^*$-hyperoctants (unshaded areas) are flipped to corresponding $\mathbf{0}$-hyperoctants. Different types of hatches mark the correspondence. This example has $\boldsymbol{i}^* > \mathbf{0}$, so $\boldsymbol{i}^a = \boldsymbol{i}^*$ and lines 11-17 of Algorithm 1 are not invoked. The iteration in lines 3-10 uses the total order (3).

Input: polynomial degree $p \in \mathbb{N}^+$; dimensionality $\mathtt{D} \in \mathbb{N}^+$; $\boldsymbol{i}^* \in \mathbb{Z}^{\mathtt{D}}$
Output: a poised stencil $\mathcal{S}(\boldsymbol{i}^*)$ with $N_p^{\mathtt{D}}$ multi-indices.

```
1  i^a ← abs(i^*)
2  S ← an empty set of multi-indices
3  foreach q ∈ P_p^D do
4  |  for d = 1, 2, ··· , D do
5  |  |  if q_d > i_d^a then
6  |  |  |  q_d ← i_d^a − q_d
7  |  |  end
8  |  end
9  |  add q into S
10 end
11 for d = 1, 2, ··· , D do
12 |  if i_d^* < 0 then
13 |  |  forall i ∈ S do
14 |  |  |  flip the sign of i_d
15 |  |  end
16 |  end
17 end
18 return S
```

Algorithm 1: Principal stencil transformation: **align-cut-and-flip**.



(a) $\boldsymbol{i}^* = (p, p)^T$ (b) $\boldsymbol{i}^* = (p, 1)^T$ (c) $\boldsymbol{i}^* = (1, 1)^T$ (d) $\boldsymbol{i}^* = (-1, 1)^T$

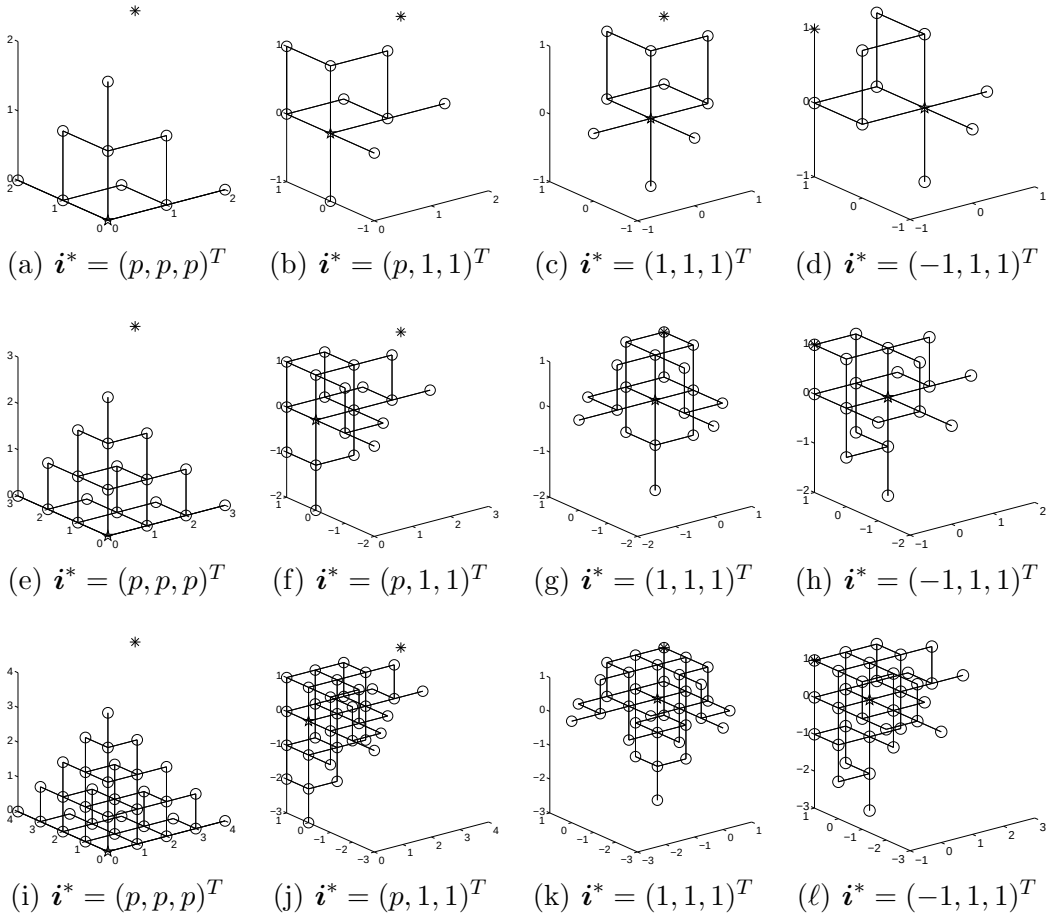
(e) $\boldsymbol{i}^* = (p, p)^T$ (f) $\boldsymbol{i}^* = (p, 1)^T$ (g) $\boldsymbol{i}^* = (1, 1)^T$ (h) $\boldsymbol{i}^* = (-1, 1)^T$

(i) $\boldsymbol{i}^* = (p, p)^T$ (j) $\boldsymbol{i}^* = (p, 2)^T$ (k) $\boldsymbol{i}^* = (1, 2)^T$ (ℓ) $\boldsymbol{i}^* = (-1, 2)^T$

Fig. 6. Examples of 2D stencils generated by Algorithm 1. $p = 3, 4, 5$ for the first, the second, and the third row, respectively. '$*$' represents $\boldsymbol{i}^*$, a star-shaped pentagon the coarse cell $\mathbf{0}$, a circle a nonzero multi-index in the stencil.

The algorithm of poised-stencil generation is a mapping from a multi-index $\boldsymbol{i}^*$to a poised stencil $\mathcal{S}(\boldsymbol{i}^*)$ with the same cardinality as that of the principal stencil $\mathcal{P}_p^{\mathtt{D}}$. It is assumed that $\mathbf{0}$ always represent the coarse cell whose fine ghosts need interpolation; this incurs no loss of generality because the coordinates can be translated so that the origin coincide with the coarse cell being processed. As illustrated in Fig. 5(a), we first *align* the principal stencil $\mathcal{P}_p^{\mathtt{D}}$ with the coarse cell $\mathbf{0}$, then observe that the hyperplanes through $\boldsymbol{i}^a = \mathrm{abs}(\boldsymbol{i}^*)$ and orthonormal to the coordinate axes *cut* the $\mathtt{D}$-dimensional space into a set of $2^{\mathtt{D}}$ $\mathtt{D}$-hyperoctants; likewise those hyperplanes through the origin $\mathbf{0}$ lead to another set of $\mathtt{D}$-hyperoctants. For all $\boldsymbol{i}^a$-hyperoctants except the all-negative one, we *flip* the multi-indices to corresponding $\mathbf{0}$-hyperoctants, as shown in Fig. 5(b) and lines 3-10 of Algorithm 1. Finally, lines 11-17 of Algorithm 1 *flip* the transformed stencil as a whole to select one of its symmetries according to $\mathrm{sgn}(\boldsymbol{i}^*)$. Clearly, the *align-cut-and-flip* procedures amount to a one-to-one mapping between the multi-indices of $\mathcal{S}$ and $\mathcal{P}_p^{\mathtt{D}}$, thus the structure of the principal stencil is preserved in the transformed stencils.

To further illustrate Algorithm 1, examples of transformed stencils are shown in Fig. 6 and Fig. 7 for $\mathtt{D} = 2$ and $\mathtt{D} = 3$ respectively. Any generated stencil is bounded within a box of size $(p+1)^{\mathtt{D}}$ in only one $\boldsymbol{i}^*$-hyperoctant. In particular,



(a) $\boldsymbol{i}^* = (p, p, p)^T$ (b) $\boldsymbol{i}^* = (p, 1, 1)^T$ (c) $\boldsymbol{i}^* = (1, 1, 1)^T$ (d) $\boldsymbol{i}^* = (-1, 1, 1)^T$

(e) $\boldsymbol{i}^* = (p, p, p)^T$ (f) $\boldsymbol{i}^* = (p, 1, 1)^T$ (g) $\boldsymbol{i}^* = (1, 1, 1)^T$ (h) $\boldsymbol{i}^* = (-1, 1, 1)^T$

(i) $\boldsymbol{i}^* = (p, p, p)^T$ (j) $\boldsymbol{i}^* = (p, 1, 1)^T$ (k) $\boldsymbol{i}^* = (1, 1, 1)^T$ (ℓ) $\boldsymbol{i}^* = (-1, 1, 1)^T$

Fig. 7. Examples of 3D stencils generated by Algorithm 1. $p = 2, 3, 4$ for the first, the second, and the third row, respectively. See caption of Fig. 6 for symbols.

if $\boldsymbol{i}^* \geq p\mathbb{1}$, Algorithm 1 produces principal stencils, as shown in the first columns of Fig. 6 and Fig. 7. The stencils in the fourth columns of the two figures are obtained by flipping the signs of the x-coordinates of those in the third columns, this illustrates lines 11-17 of Algorithm 1.

Define the *complete matching set* as

$$\mathbf{M}_p^{\mathtt{D}} = \left\{ \mathcal{S}(\boldsymbol{i}^*) : \mathbf{0} \leq \mathrm{abs}(\boldsymbol{i}^*) \leq \left\lfloor \frac{p}{2} \right\rfloor \mathbb{1} \right\}, \tag{15}$$

where $\mathcal{S}(\boldsymbol{i}^*)$ is a stencil generated by Algorithm 1. To choose a stencil from $\mathbf{M}_p^{\mathtt{D}}$ for a coarse cell $\boldsymbol{i}$, one could first compute $\boldsymbol{j}^*(\boldsymbol{i})$ from (13) and then determine $\boldsymbol{i}^*$ from $\boldsymbol{j}^*$ by

$$i_d^* = \begin{cases} j_d^*, & \text{if } |j_d^*| \leq \left\lfloor \frac{p}{2} \right\rfloor; \\ \mathrm{sgn}(j_d^*) \left\lfloor \frac{p}{2} \right\rfloor, & \text{otherwise.} \end{cases} \tag{16}$$

It follows from (13), (16), and Algorithm 1 that $\mathcal{S}(\boldsymbol{i}^*) \subseteq \Omega^\ell$ holds[2]. The

[2] strictly speaking, this needs another condition that the box containing $\boldsymbol{i}$ has a

choice of $\boldsymbol{i}^*$ in (16) aims to center $\mathcal{S}(\boldsymbol{i}^*)$ as much as possible; meanwhile, no component of $\boldsymbol{i}^*$ has an absolute value bigger than $\left\lfloor \frac{p}{2} \right\rfloor$. Consequently, (15) is complete in that it contains all possible cases of fitting the stencil to the available points.

Currently a rigorous proof of the poisedness of the generated stencils is unavailable, partly because of the scarcity of theorems on concave sets, and partly because of its connection to the open problem of poised lattice classification [4]. Nonetheless, the author gives a strong heuristic argument that multi-indices in the transformed stencil and the principal stencil have a simple one-to-one correspondence by Algorithm 1. In other words, each multi-index in a transformed stencil corresponds to a monomial base function of $\Pi_p^{\mathtt{D}}$. Numerical calculations also verify the poisedness of the transformed stencils for $(p, \mathtt{D}) \in [1, 9] \times [1, 5]$.

2.3 The CFI Matrix

Given a poised stencil, the CFI problem is conceptually simple: a square matrix maps $\Phi_{\mathcal{S}}$ to the monomial coefficients $c_{\mathbf{q}}$'s and another rectangular matrix maps $c_{\mathbf{q}}$'s to $\Phi_{\mathcal{U}}$; hence the product of these two matrices maps $\Phi_{\mathcal{S}}$ to $\Phi_{\mathcal{U}}$.

Because the cardinality of $\mathcal{S}$ equals the dimension of $\Pi_p^{\mathtt{D}}$, integrating (2) over cells in $\mathcal{S}$ yields a square linear system

$$\mathbf{A}\mathbf{a} = \Phi_{\mathcal{S}}, \tag{17}$$

where the $J(\mathbf{q})$th component of the unknown vector is $a_{\mathbf{q}} = c_{\mathbf{q}} h^{|\mathbf{q}|}$, and an element of the basis matrix $\mathbf{A}$ is

$$\mathbf{A}_{\boldsymbol{i},\mathbf{q}} = \prod_{d=1}^{\mathtt{D}} \left(\frac{z_d^{q_d+1}}{q_d+1} \Big|_{i_d-\frac{1}{2}}^{i_d+\frac{1}{2}} \right). \tag{18}$$

Similarly, the integral of (2) within a fine cell $\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})$ leads to

$$\langle \phi \rangle_{\boldsymbol{i}'} = r^{\mathtt{D}}\, \mathbf{b}^{\boldsymbol{i}'} \cdot \mathbf{a} = r^{\mathtt{D}} \sum_{\mathbf{q}} b_{\mathbf{q}}^{\boldsymbol{i}'} a_{\mathbf{q}},$$

where the $J(\mathbf{q})$th component of the row vector is

$$b_{\mathbf{q}}^{\boldsymbol{i}'} = \prod_{d=1}^{\mathtt{D}} \left(\frac{z_d^{q_d+1}}{q_d+1} \Big|_{-\frac{2i'_d-r}{2r}}^{\frac{2i'_d-r+2}{2r}} \right). \tag{19}$$

size at least $(p+1)^{\mathtt{D}}$, which is usually satisfied by the grid generation algorithm.

The automatic conservation property of the proposed method can be confirmed by noticing $\bigcup_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} \mathcal{C}(\boldsymbol{i}', h/r) = \mathcal{C}(\boldsymbol{i}, h)$, $\mathbf{A}_{\boldsymbol{i},\mathbf{q}} = \sum_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} b_{\mathbf{q}}^{\boldsymbol{i}'}$, and

$$\frac{1}{r^{\mathtt{D}}} \sum_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} \langle \phi \rangle_{\boldsymbol{i}'} = \sum_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} \mathbf{b}^{\boldsymbol{i}'} \cdot \mathbf{a} = \sum_{\boldsymbol{i}' \in \mathcal{U}(\boldsymbol{i})} \sum_{\mathbf{q}} b_{\mathbf{q}}^{\boldsymbol{i}'} a_{\mathbf{q}}, = \sum_{\mathbf{q}} \mathbf{A}_{\boldsymbol{i},\mathbf{q}} a_{\mathbf{q}} = \langle \phi \rangle_{\boldsymbol{i}}.$$

Apart from the finite volume formulation and the fact that the fine ghosts unite to the coarse cell, $\#\mathcal{S} = N_p^{\mathtt{D}}$ is a necessary condition for automatic conservation. Indeed, this is why $\#\mathcal{S} = N_p^{\mathtt{D}}$ is required in Definition 1.

The smallest condition number of the basis matrices grows rapidly with p and $\mathtt{D}$, e.g., it is $O(10^5)$ for $p = 6$, $D = 3$. Thus solving (17) for $\mathbf{a}$ and using it for the evaluation of the unknown values incurs *unnecessary* loss of accuracy. To avoid this, the coarse cell averages are directly mapped to fine ghost averages by the CFI matrix

$$\mathbf{B} = \mathbf{A}'\mathbf{A}^{-1} \Rightarrow \Phi_{\mathcal{U}} = \mathbf{B}\Phi_{\mathcal{S}}, \tag{20}$$

where $\mathbf{A}$ is defined by (18) and the rows of $\mathbf{A}'$ are the row vectors $\mathbf{b}^{\boldsymbol{i}'}$'s in (19). Because only integers are involved in (18), (19), and (20), the assembly of $\mathbf{B}$ from these equations can and should be made *exact* by rational number calculations. Avoiding the ill-conditioning of the basic matrix $\mathbf{A}$, the conditioning of the proposed method is measured by the ∞-norm of $\mathbf{B}$. The accuracy advantage of exactly calculating $\mathbf{B}$ is shown in Table 1 by the much larger magnitude of $\mathrm{cond}(\mathbf{A})$ than that of $\|\mathbf{B}\|_\infty$. Note that norms other than $\|\mathbf{B}\|_\infty$ are not appropriate metrics for conditioning or amplification of the CFI linear map because $\mathbf{B}$ is only an *organizing convenience* for the fine ghost cells. Alternatively, we might as well state that CFI of each fine ghost cell is performed by vector inner products.

At the compile time, we enumerate all distinct 4-tuples $(p, D, r, \boldsymbol{i}^*)$ with $\boldsymbol{i}^* \geq 0, \mathcal{S}(\boldsymbol{i}^*) \in \mathbf{M}_p^{\mathtt{D}}$ and precompute the corresponding CFI matrices *once and for all*. For each $(p, \mathtt{D})$, the number of stencils to be stored is thus $(1 + \lfloor p/2 \rfloor)^{\mathtt{D}}$. At the runtime, we select a CFI matrix $\mathbf{B}$ for each coarse cell $\boldsymbol{i}$ and perform CFI by matrix-vector products as in (20); the complexity is clearly $O(1)$. In selecting $\mathbf{B}$, the first method is to directly calculate $\boldsymbol{j}^*(\boldsymbol{i})$ by (13) and deduce $\boldsymbol{i}^*$ from (16). However, when two boxes are adjacent to each other, there might exist a better stencil (in terms of $\|\mathbf{B}\|_\infty$) than that chosen based on a single box. Considering this, the second method organizes all the stencils in a priority queue with $\|\mathbf{B}\|_\infty$ as the priority. For each coarse cell whose underlying fine ghost cells need interpolation, the queue is popped until the current queue top or one of its symmetries fits into Ω^ℓ. The number of pops before finding a matching stencil depends on the coarse-fine geometry and the worst case is the length of the queue $(1 + \lfloor p/2 \rfloor)^{\mathtt{D}}$. The first method is advantageous in its speed while the second method in always obtaining the stencil with the smallest $\|\mathbf{B}\|_\infty$, even for arbitrarily complex coarse-fine geometry; one reasonable settlement is to use the first method for large $(p, \mathtt{D})$'s and the

Table 1
A compare of the ∞-norm of the CFI matrices to the condition numbers of the basis matrices. Here the stencil is generated by $\boldsymbol{i}^* = \lfloor \frac{p}{2} \rfloor \mathbb{1}$.

	$\|\mathbf{B}\|_\infty$, $r=2$			$\|\mathbf{B}\|_\infty$, $r=4$			cond($\mathbf{A}$)		
	$\mathtt{D}=2$	$\mathtt{D}=3$	$\mathtt{D}=4$	$\mathtt{D}=2$	$\mathtt{D}=3$	$\mathtt{D}=4$	$\mathtt{D}=2$	$\mathtt{D}=3$	$\mathtt{D}=4$
$p=2$	1.75	2.50	3.50	2.19	3.62	5.62	8.67e+00	1.52e+01	2.31e+01
$p=3$	2.06	2.69	3.50	2.45	3.44	5.91	3.44e+01	5.35e+01	7.82e+01
$p=4$	2.31	4.27	7.81	3.28	8.25	18.62	2.14e+02	8.17e+02	1.55e+03
$p=5$	2.35	3.75	5.99	2.93	7.01	17.11	2.56e+03	4.45e+03	9.01e+03
$p=6$	2.81	6.80	16.30	4.42	17.76	56.46	2.13e+04	1.61e+05	3.77e+05

second method for smaller ones.

For pure hyperbolic problems with shocks and large gradients in ϕ, $\|\mathbf{B}\|_\infty > 1$ may create oscillations. However, special treatments of the convection term such as limiters or non-oscillatory schemes are necessary *anyway* even if there is no coarse-fine interface. Operating on a set of codimension one, a CFI method should not be responsible for a stability problem encompassing the whole computational domain. On the other hand, any generated stencil is contained in a box of size $(p+1)^{\mathtt{D}}$; thus, as $h \to 0$, $\Phi_{\mathcal{S}}$ tends to a constant vector for a smooth function. Consequently, all components of $\Phi_{\mathcal{U}}$ are the same constant, due to the fact that each row of $\mathbf{B}$ sums up to 1. In other words, the amplification of the CFI matrix approaches 1 as $h \to 0$. This is the reason that *local* polynomial interpolation has been acceptable and, as a matter of fact, successful.

Precomputing the CFL matrix from pre-selected stencils applies equally well to LLS approaches, since a well-posed equality-constrained least square problem can always be converted to an unconstrained one [5, p.585] and solved by QR factorizations. A LLS stencil more symmetric than a transformed principal stencil can be obtained by simply adding more source points to the latter. Eventually this also leads to a linear map from $\Phi_{\mathcal{S}}$ to $\Phi_{\mathcal{U}}$, with its distinguishing feature as $\#\mathcal{S} > N_p^{\mathtt{D}}$. For the case $p = 5, \mathtt{D} = 2$, the perfectly symmetric LLS stencil $\mathcal{S}_L = \{\boldsymbol{i} \in \mathbb{Z}^2 : |\boldsymbol{i}| \leq 3\}$ has $\|\mathbf{B}\|_\infty = 2.11, 2.58$ for $r = 2, 4$, which are even smaller than those (2.35, 2.93) in Table 1; for higher degrees and dimensions, a perfectly symmetric stencil might contain too many source points. Nonetheless, a transformed principal stencil enables a user to have fine control over the number of additional points and to decide her own balance between the computational cost and the conditioning $\|\mathbf{B}\|_\infty$. To sum up, the algorithms proposed in this section also benefits the LLS approach.

Finally, AMRCFI [10], the accompanying MATLAB® package implementing the proposed algorithms, is made freely available so that an interested reader can find more details, generate CFI matrices of specific ranges, or reproduce some of the test results presented in the next section.

3 Tests

In this section, the CFI matrices are first tested using polynomials and sinusoidal functions to examine the conditioning and accuracy, then we employ them in complex fourth-order AMR computations to give the reader more confidence on the stability and convergence of the proposed method. Profiling the vortex ring test shows that CFI consumes about one third of the total CPU time for one Poisson solve while a CFI without precomputing the linear map quadruples the total running time.

3.1 Polynomial interpolation: conditioning

In this test, $\Phi_{\mathcal{S}}$ and $\Phi_{\mathcal{U}}$ are first initialized by averaging polynomial functions of various degrees, with their coefficients randomly generated with zero mean and unit standard deviation. The CFI matrices of the same degrees are then used to compute the error

$$E = \frac{\|\Phi_{\mathcal{U}} - \mathbf{B}\Phi_{\mathcal{S}}\|}{\|\Phi_{\mathcal{U}}\|}. \tag{21}$$

The calculations are performed for $(p, D, r) \in (2, 3, 4, 5, 6) \times (2, 3, 4) \times (2, 4)$, and $\boldsymbol{i}^* = \mathbf{0}, (\lfloor \frac{p}{2} \rfloor, 0, ..., 0), \lfloor \frac{p}{2} \rfloor \mathbb{1}$. The maximum error $E_{\max}$ always occurs at $(p, D, r) = (6, 4, 4)$ with $E_{\max} = 1.2 \times 10^{-14}, 9.6 \times 10^{-15}, 1.9 \times 10^{-15}$ for the above three $\boldsymbol{i}^*$'s, respectively. In all cases, $E_{\max}$ is only a fraction of the upper bound $\|\mathbf{B}\|_\infty \epsilon$ with ϵ being the machine epsilon of double-precision floating-point numbers.

3.2 Sinusoidal interpolation: convergence

In this test, $\Phi_{\mathcal{S}}$ and $\Phi_{\mathcal{U}}$ are initialized by cell-averages of a sinusoidal function

$$f(\mathbf{x}) = \prod_{d=1}^{\mathtt{D}} \cos(x_d - 1). \tag{22}$$

For the case of $\mathtt{D} = 3$, $r = 2$, the interpolation errors (21) and convergence rates on three successively refined grids are shown in Table 2. The convergence rates are uniformly close to $p + 1$, i.e., the accuracy of the interpolation is one order higher than the degree of the interpolating polynomial.

Table 2
The relative errors and convergence rates based on the ∞-norms of CFI errors (21) for the sinusoidal interpolation test. $\mathtt{D} = 3$, $r = 2$.

$\boldsymbol{i}^* = \mathbf{0}$					
	$h = 1/10$	$\mathcal{O}_h$	$h = 1/20$	$\mathcal{O}_h$	$h = 1/40$
$p = 2$	5.37e-04	2.92	7.09e-05	2.96	9.10e-06
$p = 3$	3.04e-05	3.91	2.03e-06	3.96	1.31e-07
$p = 4$	3.16e-06	4.87	1.08e-07	4.94	3.51e-09
$p = 5$	1.76e-07	5.93	2.88e-09	5.97	4.59e-11
$p = 6$	2.03e-08	6.80	1.82e-10	6.89	1.54e-12
$\boldsymbol{i}^* = (\lfloor \frac{p}{2} \rfloor, 0, 0)$					
	$h = 1/10$	$\mathcal{O}_h$	$h = 1/20$	$\mathcal{O}_h$	$h = 1/40$
$p = 2$	4.28e-04	2.93	5.62e-05	2.96	7.21e-06
$p = 3$	1.69e-05	3.82	1.20e-06	3.91	7.93e-08
$p = 4$	2.20e-06	4.86	7.57e-08	4.94	2.47e-09
$p = 5$	1.48e-07	6.06	2.22e-09	5.97	3.53e-11
$p = 6$	1.44e-08	6.79	1.30e-10	6.89	1.10e-12
$\boldsymbol{i}^* = \lfloor \frac{p}{2} \rfloor \mathbb{1}$					
	$h = 1/10$	$\mathcal{O}_h$	$h = 1/20$	$\mathcal{O}_h$	$h = 1/40$
$p = 2$	4.97e-04	2.90	6.65e-05	2.95	8.60e-06
$p = 3$	2.11e-05	3.94	1.38e-06	3.97	8.82e-08
$p = 4$	6.34e-07	4.98	2.00e-08	4.99	6.28e-10
$p = 5$	1.22e-07	5.92	2.01e-09	5.96	3.22e-11
$p = 6$	3.01e-09	6.92	2.49e-11	6.86	2.14e-13

3.3 Vortex Rings: Poisson's Equation

Finite-volume discretization of Poisson's equation $\nabla^2 \phi = \rho$ on structured AMR grids yields

$$L^{\text{comp}} \langle \phi \rangle^{\text{comp}} = \langle \rho \rangle^{\text{comp}}, \tag{23}$$

where $\langle \phi \rangle^{\text{comp}}$ and $\langle \rho \rangle^{\text{comp}}$ are composite arrays with multiple levels, and L^{comp} is a fourth-order discrete composite Laplacian operator with *refluxing* at the coarse-fine interface, i.e., the flux calculated from the coarse level is replaced by the average of those obtained from the fine level. See [11] for more details concerning AMR discretization.

The linear system (23) is solved by a fourth-order multigrid method, which applies V-cycle iterations [2, ch.3] to all AMR levels after converting (23) to the canonical residual-correction form by the fifth-order CFI matrices and periodic domain boundary conditions. As for multigrid iteration on the *residual equation*, it is found that we can simply set the fine ghost cell values to zero

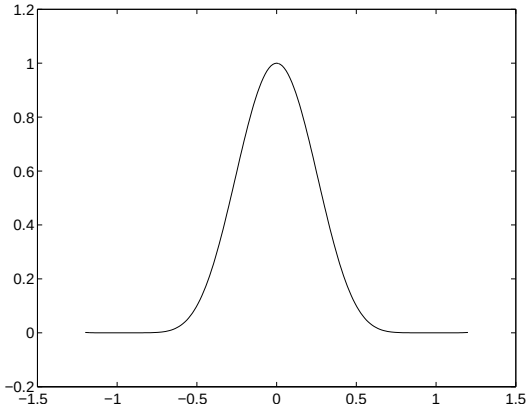


Fig. 8. The polynomial in (24) with unit compact support and unit maximum value;

with no adverse influence on the multigrid convergence.

Two types of errors are calculated. The truncation error results from the difference between the discrete Laplacian $L^{\text{comp}}\langle\phi\rangle$ and $\langle\nabla^2\phi\rangle$ while the solution error from that between $\langle\phi\rangle^{\text{comp}}$ and $\langle\phi\rangle$. The 'exact' cell-averages are approximated by a sixth order Newton-Cotes quadrature formula. A composite sinusoidal function is used for the truncation error while a complex polynomial function imitating a pair of vortex rings for the solution error.

A vortex ring here is a *ring torus*, i.e., a surface of revolution generated by revolving a circle about an axis coplanar and disjoint with the circle. It can also be interpreted as the Cartesian product of this circle to another bigger circle formed by revolving the center of the smaller circle around the axis. Let $\mathbf{n}_v$ denote the direction of the axis, $\mathbf{x}_c$ the center of the big circle, and R_s, R_b the radii of the small and big circles, respectively. Project a vector $\mathbf{x}_v = \mathbf{x} - \mathbf{x}_c$ to $\mathbf{n}_v$ and denote the image as $\mathbf{x}_p = (\mathbf{x}_v \cdot \mathbf{n}_v)\mathbf{n}_v$. The normalized minimum distance from $\mathbf{x}$ to the big circle is thus

$$r_v(\mathbf{x}) = \frac{1}{R_s}\sqrt{\|\mathbf{x}_p\|_2^2 + \left(\|\mathbf{x}_v - \mathbf{x}_p\|_2 - R_b\right)^2}.$$

The vorticity is assigned to be a constant zero outside the vortex ring, a constant maximum vorticity $\phi_{\max}$ for the big circle; within a cross-section of the ring, the vorticity quickly decreases to zero at the perimeter of the small circle:

$$\phi(r_v) = \begin{cases} 0, & r_v > 1 \\ \phi_{\max}\left(r_v^2 - 1\right)^8, & r_v \le 1 \end{cases}. \tag{24}$$

See Fig. 8 for a plot of (24). Note that $\phi(0) = \phi_{\max}$, and $\phi^{(n)}(r_v) = \phi^{(n)}(-r_v) = 0$, $\forall n = 0, 1, \cdots, 7$, i.e., the vorticity is forced to be continuous up to the seventh order. To satisfy the solvability condition, the computational domain

Table 3
Errors and convergence of the vortex ring test. The solution errors are for the vortex ring pair while the truncation errors are evaluated using another exact solution $\phi(\mathbf{x}) = \Sigma_{k=2,4} \prod_{d=0}^{\mathrm{D}-1} \sin(k\pi x_d)$ on the same AMR grids.

Base grid h	1/16	Rate	1/32	Rate	1/64	Rate	1/128
Solution L_∞	3.37e-01	2.42	6.31e-02	3.89	4.26e-03	3.91	2.84e-04
Solution L_1	3.88e-02	4.08	2.29e-03	7.01	1.77e-05	4.26	9.28e-07
Solution L_2	5.22e-02	3.88	3.54e-03	4.75	1.32e-04	3.95	8.53e-06
Truncation L_∞	6.42e-02	2.51	1.13e-02	2.89	1.51e-03	3.00	1.90e-04
Truncation L_1	3.69e-03	3.91	2.45e-04	3.98	1.55e-05	4.00	9.68e-07
Truncation L_2	9.43e-03	3.40	8.96e-04	3.50	7.95e-05	3.51	6.98e-06

contains a pair of vortex rings whose maximum intensities have the same absolute value but opposite signs, so that the RHS of (23) sums to zero.

In generating the AMR hierarchy, we first initialize vorticity on the coarsest level, tag the cells where the absolute value of vorticity are bigger than a certain threshold, then organize the tagged cells into boxes, which are refined to be a finer level [1]. These procedures can be repeated to generate more refinement levels.

The vortex-ring solution on the finest AMR hierarchy is shown in Fig. 9, with the errors and convergence rates listed in Table 3. As expected, the L_∞ convergence rate of the solution error is close to 4 while that of the truncation error around 3. A variety of the transformed stencils are tested by the small boxes on the finest level.

The CPU time is also profiled for the solution tests in serial runs; the program is mostly written in C++ with low-level number-crunching subroutines in Fortran 77. CFI turns out to be a major time consumer of the AMR Poisson solver: for the proposed method with precomputation, 30.6% CPU time of the total solution tests are spent on CFI. Two LLS methods with and without CFI matrix precomputation from pre-selected LLS stencils are also implemented by QR factorizations. For CFI subroutines alone, the ratios of the CPU time to that of the proposed CFI method are 1.21 and 11.7, respectively. Clearly, precomputing CFI matrices is necessary for efficiency: the total CPU time for a AMR Poisson solve would be quadrupled without it.

3.4 Vortex Shear: The Convection-Diffusion Equation

This test solves the 2D convection-diffusion equation

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \mathbf{g} + \nu \nabla^2 \mathbf{u} \tag{25}$$

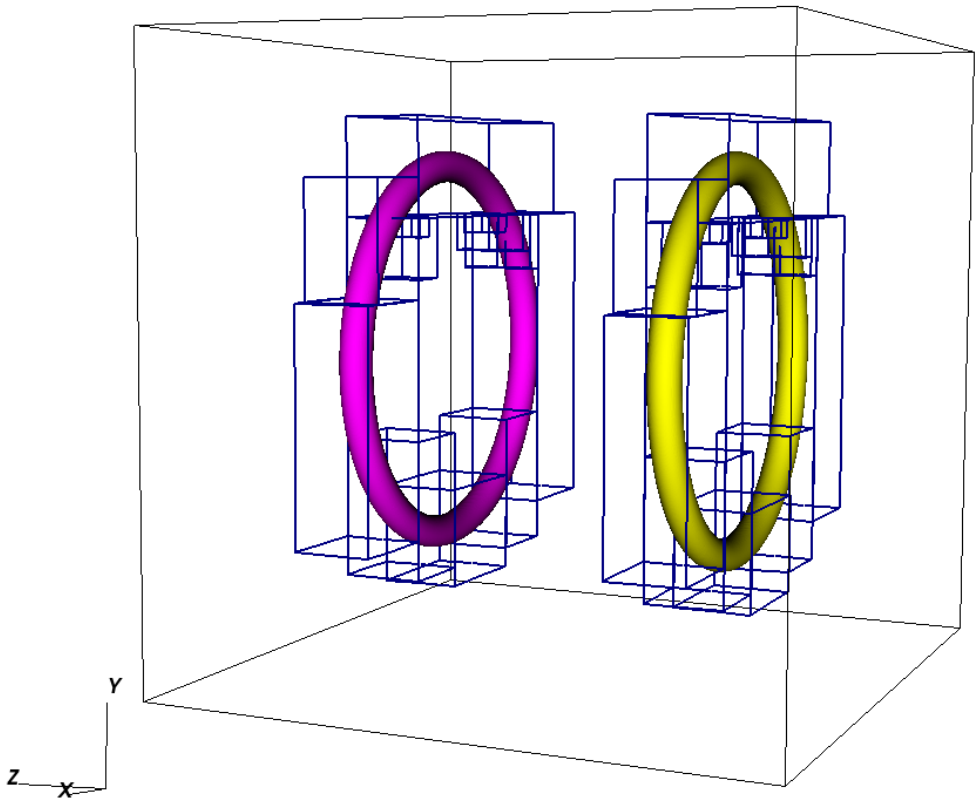


Fig. 9. The vortex-ring AMR hierarchy with base grid as 128^3 and the iso-surfaces of the solution $\phi = 0.1\phi_{\max}$. The proper nesting width is 2 and the refinement ratio $r = 2$. The cells in the coarsest level (the big box with thin solid lines) with $|\langle\phi\rangle| \geq 10^{-4}|\phi_{\max}|$ are tagged and refined to form the first refinement level (not shown) and those with $|\langle\phi\rangle| \geq 10^{-3}|\phi_{\max}|$ form the second refinement level (the boxes with thick lines). The computational domain is $[0, 10]^3$ and other parameters of the vortex ring pairs are: $\phi_{\max} = \pm 10$, $R_b = 3$, $R_s = 0.5$, $\mathbf{n}_v = (0, 0, 1)^T$, $\mathbf{x}_c = (5, 5, 5 \pm 2.5)^T$.

by a six-stage, fourth-order, additive Runge-Kutta (ARK) method, which treats the nonstiff convection and forcing terms explicitly, and the stiff diffusion term implicitly [11]. The exact solution is a divergence-free velocity field with a single vortex,

$$\mathbf{u}(x, y) = \cos\frac{\pi t}{T}\left(\sin^2(\pi x)\sin(2\pi y), -\sin(2\pi x)\sin^2(\pi y)\right), \tag{26}$$

where $T = 8$, and the forcing term $\mathbf{g}$ is derived from (26) and (25). At $t = \frac{T}{2}$, the velocity is reversed so that any scalar field advected by this vortex flow returns to its initial condition at $t = T$, see Fig. 16 in [13] for a rendering of the flow.

An AMR hierarchy in this test contains three static locally-refined grids, see Fig. 10 for the details. On three successively-refined AMR hierarchies, the

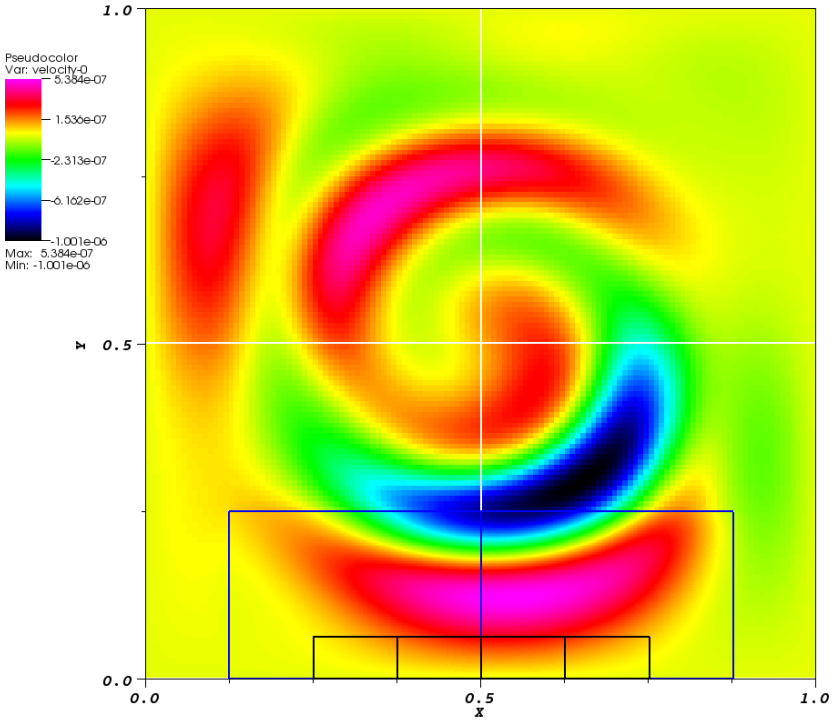
Table 4
Solution errors and convergence of the vortex shear test. $\nu = 0.01$.

Base grid h	1/32	Rate	1/64	Rate	1/128
$t_e = \frac{T}{4}$, the horizontal velocity					
Solution L_∞	2.55e-04	4.00	1.60e-05	4.00	1.00e-06
Solution L_1	3.84e-05	4.04	2.34e-06	4.01	1.45e-07
Solution L_2	6.03e-05	4.03	3.70e-06	4.00	2.30e-07
$t_e = \frac{T}{4}$, the vertical velocity					
Solution L_∞	1.74e-04	4.07	1.04e-05	4.02	6.39e-07
Solution L_1	3.58e-05	4.03	2.20e-06	4.01	1.37e-07
Solution L_2	5.02e-05	4.03	3.08e-06	4.00	1.92e-07
$t_e = T$, the horizontal velocity					
Solution L_∞	1.67e-03	4.03	1.02e-04	4.00	6.39e-06
Solution L_1	3.69e-04	4.02	2.27e-05	4.00	1.42e-06
Solution L_2	5.08e-04	4.02	3.12e-05	4.00	1.95e-06
$t_e = T$, the vertical velocity					
Solution L_∞	1.34e-03	3.99	8.44e-05	3.99	5.31e-06
Solution L_1	3.49e-04	4.02	2.16e-05	3.99	1.35e-06
Solution L_2	4.98e-04	4.02	3.07e-05	4.00	1.93e-06

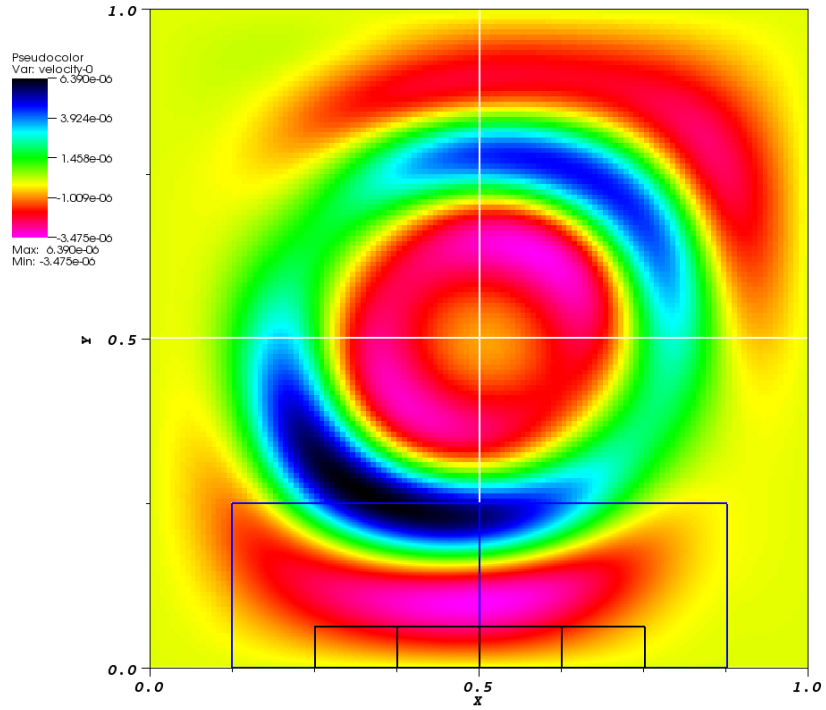
initial condition, set by cell-averaging (26) with a sixth-order Newton-Cotes quadrature formula, is advanced to two ending time instances $t_e = \frac{T}{4}, T$. The domain boundary conditions are homogeneous Dirichlet, realized through filling the domain ghost cells by the formulae in [11]. Since sub-cycling is not used, the time step size is restricted by the smallest grid size; the longer run $t_e = T$ with Courant number $C_r = 1.28$ (the ARK solver is stable provided $C_r < \frac{2.91}{\mathtt{D}}$ [11]) thus entails 12,800 time steps on the finest hierarchy.

In addition to the scenarios mentioned in the previous section, CFI also facilitates explicitly evaluating the Laplacian operator and convection operator, it is thus applied 18 times to all fine levels per time step and 203,400 times for the longer run on the finest hierarchy! This rough estimation signifies the efficiency of CFI; on the other hand, any instability in CFI tends to manifest itself after being applied so many times.

The solution errors and convergence rates are shown in Table 4. For all three types of norms, the convergence rate is uniformly four, although the shapes of the refinement levels create a slight asymmetry between the horizontal and vertical velocity components. The solution errors for the longer run on the finest hierarchy are plotted in Fig. 10; notice how the mosaic patterns vanish across the coarse-fine interface from the coarsest level into the first refinement level. With multigrid convergence ratio set to 10^{-10}, no instability of CFI is observed.



(a) $t_e = \frac{T}{4}$



(b) $t_e = T$

Fig. 10. Error plots of the horizontal velocity for the vortex shear tests. $C_r = 1.28$. The coarsest level $[0,1] \times [0,1]$ is covered by the white boxes, the first refinement level $[\frac{1}{8}, \frac{7}{8}] \times [0, \frac{1}{4}]$ by the blue boxes, and the second refinement level $[\frac{1}{4}, \frac{3}{4}] \times [0, \frac{1}{16}]$ by the black boxes. The base grid size is $h = \frac{1}{128}$ and the refinement ratio $r = 4$.

4 Conclusion

The author proposes a simple, generic, and efficient method of conservative coarse-fine interpolation for high-order AMR solvers. The essential contribution is a heuristic 'align-cut-and-flip' algorithm for generating a family of poised stencils that conforms to the local geometry of available coarse cells. Since the number of points in a poised stencil equals the dimension of the polynomial space, this algorithm is also very useful for least square approaches, as more points can be added without influencing the poisedness of the stencil. By further exploiting the static geometry of structured AMR, CFI is reduced to multiplying the CFI matrix with the known values at a chosen stencil. Numerical tests show satisfactory results in terms of stability and accuracy. Although the derivation assumes cell-averaged values and uniform grid size, it is straightforward to derive similar interpolation matrices for point values and non-uniform grids by modifying AMRCFI [10], the companion MATLAB® package readily available online.

Apart from CFI, physical quantities often needs to be extended smoothly across an irregular interface *within a single level*; the interpolating formulae have to meet additional physical constraints such as the isotropic stress at a free surface [14] and the jump conditions at an irregular solid-fluid interface [15]. These scenarios fall out of the scope of this work since the interpolation source will be poorly characterized by rectangular boxes. The next step along this line of research is to develop an interpolation algorithm applicable to more constraints and more complex geometries.

Acknowledgements

The author would like to acknowledge extremely insightful and constructive comments from one anonymous referee.

References

[1] M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions Systems, Man, and Cybernetics*, 21(5):1278–1286, 1991.

[2] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, second edition, 2000. ISBN:0898714621.

[3] W. Cheney and W. Light. *A Course in Approximation Theory*. American Mathematical Society, 2009. ISBN:978-0821847985.

[4] K. C. Chung and T. H. Yao. On lattices admitting unique Lagrange interpolations. *SIAM J. Numer. Anal.*, 14(4):735–743, 1977.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996. ISBN:978-0801854149.

[6] D. F. Martin, P. Colella, and D. Graves. A cell-centered adaptive projection method for the incompressible Navier-Stokes equations in three dimensions. *J. Comput. Phys.*, 227:1863–1886, 2008. doi:10/1016/j.jcp.2007.09.032.

[7] P. McCorquodale and P. Colella. A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids. *Comm. App. Math. and Comp. Sci.*, 6(1):1–25, 2011. doi:10.2140/camcos.2011.6.1.

[8] R. A. Nicolaides. On a class of finite elements generated by Lagrange interpolation. *SIAM J. Numer. Anal.*, 9(3):435–445, 1972.

[9] S. G. Williamson. *Combinatorics for Computer Science*. Computer Science Press, Inc., 1985. ISBN:0881750204.

[10] Q. Zhang. AMRCFI: A MATLAB® package for coarse-fine interpolation in adaptive mesh refinement, 2011. http://amrcfi.sourceforge.net/.

[11] Q. Zhang, H. Johansen, and P. Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation. *SIAM J. Sci. Comput.*, submitted, 2011. https://seesar.lbl.gov/anag/publication.html.

[12] Q. Zhang, H. Johansen, and P. Colella. A fourth-order accurate projection method for the incompressible Navier-Stokes equations with adaptive mesh refinement. *in progress*, 2011.

[13] Q. Zhang and P. L.-F. Liu. A new interface tracking method: The polygonal area mapping method. *J. Comput. Phys.*, 227(8):4063–4088, 2008. doi:10.1016/j.jcp.2007.12.014.

[14] Q. Zhang and P. L.-F. Liu. HyPAM : A hybrid continuum-particle model for incompressible free surface flows. *J. Comput. Phys.*, 228(4):1312–1342, 2009. doi:10.1016/j.jcp.2008.10.029.

[15] Q. Zhang and P. L.-F. Liu. Handling solid-fluid interfaces for viscous flows: explicit jump approximation vs. ghost cell approaches. *J. Comput. Phys.*, 229(11):4225–4246, 2010. doi:10.1016/j.jcp.2010.02.007.