Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 201 (2004) 34–60
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A node-centered local refinement algorithm for Poisson's equation in complex geometries

Peter McCorquodale [a,*], Phillip Colella [a], David P. Grote [b], Jean-Luc Vay [a]

[a] *Lawrence Berkeley National Laboratory, MS 50A-1148, 1 Cyclotron Rd, Berkeley, CA 94720, USA*
[b] *Lawrence Livermore National Laboratory, Livermore, CA 94550, USA*

Received 19 December 2003; received in revised form 2 April 2004; accepted 13 April 2004
Available online 25 June 2004

Abstract

This paper presents a method for solving Poisson's equation with Dirichlet boundary conditions on an irregular bounded three-dimensional region. The method uses a nodal-point discretization and adaptive mesh refinement (AMR) on Cartesian grids, and the AMR multigrid solver of Almgren. The discrete Laplacian operator at internal boundaries comes from either linear or quadratic (Shortley–Weller) extrapolation, and the two methods are compared. It is shown that either way, solution error is second order in the mesh spacing. Error in the gradient of the solution is first order with linear extrapolation, but second order with Shortley–Weller. Examples are given with comparison with the exact solution. The method is also applied to a heavy-ion fusion accelerator problem, showing the advantage of adaptivity. Published by Elsevier Inc.

PACS: 65N06; 65N55; 78M20

Keywords: Finite difference methods; Poisson equation; Adaptive mesh refinement; Cartesian grid methods; Multigrid methods; Shortley–Weller

1. Introduction

We present a numerical method for solving Poisson's equation with Dirichlet boundary conditions

$$\Delta\varphi = \rho \text{ on } \Omega, \qquad \varphi = g \text{ on } \partial\Omega \tag{1}$$

on a bounded three-dimensional region Ω. Our motivation is to combine adaptive mesh refinement with complex geometry and eventually to include particle-in-cell (PIC) simulation following the ideas in [6]. We use a nodal-point scheme instead of a cell-centered or finite-volume discretization (as in [7] or [1]) because the nodal discretization is a relatively simple way to treat geometry: a node is either inside or outside the domain.

* Corresponding author. Tel.: +1-510-495-2458; fax: +1-510-495-2505.
E-mail address: pwmccorquodale@lbl.gov (P. McCorquodale).
URL: http://seesar.lbl.gov/anag/.

0021-9991/$ - see front matter. Published by Elsevier Inc.
doi:10.1016/j.jcp.2004.04.022

Our approach uses data on nodes of rectangular Cartesian grids. At nodes away from the domain boundary, we take the standard seven-point discretization for (1), with truncation error that is second order in the mesh spacing. At nodes adjoining the boundary, we use the same regular discretization after extrapolating either linearly or quadratically from nodes in the domain. We combine this discretization with an adaptive mesh refinement (AMR) procedure based on the block-structured approach of Almgren [2]. The term "adaptive" is customarily used in this field to refer to solvers on locally refined grids, even when the grids are fixed in advance. Previous work on node-centered Poisson solvers on irregular domains, such as [5,11], has been done on uniform grids only.

Other approaches to solving Poisson's equation in complex geometries include unstructured grid frameworks, but these do not work as easily with PIC methods, or with multigrid solvers. Cartesian grid methods have the advantage over unstructured grids of easier grid generation, particularly in three dimensions, for which unstructured grid generation is an open problem. An approach closely related to ours was developed by Young et al. [14] using a variational formulation based on rectangular finite elements. We also note that their method uses conjugate gradient with an incomplete LU preconditioner as a solver, and requires a great deal more memory than the matrix-free multigrid solver used here. We prefer to use a simpler method that is as close to a classical finite-difference discretization method as possible and for which there is considerable experience in combining them with PIC methods. There is an extensive literature on Cartesian grid methods applied to other problems. For a survey, see [4]. These are mostly volume-of-fluid methods for hyperbolic problems. The state of the art for both algorithms and software for elliptic problems is far less well developed.

The truncation error of our method at the domain boundary varies with the mesh spacing in zeroth order if the extrapolation is linear, or in first order if quadratic. However, even with zeroth-order truncation error on the boundary, the overall induced solution error is still second-order accurate, as we show using a modified-equation analysis as in Johansen and Colella [7]. We solve the linear system using a simple domain-decomposition point-relaxation strategy. We show evidence that the algorithm is second-order accurate in L_∞ norm for various exact solutions, and compare the adaptive and nonadaptive calculations. We show that the gradient computed from the solution is also second-order accurate in L_∞ norm, provided the extrapolation at boundaries is quadratic; if extrapolation is linear, then the gradient is only first-order accurate in L_∞ norm. Thus if gradients are needed, as they are with particle simulations, then second-order accuracy requires quadratic extrapolation at boundary nodes, so that linear extrapolation as suggested in [5] is no longer sufficient. Finally, we demonstrate that our method works efficiently with good multigrid performance on a real physical problem, solving for the potential inside a heavy-ion fusion accelerator.

2. Nodal discretization

2.1. AMR spatial discretization

Adaptive mesh refinement methods are based on sets of grids on multiple levels of refinement, such that grids on the same level have the same mesh spacing. For each level in the AMR hierarchy, the underlying discretization of space is given as lattice points $(i_0, \ldots, i_{\mathbf{D}-1}) = \boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}}$, where $\boldsymbol{i}$ corresponds to a point of space $\boldsymbol{x}_0 + h \cdot \boldsymbol{i}$, with $\boldsymbol{x}_0 \in \mathbb{R}^{\mathbf{D}}$ a fixed origin of coordinates for all levels, and h the mesh spacing at the particular level.

The problem domain is discretized using node-centered grids. A grid $\Gamma \subset \mathbb{Z}^{\mathbf{D}}$ consists of the lattice points within a rectangular region bounded by two points $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{Z}^{\mathbf{D}}$, where $\boldsymbol{p} = (p_0, \ldots, p_{\mathbf{D}-1})$ is the lower corner of Γ and $\boldsymbol{q} = (q_0, \ldots, q_{\mathbf{D}-1})$ is the upper corner. Thus $\Gamma = \{\boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}} : \boldsymbol{p} \leqslant \boldsymbol{i} \leqslant \boldsymbol{q}\}$, where the notation $\boldsymbol{a} \leqslant \boldsymbol{b}$ means each entry of $\boldsymbol{a}$ is less than or equal to the corresponding entry of $\boldsymbol{b}$.

We will find it useful to define a number of operators on points and subsets of $\mathbb{Z}^{\mathbf{D}}$. We denote by $\Gamma + \boldsymbol{i}$ the translation of a set Γ by a point $\boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}}$. We define a coarsening operator $\mathscr{C}_r : \mathbb{Z}^{\mathbf{D}} \to \mathbb{Z}^{\mathbf{D}}$, where r is a positive integer, by

$$\mathscr{C}_r(\boldsymbol{i}) = \left(\left\lfloor \frac{i_0}{r} \right\rfloor, \ldots, \left\lfloor \frac{i_{\mathbf{D}-1}}{r} \right\rfloor \right).$$

The coarsening operator acting on subsets of $\mathbb{Z}^{\mathbf{D}}$ can be extended in a natural way to a node-centered grid: if $\Gamma = \{\boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}} : \boldsymbol{p} \leqslant \boldsymbol{i} \leqslant \boldsymbol{q}\}$ then $\mathscr{C}_r(\Gamma) = \{\boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}} : \mathscr{C}_r(\boldsymbol{p}) \leqslant \boldsymbol{i} \leqslant \mathscr{C}_r(\boldsymbol{q})\}$.

We extend this discretization of space to represent a nested hierarchy of grids that discretize the same continuous spatial domain. We assume that our problem domain can be discretized by a nested hierarchy of grids $\Gamma_N^0, \ldots, \Gamma_N^{l_{\max}}$, with $\Gamma_N^{l+1} = \mathscr{C}_{n_{\text{ref}}^l}(\Gamma_N^l)$, where n_{ref}^l is the refinement ratio between levels l and $l+1$.

AMR calculations are performed on a hierarchy of node-centered meshes $\Omega_N^l \subset \Gamma_N^l$, with $\Omega_N^l \supset \mathscr{C}_{n_{\text{ref}}^l}(\Omega_N^{l+1})$. Typically, Ω_N^l is decomposed into a union of node-centered grids, $\mathscr{R}(\Omega_N^l)$, such that any two distinct $\Gamma, \Gamma' \in \mathscr{R}(\Omega_N^l)$ intersect only on grid faces.

For a node-centered union of rectangles Ω_N at a fixed level of refinement, we define the *interior* nodes $\Omega_{N,\text{int}}$ as those for which all neighbors along coordinate axes are also in Ω_N. That is

$$\Omega_{N,\text{int}} = \Omega_N \cap \bigcap_{d=0,\ldots,\mathbf{D}-1} \left(\left(\Omega_N + \boldsymbol{e}^d\right) \cap \left(\Omega_N - \boldsymbol{e}^d\right) \right),$$

where $\boldsymbol{e}^d$ is the unit vector in dimension d.

A discretized dependent variable in AMR is a *level array*

$$\varphi^l : \Omega_N^l \to \mathbb{R}^m.$$

We denote by $\varphi_{\boldsymbol{i}} \in \mathbb{R}^m$ the value of φ at node $\boldsymbol{i} \in \Omega_N^l$, corresponding to the point $\boldsymbol{x}_0 + h \cdot \boldsymbol{i} \in \mathbb{R}^{\mathbf{D}}$ where h is the mesh spacing at level l.

From a formal numerical analysis standpoint, a solution on an adaptive mesh hierarchy $\{\Omega_N^l\}_{l=0}^{l_{\max}}$ approximates the exact solution to the PDE only on interior nodes, and only on those interior nodes that are not covered by interior nodes at a finer level. These are defined as the *valid* nodes of Ω_N^l (see Fig. 1)

$$\Omega_{N,\text{valid}}^l = \Omega_{N,\text{int}}^l - \mathscr{C}_{n_{\text{ref}}^l}(\Omega_{N,\text{int}}^{l+1}).$$

A composite array φ^{comp} is a collection of discrete values defined on the valid nodes at each of the levels of refinement

$$\varphi^{\text{comp}} = \{\varphi^{l,\text{valid}}\}_{l=0}^{l_{\max}}, \qquad \varphi^{l,\text{valid}} : \Omega_{N,\text{valid}}^l \to \mathbb{R}^m.$$

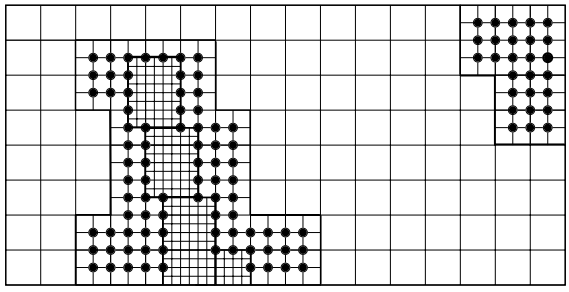

Fig. 1. A two-dimensional grid hierarchy with three levels of refinement. The valid nodes at the middle level are indicated by ●.

2.2. Single-level Laplacian operator

The discrete Laplacian operator Δ^h on a level with mesh spacing h is defined on the valid nodes of that level, as

$$(\Delta^h \varphi^{N,l})_{\boldsymbol{i}} = \sum_{d=0}^{\mathbf{D}-1} \left(\Delta_d^h \varphi^{N,l}\right)_{\boldsymbol{i}}, \tag{2}$$

where Δ_d^h is a second-derivative operator in dimension d

$$\left(\Delta_d^h \varphi^{N,l}\right)_{\boldsymbol{i}} = \frac{\varphi^{N,l}_{\boldsymbol{i}-\boldsymbol{e}^d} - 2\varphi^{N,l}_{\boldsymbol{i}} + \varphi^{N,l}_{\boldsymbol{i}+\boldsymbol{e}^d}}{h^2}. \tag{3}$$

The truncation error of this method can be analyzed. Let φ^e be the exact solution, and for all valid $\boldsymbol{i}$, $\varphi^{e,h}_{\boldsymbol{i}} = \varphi^e(\boldsymbol{i}h + \boldsymbol{x}_0)$. Then the truncation error is defined as

$$\tau_h = \Delta\varphi^e - \Delta^h \varphi^{e,h}.$$

If all the points in the stencil (2) and (3) for $\boldsymbol{i}$ are valid nodes, then it is well known that $(\tau_h)_{\boldsymbol{i}} = \mathrm{O}(h^2)$.

2.3. Multilevel Laplacian operator

There are three cases in which $\boldsymbol{i}$ is a valid node at level l, but $\boldsymbol{j} = \boldsymbol{i} \pm \boldsymbol{e}^d$ is not (see Fig. 2).

(a) The node $\boldsymbol{j}$ is on the physical boundary. Then we take the value of $\varphi^{N,l}_{\boldsymbol{j}}$ from the physical boundary conditions.

(b) The node $\boldsymbol{j}$ is covered by a grid at finer level $l+1$. Then we project the valid data from the finer level, $\varphi^{N,l}_{\boldsymbol{j}} = \varphi^{N,l+1}_{n^l_{\mathrm{ref}}\boldsymbol{j}}$.

(c) The node $\boldsymbol{j}$ is on the boundary with coarser level $l-1$. Then we interpolate from the coarser-level nodes on the coarse/fine interface, as described in Section 2.4.

2.4. Coarse/fine boundary interpolation

When the stencil for evaluating the operator on $\varphi^{N,l}$ at level l includes a node $\boldsymbol{j}$ lying on the boundary with the next coarser level $l-1$, as shown in Fig. 2(c), we must interpolate $\varphi^{N,l}_{\boldsymbol{j}}$ from $\varphi^{N,l-1}$ at the valid coarser-level nodes.

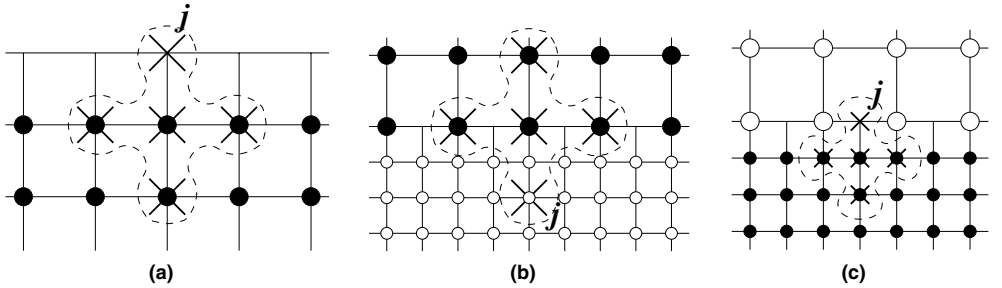

Fig. 2. Cases in which data at non-valid nodes is needed in computing the discrete Laplacian at a valid node. Nodes of the stencil are indicated by ×, valid nodes at the current level by ●. (a) Stencil reaches physical boundary (at top point). (b) Stencil is partially covered by the next finer level (at bottom point). (c) Stencil intersects boundary with coarser level (at top point).

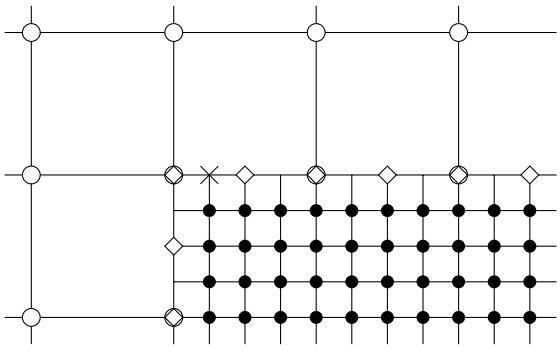

Fig. 3. A two-dimensional example with refinement ratio of 4. We interpolate to finer-level nodes on the coarse/fine interface in two stages. First stage: interpolate from the valid coarser-level nodes indicated by ○ to interface nodes at the intermediate level, indicated by ◇. Intermediate-level interface nodes that are also coarser-level nodes are indicated by ○ superimposed with ◇. Second stage: interpolate from the intermediate-level interface nodes to the finer-level interface nodes. In interpolating to the node marked ×, we use only data at nodes indicated by ◇ in this figure, and not data at other coarser-level or intermediate-level nodes.

We assume that the refinement ratio n_{ref}^{l-1} is a power of 2. Then the interpolation from level $l-1$ to level l is a composition of $\log_2(n_{\text{ref}}^{l-1})$ successive interpolations each with refinement ratio of 2.

If $n_{\text{ref}}^{l-1} > 2$ then we first interpolate from level $l-1$ to the interface with the grids of level l coarsened by $n_{\text{ref}}^{l-1}/2$. The remaining inter-level interpolations use data only on the coarse/fine interface and the physical boundary conditions, which are set at each intermediate stage (see Fig. 3).

In the remainder of this section, we assume a refinement ratio of 2.

If the level-l node $\boldsymbol{j}$ coincides with a node at level $l-1$, then we project the value of $\varphi_{\boldsymbol{j}/2}^{N,l-1}$. In other cases, in two dimensions we use quadratic interpolation along one-dimensional interfaces, and in three dimensions we use biquadratic interpolation along two-dimensional interfaces.

2.4.1. Two-dimensional problem: one-dimensional interface

In the two-dimensional problem, the coarse/fine interface is one-dimensional. Along the interface, the coordinates vary in only one of the two dimensions, say the first. Let (a, b) be the coordinates of a point of approximation on the interface. If h is the mesh spacing at the finer level, then the coarse nodes on the interface are at $(a \pm h, b)$, $(a \pm 3h, b)$, etc. We interpolate to the fine nodes with a piecewise quadratic function.

The piecewise linear interpolation function f_1 on the interface has values equal to $\varphi^{N,l-1}$ at the coarse nodes. At the fine nodes

$$f_1(a) = \frac{f_1(a-h) + f_1(a+h)}{2}. \tag{4}$$

This is extended to the piecewise quadratic interpolation function, f_2, which also has values equal to $\varphi^{N,l-1}$ at the coarse nodes. At the fine nodes

$$f_2(a) = f_1(a) - \frac{h^2}{2} f_2''(a). \tag{5}$$

The second derivative $f_2''(a)$ then comes from the coarse values $f_2(a-h)$, $f_2(a+h)$, and either $f_2(a-3h)$ or $f_2(a+3h)$ or both, using one of the following formulas:

$$f_2''(a) = \frac{f_2(a-3h) - f_2(a-h) - f_2(a+h) + f_2(a+3h)}{2(2h)^2}, \tag{6a}$$

$$f_2''(a) = \frac{f_2(a-3h) - 2f_2(a-h) + f_2(a+h)}{(2h)^2}, \tag{6b}$$

$$f_2''(a) = \frac{f_2(a-h) - 2f_2(a+h) + f_2(a+3h)}{(2h)^2}. \tag{6c}$$

We use (6a) if both $(a-3h, b)$ and $(a+3h, b)$ are valid coarser-level nodes. In case the nodes $(a \pm 3h, b)$ are not both valid on the coarser level, at least one of them must be valid because every grid has length of at least two cells in each dimension. So we use either (6b) or (6c).

2.4.2. Three-dimensional problem: two-dimensional interface

In the three-dimensional problem, the coarse/fine interface is two-dimensional, so that along the interface, the coordinates vary in only two of the three dimensions, say the first two. Let (a, b, c) be the coordinates, in level l, of a point of approximation on the interface with level $l-1$. If both a and b are even, then (a, b, c) is a coarse node, and we project the value $\varphi^{N,l-1}(a/2, b/2, c/2)$. If either a or b is even, but not both (see Fig. 4, left side) then we interpolate along the line as in Section 2.4.1. In the remainder of this section we consider the case in which both a and b are odd (see Fig. 4, right side). Then the coarse nodes are at $(a \pm h, b \pm h, c)$, $(a \pm 3h, b \pm h, c)$, $(a \pm h, b \pm 3h, c)$, etc. We interpolate to the fine nodes with a piecewise biquadratic function.

The piecewise bilinear interpolation function f_1 on the interface has values equal to $\varphi^{N,l-1}$ at the coarse nodes. At the fine nodes

$$f_1(a, b) = \frac{1}{4} \sum_{s=-1,+1} \sum_{t=-1,+1} f_1(a + sh, b + th). \tag{7}$$

This is extended to the piecewise biquadratic interpolation function f_2, which also has values equal to $\varphi^{N,l-1}$ at the coarse nodes. At the fine nodes,

$$f_2(a, b) = f_1(a, b) - \frac{h^2}{2}\left(\frac{\partial^2 f_2}{\partial x^2}(a, b) + \frac{\partial^2 f_2}{\partial y^2}(a, b)\right), \tag{8}$$

where f_1 is the linear interpolant (7). Our value for the second derivative $\partial^2 f_2/\partial x^2(a, b)$ comes from the mean of estimates of $\partial^2 f_2/\partial x^2(a, b-h)$ and $\partial^2 f_2/\partial x^2(a, b+h)$. Likewise for $\partial^2 f_2/\partial y^2(a, b)$ from

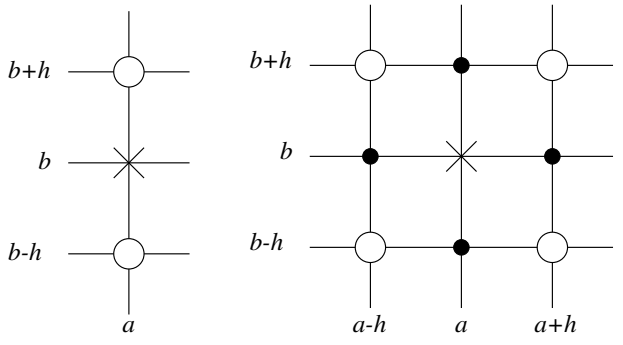

Fig. 4. Interpolating a value at the point (a, b), marked $\times$, from coarse nodes indicated by $\bigcirc$. Left: When a is even, interpolate along the line where the first coordinate is fixed at a. Right: When both a and b are odd, interpolate from the four neighboring coarse nodes and their coarse neighbors. In the piecewise biquadratic interpolant, second partial derivatives are estimated at points indicated by $\bullet$.

$\partial^2 f_2/\partial y^2(a-h,b)$ and $\partial^2 f_2/\partial y^2(a+h,b)$. We calculate these second derivatives from points along a line, following Eq. (6) as described in Section 2.4.1.

2.5. Computing gradients

After solving for the electrostatic potential, we calculate its gradient in order to find the electric field. On a single level with mesh spacing h, the gradient is computed at a valid node $\boldsymbol{i}$ as:

$$(\nabla^h \varphi^{N,l})_{\boldsymbol{i}} = \sum_{d=0}^{\mathbf{D}-1} \left(D_d^h \varphi^{N,l}\right)_{\boldsymbol{i}} \cdot \boldsymbol{e}^d, \tag{9}$$

where D_d^h is a first-derivative operator in dimension d

$$\left(D_d^h \varphi^{N,l}\right)_{\boldsymbol{i}} = \frac{\varphi^{N,l}_{\boldsymbol{i}+\boldsymbol{e}^d} - \varphi^{N,l}_{\boldsymbol{i}-\boldsymbol{e}^d}}{2h}. \tag{10}$$

This stencil for the gradient at $\boldsymbol{i}$ uses the same points as for the discrete Laplacian operator, except for $\boldsymbol{i}$ itself. See Section 2.3 for the discussion of cases in which $\boldsymbol{i}$ is a valid node at level l, but $\boldsymbol{j} = \boldsymbol{i} \pm \boldsymbol{e}^d$ is not. These cases are treated in exactly the same way for the gradient as for the discrete Laplacian. In particular, coarse/fine boundary interpolation is as described in Section 2.4.

For an exact solution φ^e with $\varphi_{\boldsymbol{i}}^{e,h} = \varphi^e(\boldsymbol{i}h + \boldsymbol{x}_0)$ for valid $\boldsymbol{i}$, consider the error in the gradient

$$\eta_h = \nabla^h \varphi^{e,h} - \nabla \varphi^e.$$

If all the points in the stencil (9) and (10) for $\boldsymbol{i}$ are valid nodes, then $(\eta_h)_{\boldsymbol{i}} = \mathrm{O}(h^2)$.

2.6. Truncation error of multilevel operator

Section 2.3 listed the three cases in which $\boldsymbol{i}$ is a valid node at level l, but $\boldsymbol{j} = \boldsymbol{i} \pm \boldsymbol{e}^d$ is not.

If $\boldsymbol{j}$ is on the physical boundary (case (a)), then there is no error in taking $\varphi_{\boldsymbol{j}}^{N,l}$ from the boundary condition, so we still have $(\tau_h)_{\boldsymbol{i}} = \mathrm{O}(h^2)$.

If $\boldsymbol{j}$ is covered by a grid at finer level $l+1$ (case (b)), then there is no error in projecting the valid data from the finer level, $\varphi_{\boldsymbol{j}}^{N,l} = \varphi_{n_{\mathrm{ref}}^l \boldsymbol{j}}^{N,l+1}$, so again, $(\tau_h)_{\boldsymbol{i}} = \mathrm{O}(h^2)$.

The rest of this section treats truncation error when $\varphi_{\boldsymbol{j}}^{N,l}$ is interpolated from $\varphi^{N,l-1}$ at the valid coarser-level nodes (case (c)). The truncation error τ_h is at best $\mathrm{O}(h^2)$, as it is with a single-level discrete Laplacian operator as shown in Section 2.2. Because of the h^2 denominator in (3), the truncation error is two orders of accuracy below that of the approximation error at the interface point.

When interpolating along a one-dimensional interface, with (5) and the four-point formula (6a), the interpolation is actually cubic, and hence fourth-order accurate

$$f_2(a) = \varphi^{N,l}(a,b) + \mathrm{O}(h^4), \tag{11}$$

so that when using the stencil (2) and (3) with such a point, the truncation error is $\tau_h(a,b) = \mathrm{O}(h^2)$. Formulas (6b) and (6c) with (5) are for quadratic interpolation and hence give third-order accuracy

$$f_2(a) = \varphi^{N,l}(a,b) + \mathrm{O}(h^3), \tag{12}$$

which with (2) and (3) yields truncation error of $\tau_h(a,b) = \mathrm{O}(h)$.

Interpolating across a two-dimensional interface with (7) and (8), if all four second partial derivatives along lines are calculated with the four-point formula (6a), then the two-dimensional biquadratic approximation is fourth-order accurate

$$f_2(a,b) = \varphi^{N,l}(a,b,c) + \mathrm{O}(h^4), \tag{13}$$

so yields truncation error of $\tau_h(a,b,c) = \mathrm{O}(h^2)$ when using the stencil (2) and (3) with such a point. If (6b) or (6c) is used for one or more second partial derivatives, then the formula for f_2 is third-order accurate

$$f_2(a,b) = \varphi^{N,l}(a,b,c) + \mathrm{O}(h^3) \tag{14}$$

yielding truncation error of $\tau_h(a,b,c) = \mathrm{O}(h)$.

2.7. Solution error

If the computed solution with mesh spacing h is φ^h, then the solution error $\xi_h = \varphi^h - \varphi^e$ satisfies the equations

$$\Delta^h \xi_h = \tau_h, \quad \xi_h|_{\partial\Omega} = 0. \tag{15}$$

As in [7], we can view this equation as being approximated by a continuous potential theory problem for (1), in which the charge density is piecewise constant in cells, being $(\tau_h)_{\boldsymbol{i}}$ in cell $\boldsymbol{i}$. Each cell has volume $h^{\mathbf{D}}$. The contribution of each cell to ξ_h is proportional to the total charge on the cell:

- For a cell not on the coarse/fine interface, truncation error is $\tau_h = \mathrm{O}(h^2)$, so total charge is $\mathrm{O}(h^{\mathbf{D}+2})$. There are $\mathrm{O}(1/h^{\mathbf{D}})$ such cells, leading to a contribution of $\mathrm{O}(h^2)$ to ξ_h.
- For a cell on the coarse/fine interface, the truncation error is $\tau_h = \mathrm{O}(h)$, so total charge is $\mathrm{O}(h^{\mathbf{D}+1})$. There are $\mathrm{O}(1/h^{\mathbf{D}-1})$ such cells, leading to a contribution of $\mathrm{O}(h^2)$ to ξ_h.

Hence we expect solution error of $\xi_h = \mathrm{O}(h^2)$. From a smooth solution, then, the error in the gradient computed according to Section 2.5 is $\eta_h = \mathrm{O}(h^2)$, for points on the coarse/fine interface as well as for points away from the interface.

3. Generalizing to non-rectangular domains

3.1. Operators on non-rectangular domains

When the domain Ω is not rectangular, the regular stencil (2) and (3) for the discrete Laplacian operator at a valid node may include points that are not in the domain. On a level with mesh spacing h, we use a discretization of the form

$$(\Delta^h \varphi^{N,l})_{\boldsymbol{i}} = c_{\boldsymbol{i}}^0 \cdot \varphi_{\boldsymbol{i}}^{N,l} + \sum_{d=0}^{\mathbf{D}-1} \left(c_{\boldsymbol{i}}^{2d+1} \cdot \varphi_{\boldsymbol{i}+\boldsymbol{e}^d}^{N,l} + c_{\boldsymbol{i}}^{2d+2} \cdot \varphi_{\boldsymbol{i}-\boldsymbol{e}^d}^{N,l} \right) + c_{\boldsymbol{i}}^{2\mathbf{D}+1}. \tag{16}$$

For the regular stencil, the coefficients are $c_{\boldsymbol{i}}^0 = -2\mathbf{D}/h^2$; $c_{\boldsymbol{i}}^{2d+1} = c_{\boldsymbol{i}}^{2d+2} = 1/h^2$ for $d = 0, \ldots, \mathbf{D}-1$; and $c_{\boldsymbol{i}}^{2\mathbf{D}+1} = 0$. The constant term $c_{\boldsymbol{i}}^{2\mathbf{D}+1}$ is nonzero when there are inhomogeneous internal boundary conditions, as shown below. In this case, the discrete operator is affine, not linear, and before applying the solver in Section 4, the problem must first be converted to residual–correction form.

At points near an internal boundary, where not all points of the regular stencil lie in the domain, we make an approximation for $(\Delta_d^h \varphi^{N,l})_{\boldsymbol{i}}$ $(d = 0, \ldots, \mathbf{D}-1)$ that is based on the regular 3-point stencil (3) and

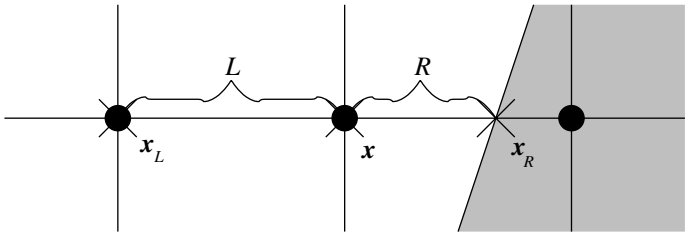

Fig. 5. Points (marked by ×) and distances used in approximating near an internal boundary, when the shaded area is outside the domain. In this example, $\varphi_{\mathrm{L}} = \varphi(\mathbf{x}_{\mathrm{L}}) = \varphi^{N,l}_{\mathbf{i}-\mathbf{e}^d}$, and $\varphi_{\mathrm{R}} = \varphi(\mathbf{x}_{\mathrm{R}})$ is obtained from the Dirichlet boundary condition.

extrapolation from available data to covered points of the stencil. For a point of evaluation at index $\boldsymbol{i}$, corresponding to point in space $\boldsymbol{x} = \boldsymbol{x}_0 + h \cdot \boldsymbol{i}$, we define $\boldsymbol{x}_{\mathrm{L}}$ and $\boldsymbol{x}_{\mathrm{R}}$ as follows (see Fig. 5):

- If $\boldsymbol{i} - \boldsymbol{e}^d \in \Omega$, then set $\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{x} - h \cdot \boldsymbol{e}^d$. Otherwise, set $\boldsymbol{x}_{\mathrm{L}}$ to be the first point on the segment from $\boldsymbol{x}$ to $\boldsymbol{x} - h \cdot \boldsymbol{e}^d$ that lies on the internal boundary.
- If $\boldsymbol{i} + \boldsymbol{e}^d \in \Omega$, then set $\boldsymbol{x}_{\mathrm{R}} = \boldsymbol{x} + h \cdot \boldsymbol{e}^d$. Otherwise, set $\boldsymbol{x}_{\mathrm{R}}$ to be the first point on the segment from $\boldsymbol{x}$ to $\boldsymbol{x} + h \cdot \boldsymbol{e}^d$ that lies on the internal boundary.

We also define φ_{L} as the value of φ at $\boldsymbol{x}_{\mathrm{L}}$ from either $\varphi^{N,l}_{\boldsymbol{i}-\boldsymbol{e}^d}$ or the boundary condition, and similarly φ_{R} as the value of φ at $\boldsymbol{x}_{\mathrm{R}}$ from either $\varphi^{N,l}_{\boldsymbol{i}+\boldsymbol{e}^d}$ or the boundary condition. We set distances $L = ||\boldsymbol{x} - \boldsymbol{x}_{\mathrm{L}}||$ and $R = ||\boldsymbol{x}_{\mathrm{R}} - \boldsymbol{x}||$.

With a Dirichlet boundary condition, two different approximations are used:

- The Shortley–Weller approximation [11], based on quadratic extrapolation from $\varphi^{N,l}_{\boldsymbol{i}}$, φ_{L} and φ_{R} to the covered grid nodes of the regular stencil

$$\left(\Delta^h_d \varphi^{N,l}\right)_{\boldsymbol{i}} = \frac{2}{L \cdot (L+R)} \varphi_{\mathrm{L}} - \frac{2}{L \cdot R} \varphi^{N,l}_{\boldsymbol{i}} + \frac{2}{R \cdot (L+R)} \varphi_{\mathrm{R}}. \tag{17}$$

The error in this approximation to the second derivative is $\mathrm{O}(L - R) + \mathrm{O}(L^2 + R^2) = \mathrm{O}(h)$.

- Linear extrapolation to the covered grid nodes of the regular stencil

$$\left(\Delta^h_d \varphi^{N,l}\right)_{\boldsymbol{i}} = \frac{1}{h \cdot L} \varphi_{\mathrm{L}} - \frac{1}{h} \left(\frac{1}{L} + \frac{1}{R} \right) \varphi^{N,l}_{\boldsymbol{i}} + \frac{1}{h \cdot R} \varphi_{\mathrm{R}}. \tag{18}$$

The error in this approximation to the second derivative is $\mathrm{O}((L + R)/h) = \mathrm{O}(1)$.

The coefficients $c^0_{\boldsymbol{i}}, \ldots, c^{2\mathbf{D}+1}_{\boldsymbol{i}}$ in (16) are the sums of coefficients in (17) or (18) over d from 0 to $\mathbf{D} - 1$. In particular, $c^0_{\boldsymbol{i}}$ is the sum of the coefficients of $\varphi^{N,l}_{\boldsymbol{i}}$ in Eqs. (17) or (18). For a particular dimension d, the coefficient of φ_{L} contributes to either $c^{2d+2}_{\boldsymbol{i}}$ or $c^{2\mathbf{D}+1}_{\boldsymbol{i}}$, depending on whether $\boldsymbol{x}_{\mathrm{L}}$ is a grid point or a boundary point. Likewise, the coefficient of φ_{R} contributes to either $c^{2d+1}_{\boldsymbol{i}}$ or $c^{2\mathbf{D}+1}_{\boldsymbol{i}}$, depending on whether $\boldsymbol{x}_{\mathrm{R}}$ is a grid point or a boundary point.

3.2. Coarse/fine boundary interpolation

If the domain is not rectangular, then coarse/fine boundary interpolation is more complicated. Near a fine node on the coarse/fine interface, some of the coarse-node neighbors may be outside the domain.

As in the case of rectangular domains, we assume a refinement ratio of 2. For higher refinement ratios, we perform a composition of interpolations with refinement ratio of 2.

Again, if a level-l node $\boldsymbol{j}$ of the stencil for the discrete Laplacian at $\boldsymbol{i}$ lies on the physical boundary, then we apply the physical boundary conditions; and if $\boldsymbol{j}$ coincides with a node at level $l - 1$, then we project the value of $\varphi^{N,l-1}_{\boldsymbol{j}/2}$. In other cases, where possible we use quadratic interpolation in two dimensions, biquadratic

interpolation in three dimensions. If some of the required coarse points are covered, then the interpolation formulas are modified as shown below.

For non-rectangular domains, our algorithms depend on whether or not nodes are *reachable* from other nodes. In this context, "reachable" means that the segment connecting the nodes lies entirely in the domain.

3.2.1. Two-dimensional problem: one-dimensional interface

In the two-dimensional problem, the coarse/fine interface is one-dimensional. As in Section 2.4.1, let (a, b) be the coordinates of a point on the interface over which the first coordinate varies. Let h be the finer-level mesh spacing. We assume that at least one of the coarse neighbors $(a - h, b)$ and $(a + h, b)$ is reachable from a. If both are reachable, then we use the formulas of Section 2.4.1.

If $(a + h, b)$ is *not* reachable from (a, b), then we must use one-sided formulas:

1. If $(a - h, b)$ is reachable from (a, b) but $(a - 3h, b)$ is not, then we use

$$f_0(a) = f_0(a - h).$$

2. If $(a - 3h, b)$ is reachable from (a, b) but $(a - 5h, b)$ is not, then we use

$$f_1(a) = \tfrac{3}{2} f_1(a - h) - \tfrac{1}{2} f_1(a - 3h).$$

3. If $(a - 5h, b)$ is reachable from (a, b), then we use

$$f_2(a) = \tfrac{15}{8} f_2(a - h) - \tfrac{5}{4} f_2(a - 3h) + \tfrac{3}{8} f_2(a - 5h).$$

3.2.2. Three-dimensional problem: two-dimensional interface

In the three-dimensional problem, the coarse/fine interface is two-dimensional. As in Section 2.4.2, let (a, b, c) be the coordinates of a point on the interface, along which the first two coordinates vary but the third is fixed. If either a or b is even then we interpolate along the line, following Section 3.2.1. In the remainder of this section, we assume both a and b are odd.

Letting h be the finer-level mesh spacing, then the nearest coarse neighbors of (a, b, c) on the interface are $(a \pm h, b \pm h, c)$. We assume that at least one of these is reachable from (a, b, c). If all are reachable, then we can use the formulas of Section 2.4.2.

If not all four of the nearest coarse neighbors on the interface are reachable from (a, b, c), then the first approximation is the mean of the values at the subset of these coarse neighbors that are in the domain, except that if the number is three, then one of them is ignored. See Fig. 6 for the five cases: (i) one coarse neighbor; (ii) two coarse neighbors, on a line parallel to an axis; (iii) two coarse neighbors, not on a line parallel to an axis; (iv) three coarse neighbors; (v) four coarse neighbors. In case (iv), the first approximation is the mean of the values at the two diagonally opposite coarse neighbors, as in case (iii). Note that for fully resolved geometries, except on thin domains, all cases may occur except (iii).

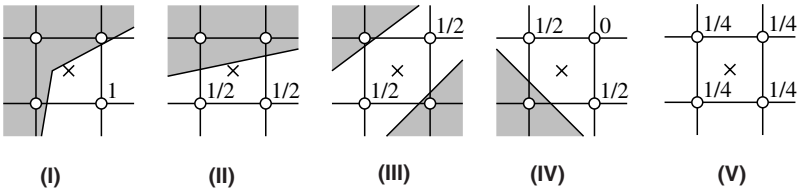

Fig. 6. The first approximation at the center node indicated by × is a weighted average of the neighboring coarse nodes that are in the domain. Weights are indicated beside the coarse nodes.

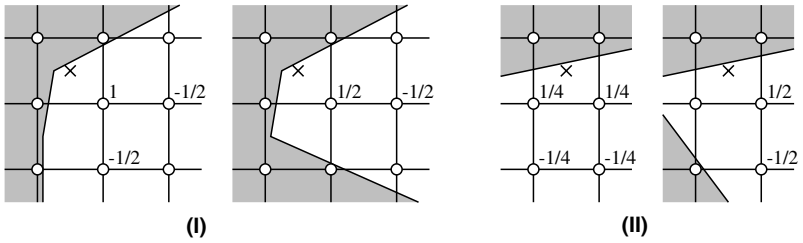

Fig. 7. One-sided bilinear correction to approximation at center node $\times$. Weights of the correction are indicated beside the coarse nodes: (i) If only one coarse neighbor of $\times$ is in the domain, we add a correction in each dimension where it is possible. With the correction coefficients on the left, the approximation becomes bilinear, but with the correction coefficients on the right, there are not enough points for a bilinear approximation. (ii) If two coarse neighbors of $\times$ are in the domain, and they lie on a line parallel to an axis, then we add a correction in the perpendicular direction if possible.

In cases (iii), (iv) and (v), this first approximation is bilinear. In cases (i) and (ii), in order to get a bilinear approximation we need to add a correction. Case (i) requires a correction in both directions, and case (ii) requires a correction in the direction perpendicular to the line containing the two coarse neighbors. See Fig. 7 for the coefficients of the corrections.

Adding to the bilinear approximation, we can make a biquadratic correction if there are enough coarse nodes in the domain to obtain estimates of the second derivatives. We use one of the formulas (6) for estimating the second derivative in each dimension, where the correction term takes the form $-\frac{h^2}{2}\frac{\partial^2 f_2}{\partial x^2}(a,b)$ or $-\frac{h^2}{2}\frac{\partial^2 f_2}{\partial y^2}(a,b)$. In cases (i) or (ii) above, in each dimension where a one-sided biquadratic correction is made we also need an additional correction of $\frac{(2h)^2}{2}\frac{\partial^2 f_2}{\partial x^2}(a,b)$ or $\frac{(2h)^2}{2}\frac{\partial^2 f_2}{\partial y^2}(a,b)$.

3.3. Computing gradients

When the domain Ω is not rectangular, the regular stencil (9) and (10) for the gradient at a valid node may include points that are not in the domain. On a level with mesh spacing h, we use a discretization of the form

$$(\nabla^h \varphi^{N,l})_{\boldsymbol{i}} = \sum_{d=0}^{\mathbf{D}-1} \left(g_{\boldsymbol{i}}^{4d} \cdot \varphi_{\boldsymbol{i}}^{N,l} + g_{\boldsymbol{i}}^{4d+1} \cdot \varphi_{\boldsymbol{i}+\boldsymbol{e}^d}^{N,l} + g_{\boldsymbol{i}}^{4d+2} \cdot \varphi_{\boldsymbol{i}-\boldsymbol{e}^d}^{N,l} + g_{\boldsymbol{i}}^{4d+3} \right) \cdot \boldsymbol{e}^d.$$

For the regular stencil, the coefficients are $g_{\boldsymbol{i}}^{4d+1} = 1/(2h)$, $g_{\boldsymbol{i}}^{4d+2} = -1/(2h)$, and $g_{\boldsymbol{i}}^{4d} = g_{\boldsymbol{i}}^{4d+3} = 0$ for $d = 0, \ldots, \mathbf{D}-1$.

At points near an internal boundary, where not all points of the regular stencil lie in the domain, we make an approximation for $D_d^h \varphi^{N,l}$ based on the regular stencil (9) and (10) and quadratic extrapolation from available data to covered points of the stencil. Using the same notation as Section 3.1 and Fig. 5, we set

$$\left(D_d^h \varphi^{N,l}\right)_{\boldsymbol{i}} = \frac{-R}{L \cdot (L+R)} \varphi_{\mathrm{L}} + \frac{R-L}{L \cdot R} \varphi_{\boldsymbol{i}}^{N,l} + \frac{L}{R \cdot (L+R)} \varphi_{\mathrm{R}}. \tag{19}$$

The error in this approximation to the first derivative is $\mathrm{O}(LR) = \mathrm{O}(h^2)$.

The same coarse/fine boundary interpolation described in Section 3.2 for computing the discrete Laplacian is also used for computing the gradient.

3.4. Truncation error

Applied at an internal boundary point $\boldsymbol{i}$, the Shortley–Weller formula (17) gives rise to truncation error $(\tau_h)_{\boldsymbol{i}} = \mathrm{O}(h)$, and the linear extrapolation formula (18) gives rise to truncation error $(\tau_h)_{\boldsymbol{i}} = \mathrm{O}(1)$.

In the two-dimensional problem, along a one-dimensional interface, if we use linear interpolation (4) at (a, b) then truncation error is $\tau_h(a, b) = \mathrm{O}(1)$. As discussed in Section 2.6, if we use cubic interpolation (6a) then $\tau_h(a, b) = \mathrm{O}(h^2)$, and if we use quadratic interpolation, (6b) or (6c), then $\tau_h(a, b) = \mathrm{O}(h)$. For the one-sided cases listed in Section 3.2.1

1. Approximation by $(a - h, b)$ alone yields first-order accuracy

$$f_0(a) = \varphi^{N,I}(a, b) + \mathrm{O}(h)$$

and hence truncation error $\tau_h(a, b) = \mathrm{O}(1/h)$.

2. Interpolation from $(a - h, b)$ and $(a - 3h, b)$ yields second-order accuracy

$$f_1(a) = \varphi^{N,I}(a, b) + \mathrm{O}(h^2)$$

and truncation error $\tau_h(a, b) = \mathrm{O}(1)$.

3. Interpolation from $(a - h, b)$, $(a - 3h, b)$, and $(a - 5h, b)$ yields third-order accuracy

$$f_2(a) = \varphi^{N,I}(a, b) + \mathrm{O}(h^3)$$

and truncation error $\tau_h(a, b) = \mathrm{O}(h)$.

Case 3 above is the only one that occurs in geometries that are resolved fully.

In three dimensions, the bilinear approximation f_1 across a two-dimensional interface is second-order accurate

$$f_1(a, b) = \varphi^{N,I}(a, b, c) + \mathrm{O}(h^2),$$

so truncation error is $\tau_h(a, b, c) = \mathrm{O}(1)$. If we have data at enough coarse-level nodes to make a biquadratic correction, as we can do with a rectangular domain and also expect with fully resolved geometries, then the new approximation f_2 is third-order accurate

$$f_2(a, b) = \varphi^{N,I}(a, b, c) + \mathrm{O}(h^3)$$

yielding truncation error of $\tau_h(a, b, c) = \mathrm{O}(h)$.

3.5. Solution error

In Section 2.6, we showed that away from coarse/fine interfaces and internal boundaries, we have truncation error $\tau_h = \mathrm{O}(h^2)$. The analysis in Section 3.4 shows that with fully resolved geometries, we have truncation error $\tau_h = \mathrm{O}(h)$ everywhere else, except on internal boundaries if the discrete Laplacian is computed by the linear extrapolation formula (18), in which case $\tau_h = \mathrm{O}(1)$ at these points.

As in Section 2.7, we can view the modified Eq. (15) for the solution error ξ_h as a potential theory problem for (1), in which the charge density is piecewise constant $(\tau_h)_{\boldsymbol{i}}$ in cell $\boldsymbol{i}$. The contribution of each cell to ξ_h is proportional to the total charge on the cell:

- For a cell not on the coarse/fine interface and not intersecting an internal boundary, truncation error is $\tau_h = \mathrm{O}(h^2)$, so total charge is $\mathrm{O}(h^{\mathbf{D}+2})$. There are $\mathrm{O}(1/h^{\mathbf{D}})$ such cells, leading to a contribution of $\mathrm{O}(h^2)$ to ξ_h.
- For a cell on the coarse/fine interface, the truncation error is $\tau_h = \mathrm{O}(h)$, so total charge is $\mathrm{O}(h^{\mathbf{D}+1})$. There are $\mathrm{O}(1/h^{\mathbf{D}-1})$ such cells, hence a contribution of $\mathrm{O}(h^2)$ to ξ_h.

- For a cell intersecting an internal boundary, if the linear extrapolation formula (18) is used, then $\tau_h = O(1)$ so total charge is $O(h^{\mathbf{D}})$. But because there is a homogeneous Dirichlet boundary condition on ξ_h, the effect of the field induced by the charge in the partial cell can be represented as an image charge of the same strength, but of opposite sign located a distance $O(h)$ from the internal-boundary cell just outside the boundary of the domain [7]. The result is that the contribution to ξ_h is a dipole field that is one order smaller in h, hence $O(h^{\mathbf{D}+1})$. There are $O(1/h^{\mathbf{D}-1})$ such cells, leading to a contribution of $O(h^2)$ to ξ_h. If the Shortley–Weller approximation (17) is used instead of linear extrapolation, then the truncation error $\tau_h = O(h)$ contributes $O(h^3)$ to ξ_h, a result that is proved under smoothness conditions by Matsunaga and Yamamoto [9].

Hence we expect solution error of $\xi_h = O(h^2)$, even when derivatives at internal boundaries are computed by linear extrapolation.

Away from internal boundaries, the error in the gradient of the solution as computed by the regular stencil would then be $\eta_h = O(h^2)$, even on the coarse/fine interface. At points adjacent to internal boundaries, the solution error gives rise to error in the gradient of $\eta_h = O(h)$ if the discrete Laplacian is from linear extrapolation (18), and $\eta_h = O(h^2)$ if from Shortley–Weller (17).

4. AMR multigrid algorithm

Pseudo-code for the AMR multigrid algorithm [2] to solve $L(\varphi) = \rho$ is shown in Fig. 9. Here L is the multilevel operator on the composite array. We note that for convergence of a solution with node-centered data, there must be at least one non-periodic dimension, because with periodic boundary conditions in all

```
procedure AMRmgRelax(φ^f, R^f, r)
{
    for i = 1, ..., NumBottomGSRB
        LevelGSRB(φ^f, R^f)
    end for
    if (r > 2) then
        for i = 1, ..., NumSmoothDown
            LevelGSRB(φ^f, R^f)
        end for
        δ^c := 0
        R^c := Average(R^f − L^nf(φ^f, φ^c ≡ 0))
        AMRmgRelax(δ^c, R^c, r/2)
        φ^f := φ^f + Interp(δ^c)
        for i = 1, ..., NumSmoothUp
            LevelGSRB(φ^f, R^f)
        end for
    end if
}

procedure LevelGSRB(φ^f, R^f)
{
    φ^f := φ^f + λ(L^nf(φ^f, φ^c ≡ 0) − R^f) on Ω_N^BLACK
    φ^f := φ^f + λ(L^nf(φ^f, φ^c ≡ 0) − R^f) on Ω_N^RED
}
```

Fig. 8. Recursive relaxation procedure using Gauss–Seidel relaxation with red–black ordering [2,3]. We set NumSmoothUp = 4, NumSmoothDown = 4, and NumBottomGSRB = 8. The relaxation parameter at a point $\boldsymbol{i}$ is $\lambda = -1/c_{\mathbf{i}}^0$ with $c_{\mathbf{i}}^0$ the diagonal coefficient in (16).

$R := \rho - L(\varphi)$
while $(||R|| > \epsilon||\rho||)$
    AMRVCycleMG(l^{max})
    $R := \rho - L(\varphi)$
end while
procedure AMRVCycleMG(level l):
{
    if $(l = l^{max})$ then $R^l := \rho^l - L^{nf}(\varphi^l, \varphi^{l-1})$
    if $(l > 0)$ then
        $\varphi^{l,save} := \varphi^l$ on Ω_N^l
        $e^l := 0$ on Ω_N^l
        AMRmgRelax$(e^l, R^l, n_{ref}^{l-1})$
        $\varphi^l := \varphi^l + e^l$
        $e^{l-1} := 0$ on Ω_N^{l-1}
        $R^{l-1} := Average(R^{l-1} - L^{nf}(e^l, e^{l-1}))$ on $\mathcal{C}_{n_{ref}^{l-1}}(\Omega_N^l)$
        $R^{l-1} := \rho^{l-1} - L^{comp,l-1}(\varphi)$ on $\Omega_N^{l-1} - \mathcal{C}_{n_{ref}^{l-1}}(\Omega_N^l)$
        AMRVCycleMG$(l-1)$
        $e^l := e^l + Interp(e^{l-1})$
        $R^l := R^l - L^{nf,l}(e^l, e^{l-1})$
        $\delta e^l := 0$ on Ω_N^l
        AMRmgRelax$(\delta e^l, R^l, n_{ref}^{l-1})$
        $e^l := e^l + \delta e^l$
        $\varphi^l := \varphi^{l,save} + e^l$
    else
        solve $L^{nf}(e^0, 0) = R^0$ on Ω_N^0, by one-level multigrid algorithm.
        $\varphi^0 := \varphi^0 + e^0$
    end if
}

Fig. 9. Pseudo-code description of the AMR multigrid algorithm.

dimensions on multiple levels of data, it is not possible to set integral weights such that the integral over the whole domain of the right-hand side is zero.

The operator at level l, $L^{\mathrm{comp},l}$, depends not only on level l but also on the coarser level $l-1$ and finer level $l+1$, as outlined in Section 2.2. The pseudo-code also refers to the no-finer-level operator $L^{\mathrm{nf}}(\varphi^{\mathrm{f}}, \varphi^{\mathrm{c}})$, which operates on the level of φ^{f}, with φ^{c} as the data at the next coarser level, and no finer level.

The smoothing operator `AMRmgRelax`$(\varphi^{\mathrm{f}}, R^{\mathrm{f}}, r)$, described in detail in Fig. 8, performs a mini-V-cycle iteration on φ^{f} for L^{nf} and right-hand side R^{f}, assuming the coarse-grid values required for the boundary conditions are identically zero. The number r is the refinement ratio. This procedure uses two routines described in Appendix: *Average* for fine-to-coarse averaging (see Section A.1), and *Interp* for coarse-to-fine interpolation (see Section A.2).

5. Convergence tests

Before describing the convergence tests, we define the norms we use to measure error on multilevel hierarchies. Let $f^{\mathrm{comp}} = \{f^l\}_{l=0}^{l_{\max}}$ be a composite array defined on the nodes of each level. The L_∞ norm of f^l is defined as the maximum value over the valid nodes at level l:

$$||f^l||_\infty = \max_{\boldsymbol{i} \in \Omega_{N,\mathrm{valid}}^l} |f_{\boldsymbol{i}}^l|.$$

The composite L_∞ norm is the maximum over all levels

$$||f^{\text{comp}}||_\infty = \max_{l=0}^{l_{\max}} ||f^l||_\infty.$$

For finite p we define the L_p norm of f^l by integrating $|f^l|^p$ over the valid nodes. If the mesh spacing on level l is h_l, then

$$||f^l||_p = \left(h_l^{\mathbf{D}} \sum_{\boldsymbol{i} \in \Omega^l_{N,\text{valid}}} |f^l_{\boldsymbol{i}}|^p \right)^{1/p}.$$

The composite L_p norm is the pth root of the sum of pth powers of the norms of each level

$$||f^{\text{comp}}||_p = \left(\sum_{l=0}^{l_{\max}} ||f^l||_p^p \right)^{1/p} = \left(\sum_{l=0}^{l_{\max}} h_l^{\mathbf{D}} \sum_{\boldsymbol{i} \in \Omega^l_{N,\text{valid}}} |f^l_{\boldsymbol{i}}|^p \right)^{1/p}.$$

If error e_h is computed with mesh spacing h, and error e_{2h} with mesh spacing $2h$, then the convergence rate of their norms is defined as

$$p = \log_2 \left(\frac{||e_{2h}||}{||e_h||} \right). \tag{20}$$

Thus $e_h = \mathrm{O}(h^p)$, so $p = 1$ indicates first-order accuracy, and $p = 2$ indicates second-order accuracy.

5.1. Solver tests on a rectangular domain

We initialize the right-hand side ρ of the solver to be a function for which we can compute the analytical solution φ in (1). We compute the solution on the two levels at different base refinements, and compare the solutions U_h with base-level mesh spacing h to the exact solutions U_e. The boundary conditions of the computed solution are inhomogeneous Dirichlet. The value at the boundary is set to the analytic solution at that location. We find the difference between the computed and exact solutions, $\xi_h = U_h - U_e$, and between the computed and exact values of the gradient, $\eta_h = \nabla^h U_h - \nabla U_e$. We also evaluate the truncation error $\tau_h = \rho - \Delta^h U_e$.

Problem 1 (*polynomial on rectangular domain*). We initialize the charge density ρ to be a function of r

$$\rho(r) = \begin{cases} \rho_0 \left(2\left(\frac{r}{R_0}\right)^3 - 3\left(\frac{r}{R_0}\right)^2 + 1 \right), & \text{if } r < R_0, \\ 0, & \text{if } r \geqslant R_0 \end{cases} \tag{21}$$

and set boundary conditions for the exact solution

$$U_e(r) = \begin{cases} \rho_0 r^2 \left(\frac{1}{6} - \frac{3}{20}\left(\frac{r}{R_0}\right)^2 + \frac{1}{15}\left(\frac{r}{R_0}\right)^3 \right), & \text{if } r < R_0, \\ \rho_0 R_0^2 \left(\frac{3}{20} - \frac{1}{15}\frac{R_0}{r} \right), & \text{if } r \geqslant R_0. \end{cases} \tag{22}$$

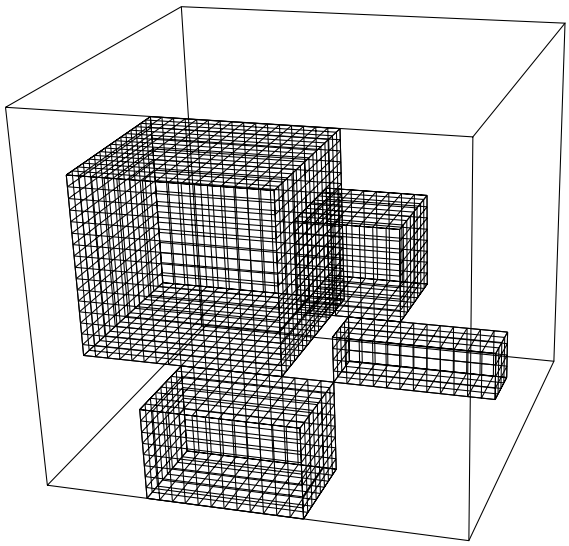

Fig. 10. Grid configuration for Problem 1. There are two levels of refinement. The edges of the coarser-level grid and the face cells of the finer-level grids are shown. The refinement ratio between the levels is two. In this illustration, the coarser-level domain contains one $16 \times 16 \times 16$ grid, and there are four grids at the finer level where the domain is $32 \times 32 \times 32$. We also use fully refined versions of this same set of grids, all partitioned so that the maximum length of any grid in any dimension is 32.

Table 1
Norms and convergence rates of solution error for Problem 1

h	$\|\|\xi_h\|\|_\infty$	p	$\|\|\xi_h\|\|_1$	p	$\|\|\xi_h\|\|_2$	p
1/32	1.49905×10^{-5}		2.23346×10^{-6}		3.69778×10^{-6}	
1/64	3.54130×10^{-6}	2.08	5.33136×10^{-7}	2.07	8.67701×10^{-7}	2.09
1/128	8.61632×10^{-7}	2.04	1.30171×10^{-7}	2.03	2.10033×10^{-7}	2.05
1/256	2.12408×10^{-7}	2.02	3.21560×10^{-8}	2.02	5.16580×10^{-8}	2.02

We use the two-level grid hierarchy shown in Fig. 10, on a domain that has unit length in each direction and is centered at the origin. Coarse/fine interpolation is biquadratic. In (21), we set $R_0 = 1/2$ and $\rho_0 = 3/4$.

In Table 1, we show the convergence of the error ξ_h in the two-level solution with mesh spacing h at the coarser level. Solution error converges quadratically to zero in all three norms, showing uniform second-order convergence of the solution across the domain.

Table 2 shows the convergence of the truncation error, τ_h, and the order of convergence of its norms. The truncation error converges to first order in L_∞ norm and to second order in L_1 and L_2 norms. These results are consistent with second-order convergence everywhere except on a set of codimension two, on which convergence is only first order. In the discretized problem space of $O(n^{\mathbf{D}}) = O(1/h^{\mathbf{D}})$ points, a set of co-

Table 2
Norms and convergence rates of truncation error for Problem 1

h	$\|\|\tau_h\|\|_\infty$	p	$\|\|\tau_h\|\|_1$	p	$\|\|\tau_h\|\|_2$	p
1/32	5.01139×10^{-3}		2.68607×10^{-4}		3.84822×10^{-4}	
1/64	2.59449×10^{-3}	0.95	6.63975×10^{-5}	2.02	9.46695×10^{-5}	2.02
1/128	1.27999×10^{-3}	1.02	1.65126×10^{-5}	2.01	2.35043×10^{-5}	2.01
1/256	6.35341×10^{-4}	1.01	4.11761×10^{-6}	2.00	5.85700×10^{-6}	2.00

Table 3
Norms and convergence rates of gradient error for Problem 1

h	$\|\eta_h\|_\infty$	p	$\|\eta_h\|_1$	p	$\|\eta_h\|_2$	p
1/32	1.80511×10^{-4}		5.65881×10^{-5}		7.27800×10^{-5}	
1/64	4.63199×10^{-5}	1.96	1.42181×10^{-5}	1.99	1.80996×10^{-5}	2.01
1/128	1.17451×10^{-5}	1.98	3.55935×10^{-6}	2.00	4.50787×10^{-6}	2.01
1/256	2.95639×10^{-6}	1.99	8.90189×10^{-7}	2.00	1.12451×10^{-6}	2.00

dimension two has $O(n^{\mathbf{D}-2}) = O(h^2/h^{\mathbf{D}})$ points. If τ_h is $O(h)$ on a set S of codimension two, but $O(h^2)$ elsewhere, then with each point weighted by $O(h^{\mathbf{D}})$ in the integrals:

$$||\tau_h||_1 = \int |\tau_h| = \int_S O(h) + \int_{\Omega - S} O(h^2) = O(h^2)O(h) + O(1)O(h^2) = O(h^2), \tag{23}$$

$$||\tau_h||_2^2 = \int |\tau_h|^2 = \int_S O(h)^2 + \int_{\Omega - S} O(h^2)^2 = O(h^2)O(h^2) + O(1)O(h^4) = O(h^4). \tag{24}$$

So $||\tau_h||_2 = O(h^2)$. These are the convergence properties expected from the analysis in Section 2.6 for bicubic interpolation on the coarse–fine interfaces except on the edges (a set of points of codimension two) where interpolation is biquadratic. The truncation error is $O(h)$ on the edges and $O(h^2)$ elsewhere.

Table 3 shows the convergence of the error in the gradient, η_h, and the order of convergence of its norms. The gradient error converges to second order in all three norms.

5.2. Solver tests on a non-rectangular domain

Problem 2 (*point charge inside a sphere*). In our tests of convergence on a non-rectangular domain, we use the following physical example. The domain lies between a cube with corners at $(\frac{1}{2}, 0, 0)$ and $(1, \frac{1}{2}, \frac{1}{2})$ and a sphere centered at $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ with radius $r = 1/\sqrt{35} = 0.169$, as illustrated in Fig. 11. We solve for the field within this domain due to a point charge at $\boldsymbol{q} = (0.52, 0.45, 0.49)$, from Laplace's equation

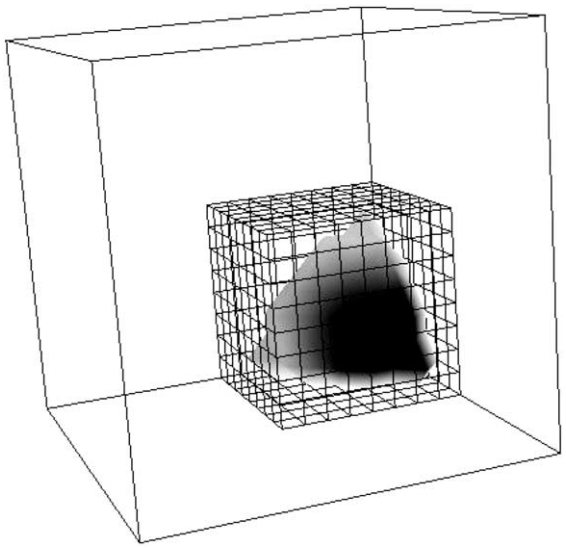

Fig. 11. In our examples, we solve for the field outside the shaded region due to a point charge within it. We solve on nested hierarchies of grids in the configuration shown with two levels of refinement.

$$\Delta\varphi = 0$$

with inhomogeneous Dirichlet boundary conditions for φ enforced on the sphere and on the faces of the cube.

We use biquadratic interpolation at coarse/fine interfaces, and either the Shortley–Weller approximation or linear extrapolation to compute the derivative near internal boundaries, as discussed in Section 3.1.

The exact solution is

$$\varphi(\boldsymbol{x}) = \frac{1}{||\boldsymbol{x} - \boldsymbol{q}||_2}. \tag{25}$$

In Tables 4 and 5, we show the norms of the solution error, ξ_h, with h the mesh spacing at the coarser level, and the order of convergence. We see that $||\xi_h|| = \mathrm{O}(h^2)$ in all three norms, with both Shortley–Weller and linear extrapolation, showing uniform second-order convergence of the solution across the domain. As shown by Gibou et al. [5], the rate of convergence of solution error is the same for both methods.

Tables 6 and 7 show norms of truncation error, τ_h, and order of convergence. With Shortley–Weller approximation, the truncation error norms are $||\tau_h||_\infty = \mathrm{O}(h)$, $||\tau_h||_1 = \mathrm{O}(h^2)$, and $||\tau_h||_2 = \mathrm{O}(h^{3/2})$. As can be

Table 4
Norms and convergence rates of solution error for Problem 2, Shortley–Weller

h	$\|\|\xi_h\|\|_\infty$	p	$\|\|\xi_h\|\|_1$	p	$\|\|\xi_h\|\|_2$	p
1/32	4.03907×10^{-3}		5.74837×10^{-5}		2.60164×10^{-4}	
1/64	1.07548×10^{-3}	1.91	1.40111×10^{-5}	2.04	6.26871×10^{-5}	2.05
1/128	2.73803×10^{-4}	1.97	3.44668×10^{-6}	2.02	1.53959×10^{-5}	2.03
1/256	6.91879×10^{-5}	1.98	8.54404×10^{-7}	2.01	3.81847×10^{-6}	2.01
1/512	1.73955×10^{-5}	1.99	2.12685×10^{-7}	2.01	9.51062×10^{-7}	2.01

Table 5
Norms and convergence rates of solution error for Problem 2, linear extrapolation

h	$\|\|\xi_h\|\|_\infty$	p	$\|\|\xi_h\|\|_1$	p	$\|\|\xi_h\|\|_2$	p
1/32	1.84283×10^{-2}		6.90030×10^{-5}		3.51467×10^{-4}	
1/64	5.64924×10^{-3}	1.71	1.70293×10^{-5}	2.02	8.71924×10^{-5}	2.01
1/128	1.83783×10^{-3}	1.62	4.22475×10^{-6}	2.01	2.18741×10^{-5}	1.99
1/256	4.88552×10^{-4}	1.91	1.05063×10^{-6}	2.01	5.46468×10^{-6}	2.00
1/512	1.26093×10^{-4}	1.95	2.62009×10^{-7}	2.00	1.36665×10^{-6}	2.00

Table 6
Norms and convergence rates of truncation error for Problem 2, Shortley–Weller

h	$\|\|\tau_h\|\|_\infty$	p	$\|\|\tau_h\|\|_1$	p	$\|\|\tau_h\|\|_2$	p
1/32	7.87124×10^{1}		3.46152×10^{-2}		5.58357×10^{-1}	
1/64	7.59587×10^{1}	0.05	1.04699×10^{-2}	1.73	2.85664×10^{-1}	0.97
1/128	4.04698×10^{1}	0.91	2.87025×10^{-3}	1.87	1.12606×10^{-1}	1.34
1/256	2.18532×10^{1}	0.89	7.38420×10^{-4}	1.96	3.98319×10^{-2}	1.50
1/512	1.09962×10^{1}	0.99	1.88378×10^{-4}	1.97	1.43731×10^{-2}	1.47

Table 7
Norms and convergence rates of truncation error for Problem 2, linear extrapolation

h	$\|\|\tau_h\|\|_\infty$	p	$\|\|\tau_h\|\|_1$	p	$\|\|\tau_h\|\|_2$	p
1/32	4.39705×10^2		7.53116×10^{-2}		2.82224×10^0	
1/64	4.49817×10^2	−0.03	3.29238×10^{-2}	1.19	2.02159×10^0	0.48
1/128	5.60579×10^2	−0.32	1.59776×10^{-2}	1.04	1.55940×10^0	0.37
1/256	5.46824×10^2	0.04	7.55070×10^{-3}	1.08	1.09924×10^0	0.50
1/512	5.51119×10^2	−0.01	3.74049×10^{-3}	1.01	7.92931×10^{-1}	0.47

Table 8
Norms and convergence rates of gradient error for Problem 2, Shortley–Weller

h	$\|\|\eta_h\|\|_\infty$	p	$\|\|\eta_h\|\|_1$	p	$\|\|\eta_h\|\|_2$	p
1/32	6.83011×10^{-1}		3.31943×10^{-3}		1.99067×10^{-2}	
1/64	2.29643×10^{-1}	1.57	8.83584×10^{-4}	1.91	5.22682×10^{-3}	1.93
1/128	6.70953×10^{-2}	1.78	2.27569×10^{-4}	1.96	1.33941×10^{-3}	1.96
1/256	1.77165×10^{-2}	1.92	5.77036×10^{-5}	1.98	3.38803×10^{-4}	1.98
1/512	4.95187×10^{-3}	1.84	1.45287×10^{-5}	1.99	8.52292×10^{-5}	1.99

Table 9
Norms and convergence rates of gradient error for Problem 2, linear extrapolation

h	$\|\|\eta_h\|\|_\infty$	p	$\|\|\eta_h\|\|_1$	p	$\|\|\eta_h\|\|_2$	p
1/32	3.30161×10^0		3.63901×10^{-3}		2.97216×10^{-2}	
1/64	2.51041×10^0	0.40	9.88397×10^{-4}	1.88	1.01984×10^{-2}	1.54
1/128	1.90368×10^0	0.40	2.62805×10^{-4}	1.91	4.00322×10^{-3}	1.35
1/256	9.23794×10^{-1}	1.04	6.73714×10^{-5}	1.96	1.38508×10^{-3}	1.53
1/512	5.26765×10^{-1}	0.81	1.71795×10^{-5}	1.97	5.05414×10^{-4}	1.45

seen by methods similar to those of (23) and (24), these norms are consistent with truncation error that is second order everywhere except a set of codimension one – the interior boundary – where it is only first order.

With linear extrapolation, the truncation error norms are $||\tau_h||_\infty = \mathrm{O}(1)$, $||\tau_h||_1 = \mathrm{O}(h)$, and $||\tau_h||_2 = \mathrm{O}(h^{1/2})$. These norms are consistent with truncation error that is second order everywhere except on a set of codimension one – the interior boundary – where it is zeroth order.

Tables 8 and 9 show norms of gradient error, η_h, and order of convergence. With Shortley–Weller approximation, the gradient error tends to quadratic in all three norms. With linear extrapolation, the gradient error norms are $||\eta_h||_\infty = \mathrm{O}(h)$, $||\eta_h||_1 = \mathrm{O}(h^2)$, and $||\eta_h||_2 = \mathrm{O}(h^{3/2})$, consistent with second order everywhere except on a set of codimension one – the internal boundary – where it is first order. Second-order convergence of the solution with first-order convergence of the gradient has also been observed by Jones and Menzies [8] in solving Poisson's equation on nonuniform grids. If a Poisson solver is to be used in conjunction with a particle-in-cell method, then the gradient must be calculated, so these results show an advantage of the Shortley–Weller approximation over linear extrapolation at internal boundaries.

5.3. Accelerator example

The high current, heavy-ion beam injector of the High Current Experiment [10] is used as an example. In this injector, an ion beam is extracted from an emitter in a triode configuration, and it then propagates into a set of electrostatic quadrupoles, where it is both accelerated and transversely confined. Fig. 12 shows the

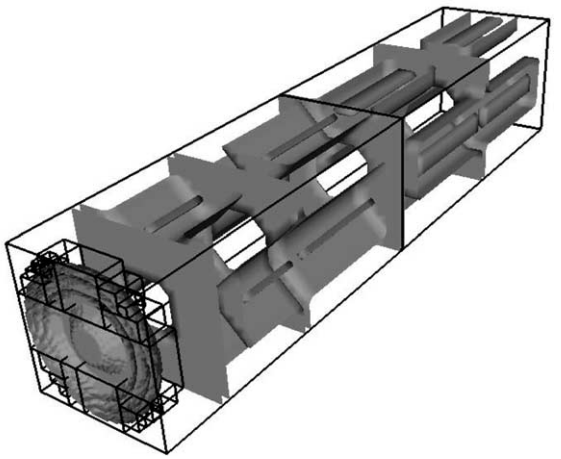

Fig. 12. Accelerator example, with grids shown at two levels of refinement. The surface shown separates grid points that are in the domain from grid points that are not; no other information was used in drawing the surface. The finer-level grids are clustered around the emitter, shown at the left.

overall configuration of the injector, with the emitter at the lower left corner of the image. The need of AMR in this particular example is driven by the need to resolve high gradients in density in the vicinity of the emitting surface and the beam edge, as well as adequately resolving the conductor geometry.

The modeling to date has been done with the Warp code [6], which provides a three-dimensional, time-dependent description. Warp is a multidimensional intense beam simulation program being developed and used at the Heavy Ion Fusion Virtual National Laboratory. The discrete-particle models in Warp combine the particle-in-cell (PIC) technique commonly used for plasma modeling with a description of the "lattice" of accelerator elements. The self-consistent field is assumed electrostatic – Poisson's equation is solved on a mesh that moves with the beam. The three-dimensional Poisson solver is limited to a Cartesian mesh with uniform spacing. Non-rectangular boundary conditions are included using the Shortley–Weller approximation. A prototype axisymmetric (r,z) AMR Poisson solver has been implemented in Warp and used to carry out time-dependent simulations of the injector. These simulations have shown the benefit and the effectiveness of the mesh refinement technique in converging to the right physical solution using less grid resolution [12,13].

In this example, the three-dimensional AMR solver is used to solve Laplace's equation, given the injector geometry as the boundary conditions. In a full simulation of the injector, Neumann boundary conditions would be applied at the edge of mesh. However, Neumann boundary conditions have not yet been implemented in the AMR solver, so Dirichlet boundaries are used. This limitation should not alter the conclusions on how effective the AMR solver is on the example, and will be once Neumann boundary conditions are implemented.

The illustration of the injector geometry in Fig. 12 shows grids at two levels of refinement. The coarser level, with mesh spacing 1/40, has two boxes that together cover a rectangular domain with a $16 \times 16 \times 80$ index space. The finer level has 23 boxes contained in the same rectangular domain with mesh spacing 1/160 in a $64 \times 64 \times 320$ index space. These finer-level boxes are concentrated near the emitter.

Fig. 13 shows the convergence of the residual, with the results for runs with fine-level mesh spacings of 1/80, 1/160, and 1/320, and either one level or two levels. In the two-level examples, the refinement ratio between the two levels is 4. As a measure of the amount of computation per iteration, Table 10 lists the number of cells and number of grids at each level for the one- and two-level examples with the different

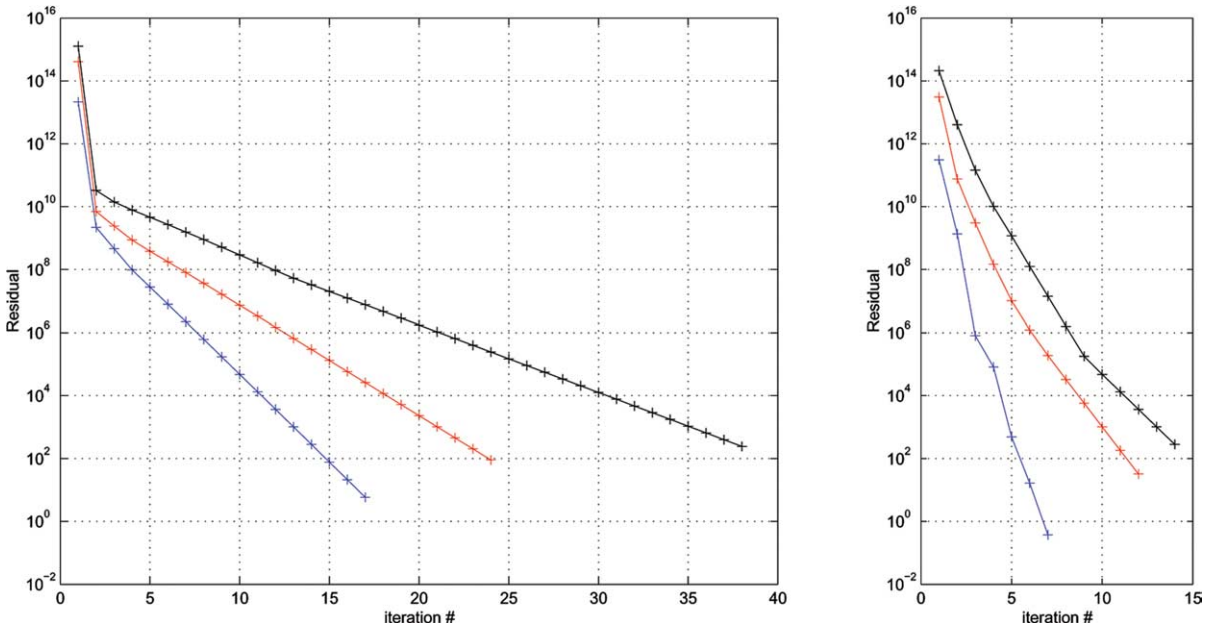

Fig. 13. Plot of the max norm of the residual with multigrid iteration number for the accelerator example on (left) one fully refined level, and (right) two levels, with finer-level grids around emitter. From top to bottom in each graph, the curves represent runs for fine-level mesh spacings of 1/320, 1/160, and 1/80.

Table 10
Number of cells and number of grids used in accelerator examples

Fine spacing	One level only		Finer of 2 levels		Coarser of 2 levels	
	Cells	Grids	Cells	Grids	Cells	Grids
1/80	163,840	4	8192	1	2560	1
1/160	1,310,720	8	39,936	23	20,480	2
1/320	10,485,760	64	198,592	781	163,840	4

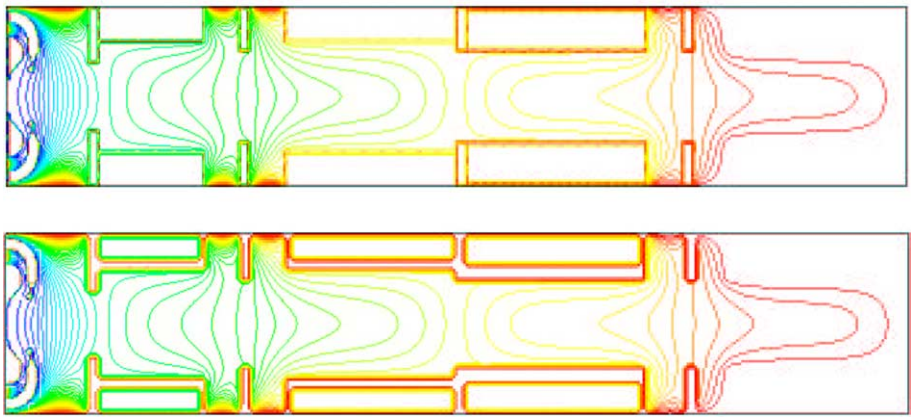

Fig. 14. Contour lines of potential on a two-dimensional slice through the center of the domain in the accelerator example, with (top) one fully refined level, and (bottom) two levels, with finer-level grids around emitter. The fine-level mesh spacing in both cases is 1/320. In the bottom plot, the coarse-level mesh spacing is 1/80.

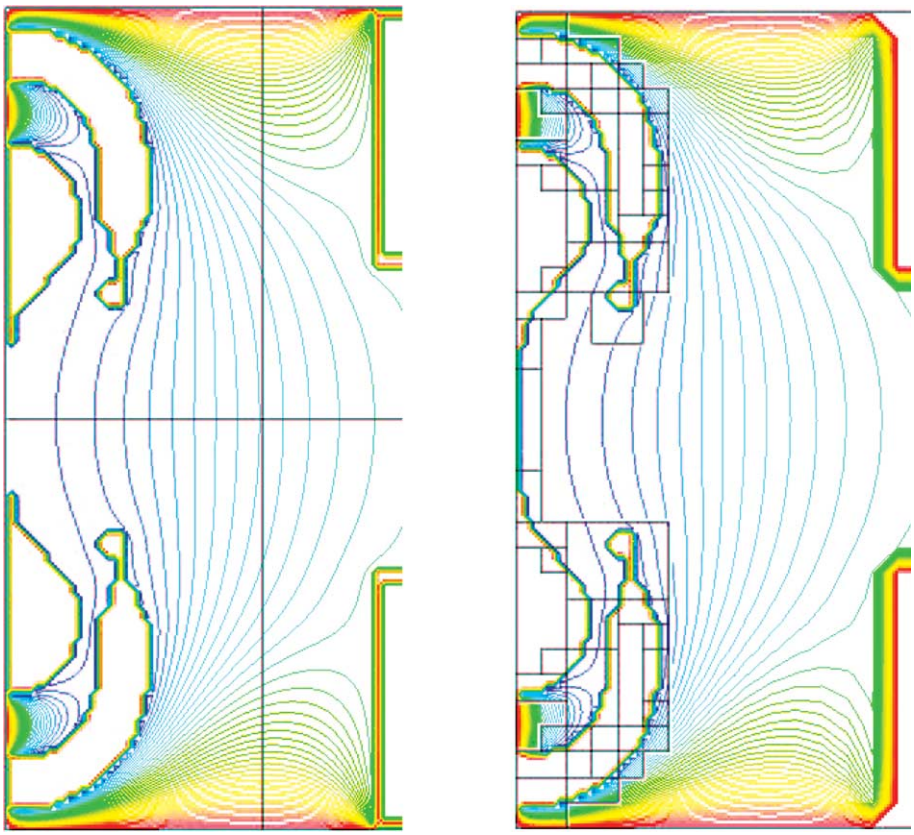

Fig. 15. Close-ups of the contour plots in Fig. 14, showing the box boundaries in black, with (left) one fully refined level, and (right) two levels, with finer-level grids around emitter.

mesh spacings. In Figs. 14 and 15, we show contour lines of the potential on a cross-section of the domain.

6. Conclusions

We have demonstrated a method of solving Poisson's equation with Dirichlet boundary conditions on adaptive meshes in complex geometries, such that the error in both the solution and the gradient is second order in the mesh spacing. We have applied this to a heavy-ion fusion accelerator problem and have illustrated the effectiveness of the method where the use of mesh refinement has significantly reduced the amount of work required to reach a given resolution where it is needed. We are applying our method to further work in PIC codes for the accelerator with particles.

Acknowledgements

This work was performed for USDOE under contracts DE-AC03-76F00098 at UC-LBNL and W-7405-ENG-48 at UC-LLNL. Research was supported in part by the Applied Mathematical Sciences program,

the SciDAC program, and the LBNL LDRD program. We thank Alex Friedman for useful discussions about the accelerator example, and Brian Van Straalen for helping to integrate the Warp code with the adaptive multigrid solver.

Appendix A. Interlevel transfer operators

A.1. Averaging

The *Average* operator is used to average from a finer level down to a coarser level, or to construct averaged residuals in multigrid iterations. Averaging from level l to level $l-1$, we assume that the refinement ratio $r = n_{\text{ref}}^{l-1}$ is a power of 2. To obtain the averaged value at node $\boldsymbol{i} \in \mathscr{C}_r(\Omega^l_{N,\text{int}})$ of level $l-1$, we use the values at the level-l nodes $r\boldsymbol{i}+\boldsymbol{q}$, where the components of $\boldsymbol{q}$ satisfy $-r/2 \leqslant q_d \leqslant r/2$ for $d = 0, \ldots, \mathbf{D}-1$.

A.1.1. On rectangular domains

For a refinement ratio of r, we define weights $w_{\boldsymbol{q}}$ on the points in

$$\Theta_r = \left\{ \boldsymbol{q} \in \mathbb{Z}^{\mathbf{D}} : -\frac{r}{2}\boldsymbol{u} \leqslant \boldsymbol{q} \leqslant \frac{r}{2}\boldsymbol{u} \right\}.$$

The averaging operator down to level $l-1$ is defined at nodes $\boldsymbol{i} \in \Omega_N^{l-1}$ where $\Theta_r + r\boldsymbol{i} \subset \Omega_N^l$. These are the interior coarse nodes of level l. Thus for $\boldsymbol{i} \in \mathscr{C}_r(\Omega^l_{N,\text{int}})$

$$\text{Average}(\varphi^{N,l})_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\Theta_r} w_{\boldsymbol{q}} \varphi^{N,l}_{r\boldsymbol{i}+\boldsymbol{q}}, \tag{A.1}$$

where the weight at point $\boldsymbol{q} = (q_0, \ldots, q_{\mathbf{D}-1}) \in \mathbb{Z}^{\mathbf{D}}$ is

$$w_{\boldsymbol{q}} = \prod_{d=0}^{\mathbf{D}-1} \frac{1}{(1+\delta_{|q_d|,r/2})r} \tag{A.2}$$

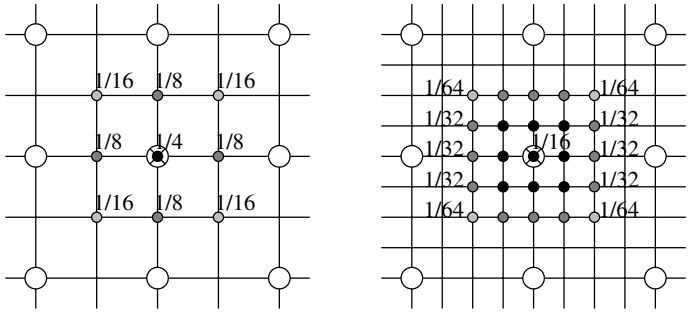

Fig. 16. Averaging down to the center coarse node indicated by ×, in two dimensions. Weights are indicated at the fine nodes, with different shadings for different weights. Left: refinement ratio is 2. Each corner node has weight 1/16, each other node on an exterior face has weight 1/8, and the center node has weight 1/4. Right: refinement ratio is 4. Each corner node has weight 1/64, each other node on an exterior face has weight 1/32, and each remaining node, in the interior, has weight 1/16.

with δ the Kronecker delta. In words, the weights of the nodes come from the tensor product of the weights in the one-dimensional trapezoidal rule.

See Fig. 16 for two-dimensional examples.

A.1.2. Case of non-rectangular domains

It is possible that a non-rectangular domain might not contain some of the points needed in averaging by the method of Section A.1.1. Then the weights will need to be adjusted so that only points in the domain are given nonzero weights.

To simplify the procedure for non-rectangular domains, averaging is performed as a composition of averagings with refinement ratio of 2. In the remainder of this section, we assume that the refinement ratio is 2.

We have a separate weight array for each coarse node. The adjustment we make in (A.2) is to zero out the weight of every fine node that is not reachable from the coarse node by a path lying entirely in the domain, consisting of edges parallel to the axes, without backtracking. The weight of the projected coarse node on the finer level is then increased so that the total of the fine-node weights at the coarse-node is 1. Thus

$$\text{Average}(\varphi^{N,l})_{\boldsymbol{i}} = \sum_{\boldsymbol{q}\in\Theta_2} w_{\boldsymbol{i},\boldsymbol{q}}\varphi^{N,l}_{2\boldsymbol{i}+\boldsymbol{q}}, \tag{A.3}$$

where for $\mathbf{0} \neq \boldsymbol{q} \in \Theta_2$, we define the weight

$$w_{\mathbf{i},\mathbf{q}} = \begin{cases} \prod_{d=0}^{\mathbf{D}-1} \frac{1}{1+|q_d|}, & \text{if } \boldsymbol{q}+2\boldsymbol{i} \text{ is reachable from } 2\boldsymbol{i}, \\ 0, & \text{otherwise} \end{cases} \tag{A.4}$$

and

$$w_{\boldsymbol{i},\mathbf{0}} = 1 - \sum_{\mathbf{0}\neq\boldsymbol{q}\in\Theta_2} w_{\boldsymbol{i},\boldsymbol{q}}.$$

As in Section 3.2, "reachable" here means that there is a path through the fine nodes that avoids backtracking, where each edge in the path is parallel to one of the axes and lies entirely in

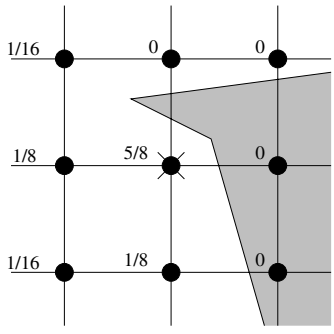

Fig. 17. Weights of fine nodes in averaging to the coarse node indicated by $\times$. The shaded area is outside the domain. Some nodes have weight 0, even though they are in the domain, because there is no path to them from $\times$ following edges that lie entirely in the domain, without backtracking.

the domain. A two-dimensional domain subset is shown in Fig. 17 with the weights on the fine nodes.

A.2. Interpolation

The *Interp* operator is used in multigrid iteration to interpolate the correction from the coarser level to the next finer level. We interpolate from the neighboring coarse nodes, using a formula that is bilinear in two dimensions, trilinear in three dimensions.

In this section, we assume we are interpolating from level $l-1$ to level l, where the refinement ratio is $r = n_{\text{ref}}^{l-1}$.

A.2.1. On rectangular domains

The interpolated value at $\mathbf{i} \in \Omega^{N,l}$ is a linear combination of the values at the $2^{\mathbf{D}}$ coarse nodes in $\Omega^{N,l-1}$ that are the vertices of the box with corners at $\mathscr{C}_r(\boldsymbol{i}) = (\lfloor i_0/r \rfloor, \ldots, \lfloor i_{\mathbf{D}-1}/r \rfloor)$ and $\mathscr{C}_r(\boldsymbol{i}) + \boldsymbol{u}$. The length fractions for each dimension $d = 0, \ldots, \mathbf{D}-1$ are

$$\varepsilon_{\mathbf{i}}^{d}(1) = i_d/r - \lfloor i_d/r \rfloor, \quad \varepsilon_{\mathbf{i}}^{d}(0) = 1 - \varepsilon_{\mathbf{i}}^{d}(1)$$

and these are multiplied to give the weights of the coarse nodes, offset from $\mathscr{C}_r(\mathbf{i})$ by $\mathbf{0} \leqslant \mathbf{q} \leqslant \mathbf{u}$

$$w_{\mathbf{i},\mathbf{q}} = \prod_{d=0}^{\mathbf{D}-1} \varepsilon_{\boldsymbol{i}}^{d}(q_d).$$

Then the interpolation formula is

$$\text{Interp}(\varphi^{N,l-1})_{\mathbf{i}} = \sum_{\mathbf{0} \leqslant \boldsymbol{q} \leqslant \boldsymbol{u}} w_{\mathbf{i},\mathbf{q}} \cdot \varphi^{N,l}_{\mathscr{C}_r(\boldsymbol{i})+\boldsymbol{q}}.$$

See Figs. 18 and 19 for illustrations in two dimensions.

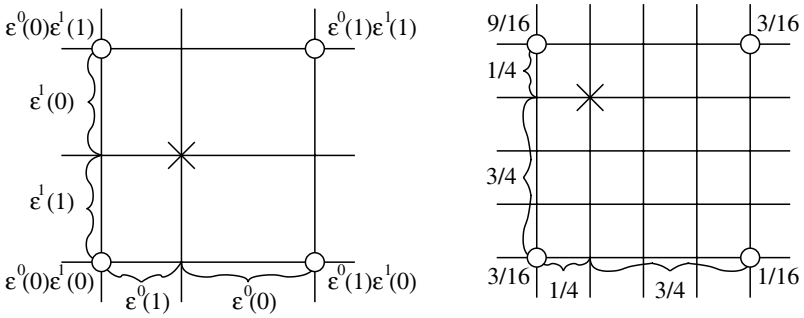

Fig. 18. Left: length fractions and weights of coarse nodes in bilinear interpolation in two dimensions to the fine node indicated by $\times$. Right: an example with refinement ratio of 4.

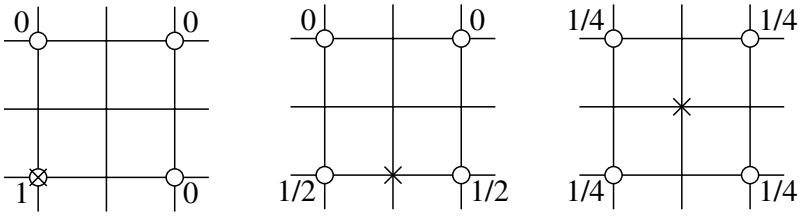

Fig. 19. Interpolation in two dimensions to a fine node indicated by $\times$, when the refinement ratio is 2. Weights are shown on the coarse nodes in three cases.

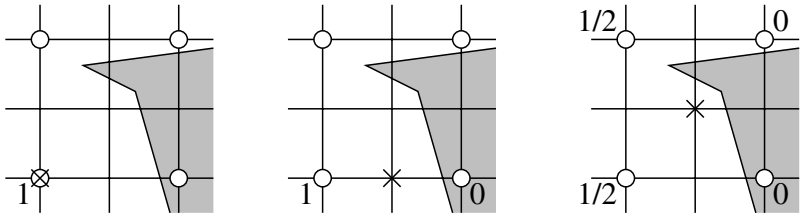

Fig. 20. Interpolation in two dimensions to a fine node indicated by $\times$, when the refinement ratio is 2. The shaded area is outside the domain. For three different fine nodes, weights are shown on the coarse nodes used for interpolation. Compare with Fig. 19.

A.2.2. Case of non-rectangular domains

When the domain is not rectangular, some of the points needed in interpolating by the method of Section A.2.1 may be outside the domain. As we did with averaging in Section A.1.2, we assume that the refinement ratio is a power of 2, and to simplify the procedure, we perform interpolation as a composition of interpolations with refinement ratio of 2. In the remainder of this section, we assume a refinement ratio of 2.

The value at $\boldsymbol{i}$ is interpolated from coarse nodes in the box with corners $\lfloor \boldsymbol{i}/2 \rfloor$ and $\lceil \boldsymbol{i}/2 \rceil = (\lceil i_0/2 \rceil, \ldots, \lceil i_{\mathbf{D}-1}/2 \rceil)$. The number of such coarse nodes is 2 to the power of the number of odd components of $\boldsymbol{i}$. We assign *equal* weight to each of these coarse nodes that is *reachable* from the fine node in the sense described in Section A.1.2: that is, connected by a path through fine nodes along edges parallel to the axes, with no backtracking. See Fig. 20 for weights used in interpolating to some of the fine nodes in a sample two-dimensional domain subset.

References

[1] M.J. Aftosmis, M.J. Berger, G. Adomavicius, A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries, in: 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2000.

[2] Ann Stewart Almgren, A Fast Adaptive Vortex Method using Local Corrections, Ph.D. Thesis, University of California, Berkeley, 1991.

[3] W.L. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, PA, 1987.

[4] P. Colella, Volume-of-fluid methods for partial differential equations, in: E.F. Toro (Ed.), Godunov Methods: Theory and Applications, Kluwer Academic/Plenum Publishers, New York, 2001.

[5] Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, Myungjoo Kang, A second-order-accurate symmetric discretization of the Poisson's equation on irregular domains, J. Comput. Phys. 176 (2002) 205–227.

[6] D.P. Grote, A. Friedman, G.D. Craig, W.M. Sharp, I. Haber, Progress toward source-to-target simulation, Nuc. Instrum. Meth. Phys. Res. A 464 (2001) 563–568.
[7] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comput. Phys. 147 (2) (1998) 60–85.
[8] W.P. Jones, K.R. Menzies, Analysis of the cell-centred finite volume method for the diffusion equation, J. Comput. Phys. 165 (2000) 45–68.
[9] Nami Matsunaga, Tetsuro Yamamoto, Superconvergence of the Shortley–Weller approximation for Dirichlet problems, J. Comput. Appl. Math. 116 (2000) 263–273.
[10] P.A. Seidl, D. Baca, F.M. Bieniosek, C.M. Celata, A. Faltens, L.R. Prost, G. Sabbi, W.L. Waldron, R. Cohen, A. Friedman, S.M. Lund, A.W. Molvik, I. Haber, The high current transport experiment for heavy-ion inertial fusion, in: Proceedings of the 2003 APS/IEEE Particle Accelerator Conference, Portland, Oregon, May 2003.
[11] G.H. Shortley, R. Weller, The numerical solution of Laplace's equation, J. Appl. Phys. 9 (1938) 334–344.
[12] J.-L. Vay, P. Colella, P. McCorquodale, D. Serafini, B. Van Straalen, A. Friedman, D.P. Grote, Progress in the study of mesh refinement for particle-in-cell plasma simulations and its application to heavy ion fusion, in: Seventh International Computational Accelerator Physics Conference, East Lansing, Michigan, October 2002.
[13] J.-L. Vay, P. Colella, P. McCorquodale, B. Van Straalen, A. Friedman, D.P. Grote, Mesh refinement for particle-in-cell plasma simulations: applications to and benefits for heavy ion fusion, Laser Part. Beams 20 (4) (2002) 569–575.
[14] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, J.E. Bussoletti, A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics, J. Comput. Phys. 92 (1) (1991) 1–66.