A cell-centred finite volume method for the Poisson problem on non-graded quadtrees with second order accurate gradients

Christopher Batty[a]

[a] *University of Waterloo, Ontario, Canada*

Abstract

This paper introduces a two-dimensional cell-centred finite volume discretization of the Poisson problem on adaptive Cartesian quadtree grids which exhibits second order accuracy in both the solution and its gradients, and requires no grading condition between adjacent cells. At T-junction configurations, which occur wherever resolution differs between neighbouring cells, use of the standard centred difference gradient stencil requires that ghost values be constructed by interpolation. To properly recover second order accuracy in the resulting numerical gradients, prior work addressing block-structured grids and graded trees has shown that quadratic, rather than linear, interpolation is required; the gradients otherwise exhibit only first order convergence, which limits potential applications such as fluid flow. However, previous schemes fail or lose accuracy in the presence of the more complex T-junction geometries arising in the case of general *non-graded* quadtrees, which place no restrictions on the resolution of neighbouring cells. We therefore propose novel quadratic interpolant constructions for this case that enable second order convergence by relying on stencils *oriented diagonally* and *applied recursively* as needed. The method handles complex tree topologies and large resolution jumps between neighbouring cells, even along the domain boundary, and both Dirichlet and Neumann boundary conditions are supported. Numerical experiments confirm the overall second order accuracy of the method in the L^∞ norm.

1. Introduction

In this work, we present a new quadtree-based cell-centred finite volume technique for solving the variable coefficient Poisson problem, $\nabla \cdot \beta \nabla u = f$, over a Cartesian domain Ω in $\mathbb{R}^2$, with the domain boundary $\partial\Omega$ satisfying either Dirichlet boundary conditions, $u = g_D$, or Neumann boundary conditions, $\frac{\partial u}{\partial n} = g_N$. The solution sought is given by u, f is a source function, and the coefficient β is assumed to be positive, smoothly varying, and bounded below by some $\epsilon > 0$.

The Poisson problem has been studied extensively, and numerical techniques to solve it find wide use across computational physics, applied mathematics, engineering, and elsewhere. Cartesian grid schemes employing finite difference and finite volume discretizations on uniform grids are likewise quite well-studied (e.g., [1]). In many problems the range of scales under consideration may be quite large, which necessitates replacing simple uniform regular grids with *spatially adaptive grids*. This allows computational effort and memory usage to be focused on regions of interest, rather than being wasted on resolving insignificant areas of the domain.

A standard cell-centred uniform grid discretization of the Poisson problem provides second order accuracy of the solution variable *and its gradients*; the primary challenge in moving to the adaptive grid setting is to preserve this same behavior. Maintaining second order accurate gradients is non-trivial because changes in grid resolution introduce so-called T-junction or hanging node configurations at which the standard cell-centred finite difference/volume stencils cannot be directly applied. Nevertheless, accurate gradients can be

Email address: christopher.batty@uwaterloo.ca (Christopher Batty)

of critical importance in various applications. Projection methods for incompressible flow, for example, use the computed pressure gradients to update the fluid's velocity field [2].

The difficulties posed by T-junctions are exacerbated if one wishes to minimize the total number of allocated cells through the use of non-graded (or unrestricted) quadtree grids. Such grids are extremely general in that they place no restriction on the size ratio between neighbouring cells (aside from being a power of two) and allow coarse cells to have multiple neighbour cells of differing sizes along a single face. As noted by Min et al., their added flexibility means they often require far fewer cells [3], with attendant benefits in terms of memory usage and computational cost. Figure 1 shows two examples of a pair of quadtrees before and after applying a 2:1 grading rule between any pair of cells sharing a face. For the extreme case of a randomly generated tree, the graded tree contains nearly twice as many cells (1.84×); for an example of refinement near an interface, the graded tree still has almost 40% more cells (1.36×). In effect, allowing for non-graded trees decouples refinement from discretization: local refinement of the tree can be chosen based solely on application-specific error criteria rather than limitations of the discretization.

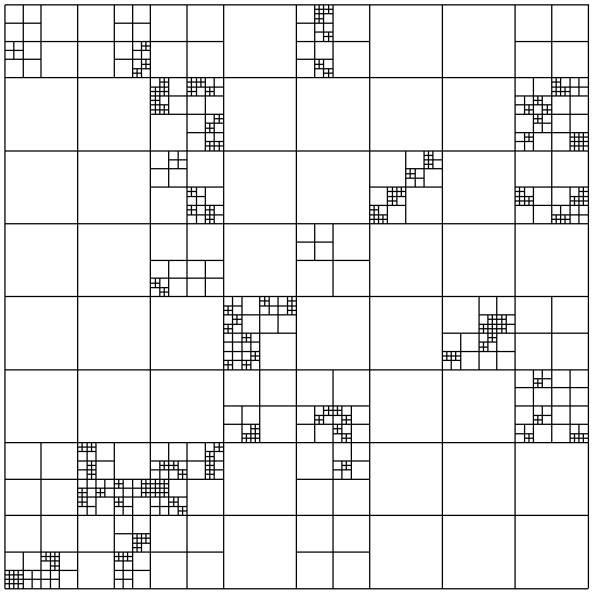
In the current work, we propose a new cell-centred finite volume discretization of the Poisson problem on non-graded quadtree grids. It preserves second order accuracy in both solutions and gradients, handles changes in resolution along domain boundaries, and supports Neumann and Dirichlet boundary conditions.

2. Existing Methods

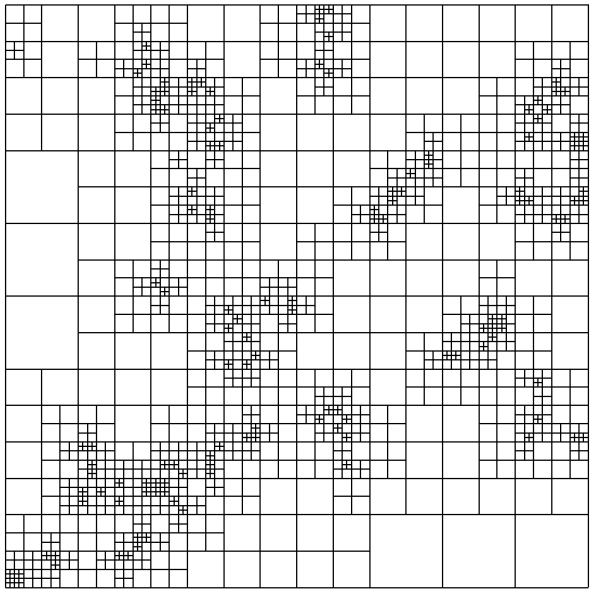
Perhaps the most common family of adaptive cell-centred finite volume schemes makes use of block-structured regular grids, consisting of a set of nested uniform grids laid atop one another. Such a structure limits the flexibility of adaptivity applied in exchange for minimizing the number of stencil cases that arise. While block-structured grids can describe very general configurations, in practice the grid structure is typically further required to be "properly nested" [4]. (Loosely speaking, this means that finer grids must be nested within coarser grids; while this allows for greater than 2:1 refinement ratios between grids, the more general tree-grids considered here are disallowed.) The result is a relatively simple and effective scheme [5, 6, 7, 8] that has, for example, been deployed in the widely used Chombo software package [4]. As will be described in detail later, this strategy relies on two nested quadratic interpolants at each T-junction face in order to recover second order accurate gradients. Recently, Pletzer et al. [9] showed that there do exist situations in which linear interpolation near T-junctions is adequate to preserve second order convergence of *solutions* and offer comparable absolute solution error, although they did not examine the accuracy of gradients. However, as we discuss in Section 3.2, the use of linear rather than quadratic interpolation implies that the gradients will converge at only first order under refinement. Liu [10] proposed a different approach based on linear interpolation, arguing that it provides stronger guarantees of conservation properties in the context of incompressible flow problems.

A more general class of block-structured grid that we do not consider is that of overset or Chimera grids, in which nested grids may overlap and be oriented arbitrarily with respect to their containing grid(s). The different grids must again be coupled in some fashion, either through higher order multivariate axis-aligned interpolation procedures at overlaps (e.g., [11]), or by directly stitching the grids together with extra irregular elements (e.g., [12]). Such methods are commonly used to combine different types of grids or meshes (Cartesian, curvilinear, unstructured) in the discretization of a complex domain, and are well-suited to problems where moderately large continuous regions of constant resolution are desired, such as problems featuring shocks.

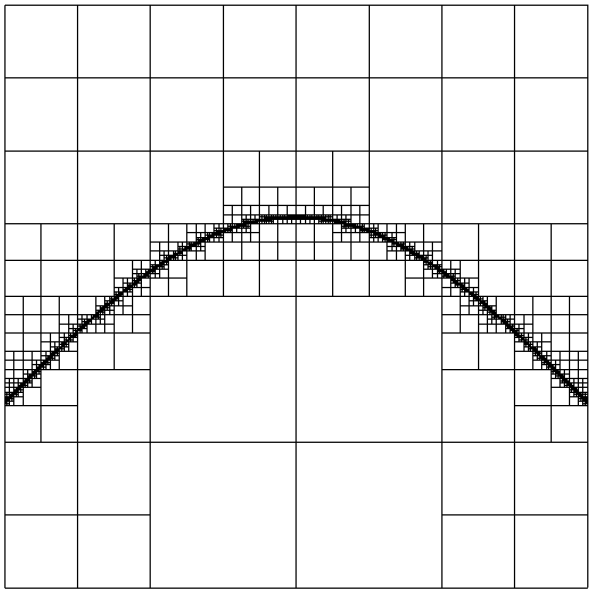
In contrast to approaches using block-structured grids, more recent methods proposed by Popinet [13] and Losasso et al. [14, 15] were among the first to propose the use of general quadtree or octree grids for Poisson problems and fluid flows, which allows for more flexible and targeted adaptivity. (The introduction and adoption of quad- and octree data structures in computer science more generally dates back a few decades earlier [16, 17].) It has been argued that in settings that do not involve shocks, such as Stefan problems or incompressible flows, these tree grids are the optimal choice of data structure [14, 3]. Popinet's tree-based scheme [13] achieves second order accuracy in the solution. However, its use of linear interpolation in some stencil cases near T-junctions drops the accuracy of the computed gradients to first order. Moreover



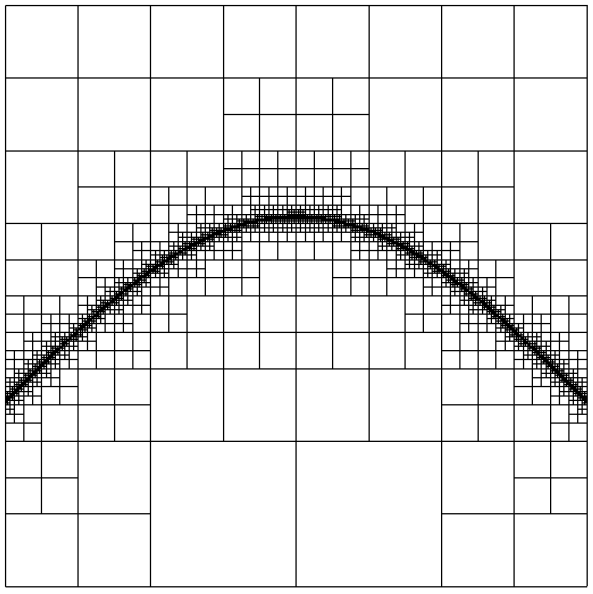
(a) Non-Graded Random



(b) Graded Random



(c) Non-Graded Interface



(d) Graded Interface

Figure 1: Top pair: A randomly generated quadtree (a) (889 cells) and its 2:1 face-graded counterpart (b) (1636 cells). Bottom pair: Non-graded refinement around an interface (c) (2551) cells and its 2:1 face-graded counterpart (d) (3478 cells).

this scheme places a 2:1 grading restriction on the resolution of cells which share *either* a face or a vertex (i.e., an even stronger condition than used in Figure 1).

To achieve more flexible adaptivity, Losasso et al. targeted the most general case of *unrestricted* or *non-graded* trees where neighbouring cells may differ arbitrarily in resolution, leading to far more complex connectivity relationships between neighbours. Their initial approach [14] employed inaccurate gradient estimates in order to preserve symmetry and positive definiteness of the resulting system, with the result that the solution is only first order accurate (its gradients presumably fail to converge). Losasso et al. later proposed an improved scheme for non-graded grids that, perhaps surprisingly, was shown to achieve second order accuracy in the solution value while maintaining symmetry and positive definiteness [15]. A similar symmetric derivation is discussed by Minion [5, 18], albeit in the block-structured grid case, before being discarded in favour of the quadratic interpolation approach discussed above. Recently, Guittet et al. applied Losasso's approach in a non-graded grid pressure projection scheme for the Navier-Stokes equations, due to the stability advantages offered by the staggered approach [19]. The main drawback of Losasso's approach is that while the solution achieves second order accuracy, its gradients are only first order accurate [3].

Min et al. observed that second order accurate gradients had traditionally been achieved at the expense of strong tree grading or nesting restrictions, and therefore sought a non-graded approach [3]. Their scheme makes use of a node-centred finite difference discretization, and is indeed able to recover second order gradients. Their use of finite differences contrasts with the cell-centred finite volume methods used in earlier work and in the current work, and in applications to projection methods in fluid flow [20], their scheme requires further modifications to preserve stability. However, it has the benefit of a fairly localized stencil.

The discussion above captures the dominant set of adaptive grid-based, second order accurate finite difference and finite volume methods for the Poisson problem of which we are aware. Table 1 provides a concise summary of the key properties of the schemes most closely related to the present work. Naturally, there are also finite element (FE) methods that can treat adaptive tree-based grids (e.g., [21, 22, 23]), as well as higher order finite volume (FV) schemes such as the 4th order scheme of Barad and Colella [24]. This FV scheme cannot handle fully non-graded trees, and is therefore not applicable to our immediate problem. While some of the FE schemes cited can handle T-junctions and non-graded trees, it remains a non-trivial task; moreover, basic FE methods also tend to have more difficulty enforcing conservation properties (though naturally there are exceptions, such as certain discontinuous Galerkin schemes). We are in large part motivated by the fact that the second order cell-centred finite volume methodology is well-established and widely used in practice, particularly for fluid flows, yet it cannot currently handle general non-graded trees; we therefore restrict our focus to this setting.

Another possible approach to finding appropriate finite difference stencils is the use of least-squares fitting, as in some meshfree methods (e.g., [25]). This approach is briefly discussed in the context of quadtrees by Min et al. [3] and Olshanskii et al. [26], but not fully pursued. In the case of the very general grids considered here, this would likely also require solving a small dense least squares problem for every possible T-junction configuration to determine the appropriate stencil.

Beyond Poisson and fluid flow problems, quad- and octree grids have to date been applied to a wide variety of additional tasks in computational physics, including level set schemes [27], Stefan and diffusion problems [28, 29], linear elasticity [30], the Poisson-Boltzmann equation [31, 32], and more [33].

3. Discretization

For simplicity of presentation, we write the problem in first order form, in terms of the solution, u, and its gradient, $\mathbf{v}$.

$$\nabla \cdot (\beta \mathbf{v}) = f \tag{1}$$

$$\nabla u = \mathbf{v} \tag{2}$$

Substituting (2) into (1) directly recovers the original system in terms of the solution variable alone, and it is the discretized counterpart of this smaller system that we solve numerically. (In the discrete case, this substitution is a simple application of the Schur complement.) We will now proceed to develop the

Method	Order, u	Order, ∇u	Method	Non-Graded Tree	Sym. Pos. Def.
Martin and Cartwright, 1996 [6]	2	2	FV	No	No
Popinet, 2003 [13]	2	1	FV	No	No
Losasso et al., 2004 [14]	1	0	FV	Yes	Yes
Losasso et al., 2005 [15]	2	1	FV	Yes	Yes
Min et al., 2006 [3]	2	2	FD	Yes	No
Current	2	2	FV	Yes	No

Table 1: A comparison of the properties of finite difference / volume schemes for related adaptive Cartesian grid schemes for the Poisson problem. "Order" indicates convergence order in the L^∞ norm. "Non-graded tree" indicates whether the method requires grading conditions between neighbouring cells. FD indicates a (node-centred) finite difference scheme. FV indicates a (cell-centred) finite volume scheme.

necessary discrete divergence and gradient operators. We temporarily neglect boundary conditions, deferring a discussion of their treatment to Section 4.

3.1. Spatial Discretization and the Discrete Divergence Operator

To address equation 1, we adopt the standard second order accurate finite volume discretization of the divergence operator for a particular cell,

$$\nabla \cdot \mathbf{v} \approx \frac{1}{A_{cell}} \sum_f v_f l_f \tag{3}$$

where f ranges over the set of faces of the given cell, $\mathbf{v}$ is a vector field, $v_f = \mathbf{v} \cdot \mathbf{n}$ is the outward oriented normal component sampled at the midpoint of each face, l_f is the length of the face, and A_{cell} is the area of the cell in question. If a cell's face is shared with multiple adjacent smaller cells due to the presence of T-junctions, we consider it to be composed of two or more smaller sub-faces (rather than as one single large face), each with its own l_f and v_f data.

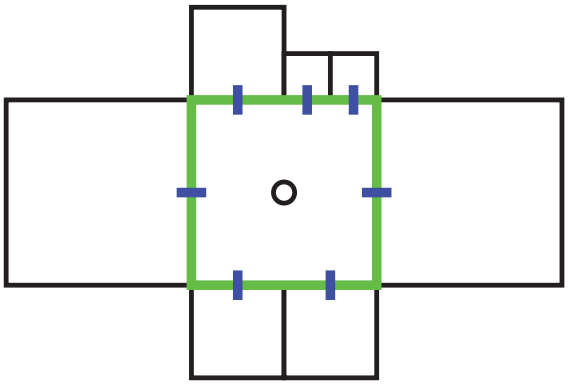


Figure 2: The discrete finite volume divergence stencil for a given cell (green square) involves the fluxes on all faces of the cell (blue dashes). The result is a scalar stored at the cell centre (black circle).

This operator provides a map from a vector field sampled on faces to a scalar divergence field sampled at cell centres. Equation 2 indicates that this vector field will represent the solution gradients. Therefore, in accordance with this picture, we store one gradient normal component $v_f = (\nabla u) \cdot \mathbf{n}$ at the midpoint of each face and one solution value, u, at the centre of each cell, as illustrated in Figure 2.

3.2. The Discrete Gradient Operator

We are left with the task of defining a gradient operator for equation 2 that properly approximates normal components of solution gradients at all cell face midpoints; this operator should provide a map from a scalar-valued field sampled on cell centres to a vector-valued gradient field whose normal component is sampled on cell faces.

We will make use of the standard second order centred finite difference estimate,

$$\frac{\partial u}{\partial x} \approx \frac{u_{forward} - u_{backward}}{h}, \tag{4}$$

where h is the side length of the smaller of the two incident cells. In regular regions of the grid, both $u_{forward}$ and $u_{backward}$ are valid samples that exist at cell centres. In regions where the grid resolution changes, exactly one of the two required data points will not exist (Figure 3), and we therefore need to interpolate from other data points to construct the missing ghost value.

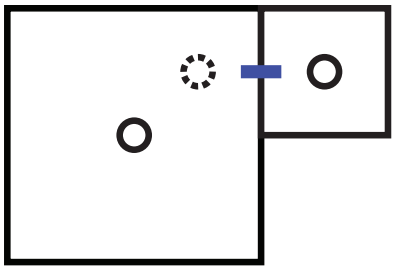


Figure 3: To apply the standard centred difference gradient operator at a face (blue dash) across a resolution jump requires that a ghost value (dashed circle) be constructed by interpolation.

Previous work has consistently demonstrated that quadratic interpolation is the minimum degree necessary to ensure that the computed gradients exhibit second order convergence [5, 18, 6, 7, 34]. For completeness we briefly summarize the argument which Martin and Cartwright provide for this effect. As a second derivative operator, the Laplacian involves division by h^2. Quadratic interpolation has an error of $O(h^3)$, so this division leaves behind an $O(h)$ error on the coarse-fine interface; however, since this interface is one spatial dimension lower than the problem domain (i.e., a set of codimension one), this localized loss of one order of accuracy still allows for the observed $O(h^2)$ global error. Intuitively, this is because the total number of cells increases quadratically under uniform refinement while the number of T-junction cells increases only linearly, the influence of the error in the interface region decreases sufficiently rapidly as to preserve overall second order accuracy. Min et al. [3] provide a comparable argument to support their node-based scheme, and the irregular domain boundary conditions of Johansen and Colella [7] rely on the same effect. Interestingly, in earlier work on the somewhat different problem of composite/overset grids, Chesshire and Henshaw also noted that quadratic interpolation is necessary for second order accuracy [35], and extended their argument to higher order settings. We refer the reader to the various sources above for further analysis and discussion of this effect.

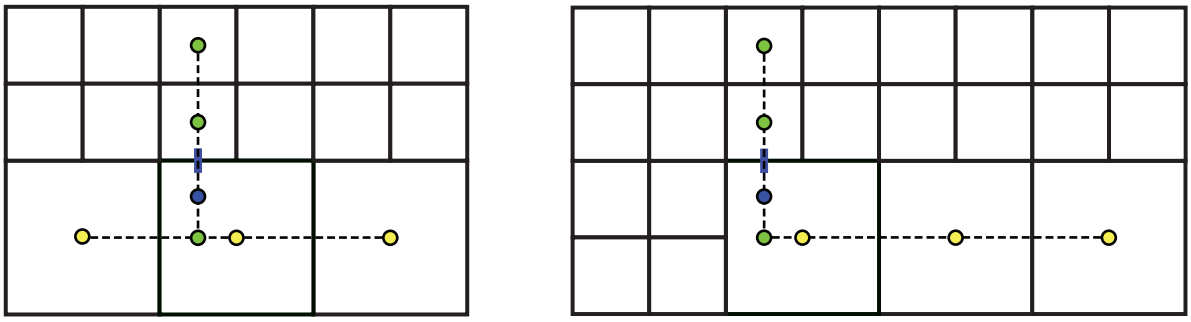
In both previous work and the current work, *the construction of these quadratic interpolants is a critical challenge.* Our principal contribution is therefore a novel strategy for constructing quadratic interpolants that can be straightforwardly applied to arbitrary *non-graded* quadtree grids.

3.3. The Method of Martin and Cartwright

To provide context for the proposed method, the next two sections briefly summarize the interpolant constructions of Martin and Cartwright [6] and Popinet [13], and discuss the restrictions they entail. (For brevity the Martin and Cartwright scheme will be referred to as Martin's method.)

Martin's method is depicted in Figure 4. It constructs the ghost value by first laying down a quadratic interpolant which uses two points on the fine side of a resolution change, and one point on the coarse side. This assumes that the fine side cell has a neighbour along the same axis at the same resolution. On the coarse end of this 3-point stencil there is no readily available data point, so a second perpendicular quadratic interpolant is constructed using three collinear coarse nodes. In the simplest case, the coarse cell and its immediately adjacent neighbours are used (Figure 4a). However, if one of these neighbours is at a different resolution, there is again no coarse sample available. Martin handles this by shifting the stencil over by one coarse cell (Figure 4b); this requires extrapolation rather than interpolation, but Pletzer et al. [9] observe that this induces no loss of accuracy. If three consecutive coarse samples at the same resolution are not available, the authors state that the method uses just the one or two coarse samples that are available, and suffers an attendant loss of accuracy.

For standard block-structured grids where each block usually has dimensions much larger than 3×3, the two cases discussed above are always sufficient up to symmetries; that is, there will always be three collinear coarse cells available. One may observe that under repeated refinement even a non-graded tree will eventually exhibit the form of a block-structured grid so that Martin's cases become sufficient, as illustrated in Figure 5. However, the intent of using non-graded quadtrees is to avoid the extra memory and computational cost associated with block-structured grids, so in typical usage scenarios the tree will be generated and used directly, without further refinement. It is therefore necessary to support multiple resolution jumps occurring close together, for which Martin's method remains inadequate.



(a) Standard case **(b)** Shifted case

Figure 4: Martin's method uses two nested quadratic interpolants (green points, yellow points) to construct a ghost value (blue disk) near faces where the resolution changes (blue dash). If the standard stencil is unavailable, a shifted stencil is used.

3.4. The Method of Popinet

Popinet's scheme assumes a tree structure with a 2:1 grading restriction between all cells sharing either a face or a vertex (i.e., cells neighbouring one another in axis-aligned *or* diagonal directions), which is stronger than the face-only grading condition illustrated in Figure 1. However, as shown in Figure 6 this method also slightly generalizes Martin's using *linear* interpolation to construct missing values when the interpolant stencils on either the fine side (red points) or coarse side (cyan points) encounter finer cells. This choice broadens the set of feasible grids and removes the need to shift the coarse stencil as compared to Martin's method, but the use of linear interpolation locally reduces the accuracy of the resulting gradient estimates to first order. Conveniently, applying standard uniform refinement on a tree-structured grid in order to numerically examine convergence behavior causes most stencil cases that require linear interpolation to quickly disappear, and depending on the problem geometry used to test convergence, this issue might not always be readily apparent. The exceptions are geometries in which a corner of a coarse cell is bordered on two sides by finer cells, as in Figure 7. (This is the case that Martin successfully treats with the shifted

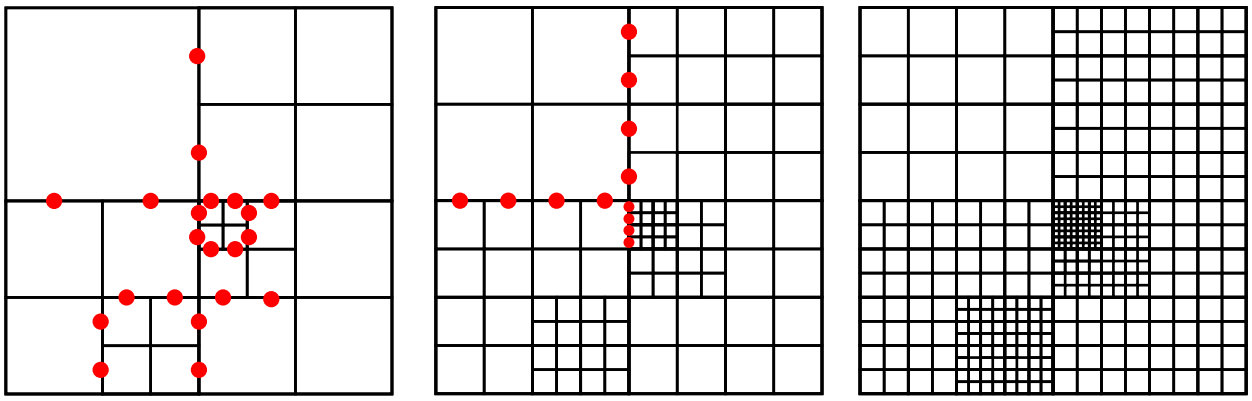


Figure 5: Red disks indicate faces where one of Martin's two standard stencils is unavailable, either due to nearby resolution changes or domain boundaries. Proceeding from left to right, repeated uniform refinement can eventually eliminate all such problematic faces, since the tree ultimately takes on the form of a block-structured grid (right).

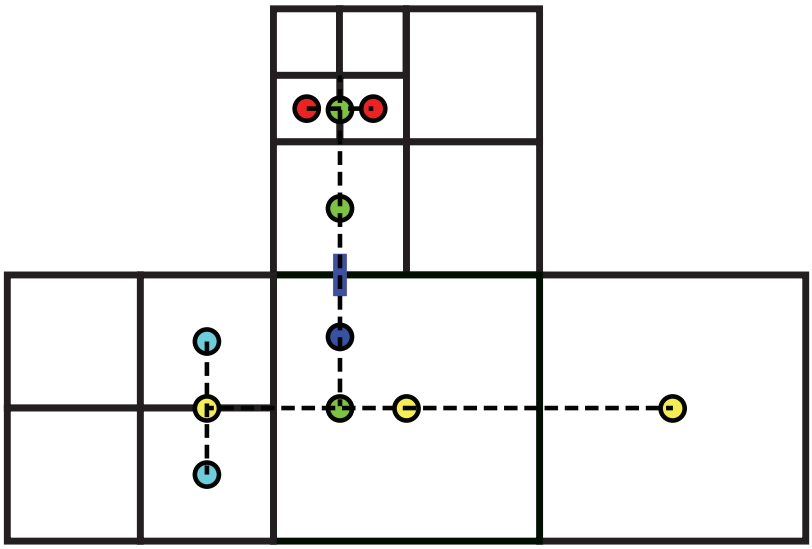


Figure 6: Popinet's scheme: Compared to Martin's method, this scheme allows more flexible adaptivity and additional nearby resolution changes, using *linear* interpolation (e.g., from cyan and red points) to recover the necessary data to complete the quadratic interpolants (yellow and green yellow points). However, a strong grading condition is enforced and gradient accuracy is reduced locally by linear interpolation.

stencil.) In such cases, Popinet's scheme still requires linear interpolation under refinement, and the resulting gradients locally fail to achieve the desired second order accuracy.

3.5. A New Diagonal Interpolation Strategy

The discussion above highlights the fact that comparable cell-centred methods cannot handle general quadtree grids without adding grading restrictions or using lower order interpolation that results in first order gradients. Our method diverges from its predecessors in the manner in which quadratic interpolants are constructed to find the ghost value. The central observation of the method is the following: searching

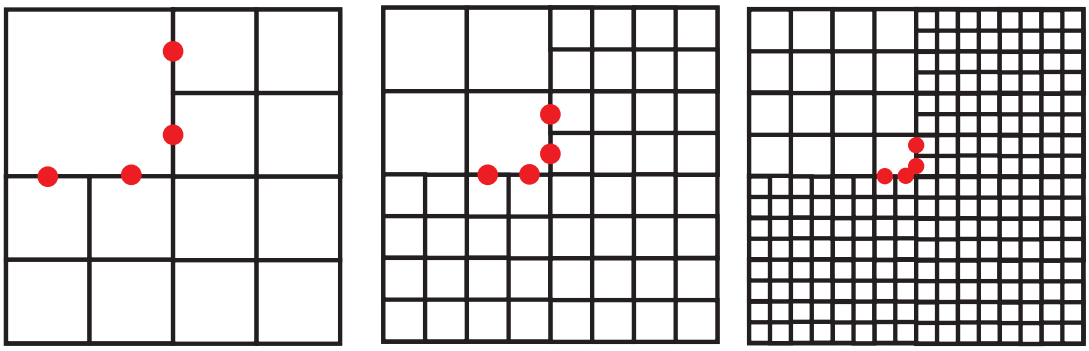


Figure 7: Red disks indicate faces where the stencils dictated by Popinet's scheme rely on linear interpolation, and hence sacrifice accuracy locally. For refined regions with non-convex corners as illustrated , such faces remain under refinement.

in axis-aligned directions for valid neighbouring cell-centred data to use for interpolation typically fares poorly across complex T-junctions, and this issue is greatly exacerbated when no grading restrictions are imposed. By contrast, we will demonstrate that the inherent geometric structure of quadtree grids ensures that searching along a *diagonal* direction typically works quite well, allowing straightforward construction of the necessary interpolants. Diagonal interpolation in general has clearly been used before; an example is the work of Johansen and Colella [7] who make use of sloped quadratic interpolants to support boundary conditions on irregular domains. However, our use of diagonal interpolants to treat coarse-fine grid interfaces is novel to the best of our knowledge.

3.6. Diagonal Interpolation with Strong Grading

To simplify presentation, we will begin by describing our discretization under the highly restricted grading required by Popinet's scheme, and proceed to gradually remove grading conditions until we arrive at the fully non-graded case. Consider a quadtree that is graded by requiring a 2:1 ratio for cells sharing either a face or a vertex, as in Popinet's method. In this case, the ghost point always lies on the nearest of the two main diagonals of its containing cell. We can therefore always use one diagonal quadratic interpolant to approximate the ghost value, as illustrated in Figure 8. Because of the grading conditions and the tree's recursive structure, the centres of the diagonal neighbours will always be perfectly aligned to support the interpolant, regardless of their resolution. This scheme already has two advantages over that of Popinet: it avoids linear interpolation so that second order accurate gradients are assured in all cases, and it requires just a single quadratic interpolation at each T-junction face rather than a nested pair.

3.7. Diagonal Interpolation with Weaker Grading

If we now drop the 2:1 restriction between diagonal neighbours (cells sharing only a vertex), but preserve 2:1 grading across faces, we can have at most a 4:1 ratio in cell size between diagonal neighbours, as in Figure 9. This implies that we may encounter even finer or coarser cells on the path of our diagonal interpolant than before. If the cell is finer or the same resolution, we use its cell-centred value directly as before. If the cell is coarser, the appropriate sample of the solution is not readily available; that is, the centre of the diagonal neighbour cell is not on the diagonal line used to define our interpolant as Figure 9 depicts. Instead, we find the intersection of our diagonal line with the larger cell's opposing diagonal, and deploy a second quadratic interpolant along *that* diagonal to construct another ghost point. In this fashion, second order accuracy in solution and gradients is preserved.

This recursion can continue as needed, but does so only if the current interpolant requires data from additional *even coarser* neighbour cells. When an interpolant encounters cells that are finer or of the same size, the tree's geometric structure guarantees that the interpolant's diagonal line intersects the cells' centres

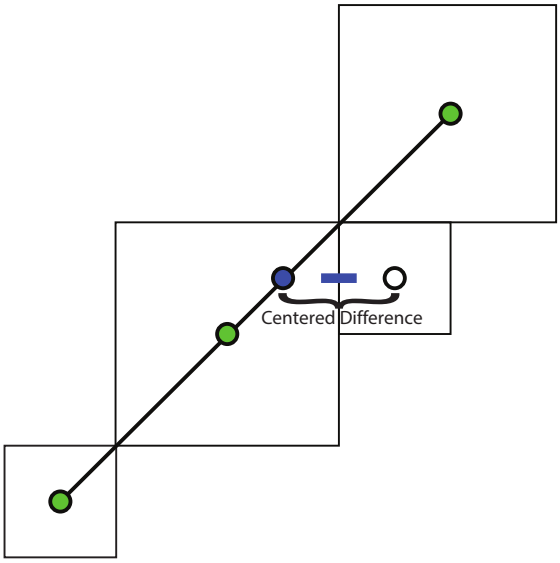


Figure 8: Strong Grading: Our gradient stencil in the strong grading case requires a single diagonal quadratic interpolation. The desired interpolated ghost sample (blue circle) is quadratically interpolated from the three cell-centred samples (green) found along the diagonal.

directly, so their values are always available; no further interpolation is required. This property ensures that recursion propagates only *up* the tree to coarser levels without spawning new interpolants at finer resolutions down the tree. Therefore even in the most pathological cases, the recursion will still terminate relatively quickly, either when the coarsest level of the tree is reached or when an interpolant encounters a domain boundary (discussed in Section 4).

3.8. Diagonal Interpolation with No Grading

We can now take the final step of removing *all* grading conditions to allow arbitrary resolution jumps. The only added challenge this introduces is that the initial ghost point to be interpolated might not lie on the main diagonal of its containing cell as it does in the preceding graded cases; this new case is shown in Figure 10.

Fortunately, as we show in Figure 11, this too can be straightforwardly compensated for: starting from the missing ghost point we draw a line at a 45 degree angle that intersects the diagonal of the ghost point's containing cell, and continue it on either end until it encounters either the diagonal of another cell or an existing cell-centred sample. We use these three points to construct an initial quadratic interpolation. At each diagonal intersection, we again use a perpendicular diagonal quadratic interpolation stencil in the manner described previously, recursing as needed. All subsequent interpolations make use only of points that lie on (main) cell diagonals as in the earlier cases we considered, so no new geometric situations arise.

There is some flexibility in choosing whether to orient the initial diagonal quadratic interpolant at 45 degrees up or down, relative to the grid axis, since either would suffice. We choose the axis with the shortest distance to a diagonal of the ghost point's containing cell; i.e., we cast the ray up if the ghost is above the midpoint of the large face, or down otherwise (with the analogous choices for faces in the other axis).

These recursive interpolation rules are relatively straightforward to implement on arbitrarily refined quadtree data structures provided they support appropriate neighbour traversal operations. High-level pseudocode for the recursive process that constructs the gradient stencil for all the grid faces is provided in Algorithms 1, 2, and 3. (For brevity and simplicity of presentation, low-level implementation details related

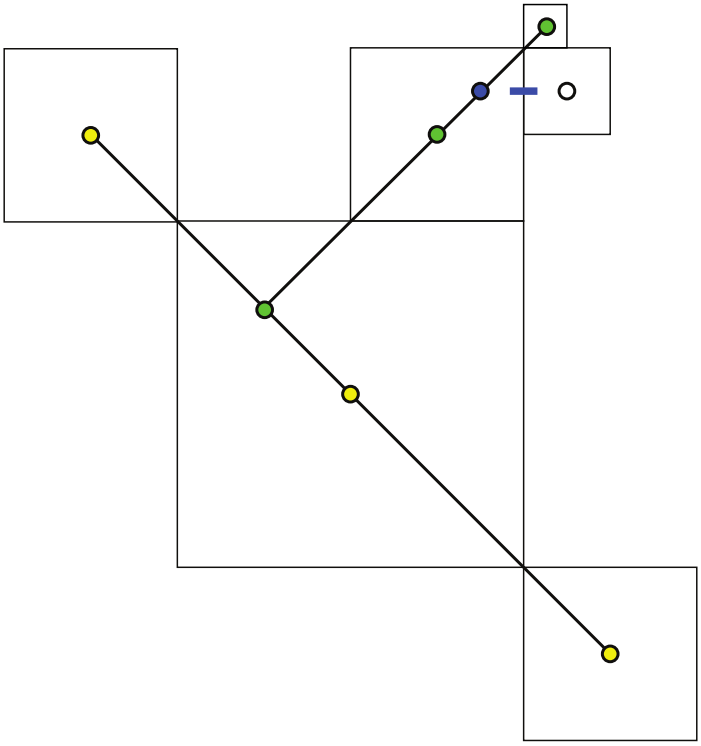


Figure 9: Weaker Grading: If the grading condition is relaxed so that only cells sharing a face must satisfy a 2:1 ratio, an appropriate ghost value can still be found using *recursive* quadratic interpolation along perpendicular diagonals.

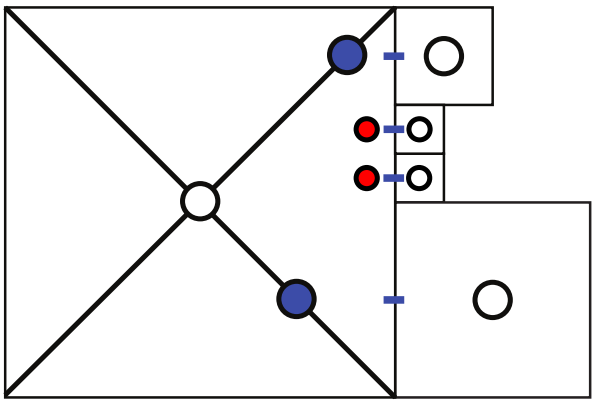


Figure 10: In the non-graded case, the missing ghost points (coloured disks) to be interpolated might not lie on a diagonal of their containing cell (red disks), although the top and bottom ones still do (blue disks).

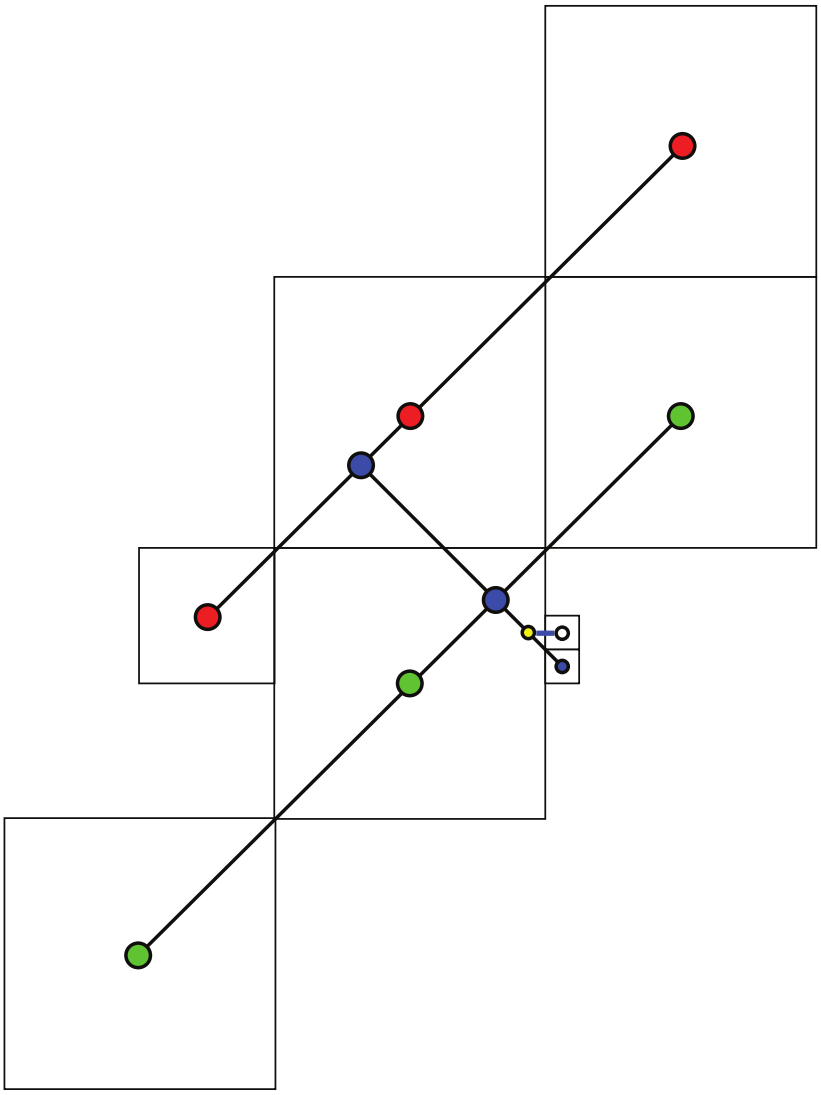


Figure 11: No Grading: If the missing ghost point (yellow) does not lie on the main diagonal of its containing cell, an extra initial diagonal interpolant (blue disks) is used. The initial search direction is aimed towards the nearest main diagonal of the cell containing the ghost point (though either main diagonal would suffice). All subsequent interpolants (green disks, red disks) will, by construction, lie on the main diagonal of the cell containing the current point being interpolated.

to tree data structures and boundary condition enforcement have been omitted.) We confirm in Section 5 that our discretization yields numerical results consistent with second order accuracy in both solutions and gradients, but first we must consider how to extend this approach to treat domain boundaries.

Algorithm 1 Gradient Stencil Construction

```
1: procedure BuildGradStencil(Quadtree Q)
2:     for all faces f in Q do
3:         if f is not on a T-junction then
4:             form centred difference stencil with incident cell-centred values
5:         else
6:             Ghost G = ghost point in larger incident cell
7:             if G is on one of its containing cell's diagonals then
8:                 Axis A = vector along cell diagonal that G lies on
9:                 Weights W = InterpolateDiagonal(Q, G, A)
10:            else
11:                Axis A = 45 degree vector towards closest diagonal of G's containing cell
12:                Weights W = InterpolateOffDiagonal(Q, G, A)
13:            end if
14:            form centred difference stencil, using interpolation weights W for G
15:        end if
16:    end for
17: end procedure
```

4. Boundary Conditions

We assume either Neumann or Dirichlet boundary conditions. A particularly simple way to enforce boundary conditions would be to require a few layers of uniform resolution cells along the entire domain boundary. This reduces boundary condition enforcement to the standard regular grid setting which is well-known to achieve the desired level of accuracy. However, similar to the 2:1 grading condition, completely uniform refinement of the whole boundary can often require creating many more cells.

If we instead allow resolution changes along the domain boundary, there are two ways in which boundary conditions come into play. First, any cell that is immediately incident on the boundary clearly needs to have Dirichlet or Neumann boundary conditions applied. The second and more subtle case is when one of the diagonal lines defining a quadratic interpolant on the interior intersects the boundary; the construction of the quadratic interpolants must be modified. We describe our treatment of these cases below.

4.1. Dirichlet Conditions on Boundary Cells

For cells that are incident to the boundary of the domain, Dirichlet conditions can be straightforwardly applied using an extra regularly spaced ghost sample on the outside of the domain. Its value is constructed through axis-aligned quadratic extrapolation (Figure 12), and this ghost sample is used in setting up the standard stencil for the gradient on the boundary face. (This boundary treatment is a special case of the classic Shortley-Weller discretization [36] which achieves second order accuracy for the more general case of irregular domains. Another alternative boundary treatment would use only linear extrapolation as done by Gibou et al. for irregular domains, but this again drops the gradient accuracy to first order [37, 38].)

The difficulty that quadratic extrapolation at the boundary introduces is that the boundary cell must have an adjacent interior neighbour *at the same resolution*, which is not always the case for general quadtrees (Figure 13, left). We can handle the need for a second interior interpolation point in one of two ways. One option is to extrapolate using a *diagonal* quadratic interpolant of the type used elsewhere in our scheme, rather than the more common axis-aligned extrapolation (Figure 13, centre). (If the line defining the interpolant encounters additional interior coarse regions, this may require recursive interpolation as before.)

Algorithm 2 Raycast to Interpolate Ghost Point on a Cell's Diagonal

procedure InterpolateDiagonal(Quadtree Q, Ghost G, Axis A)
    Find diagonally adjacent neighbour cells, Nbr_1 and Nbr_2, along axis A
    Cur = containing cell of G
    Axis B = rotation of A by 90 degrees
    G_C = centre of cell Cur
    if Size(Nbr_1) > Size(Cur) **then**
        G_1 = ghost point on diagonal of Nbr_1
        Weights W_1 = InterpolateDiagonal(Q, G_1, B)
    else
        G_1 = centre of cell Nbr_1
    end if
    if Size(Nbr_2) > Size(Cur) **then**
        G_2 = ghost point on diagonal of Nbr_2
        Weights W_2 = InterpolateDiagonal(Q, G_2, B)
    else
        G_2 = centre of cell Nbr_2
    end if
    Weights W = quadratic interpolation at G from points G_1, G_2, G_C, and weights W_1, W_2.
    return W
end procedure

Algorithm 3 Raycast to Interpolate Ghost Point *Not* on a Cell's Diagonal

procedure InterpolateOffDiagonal(Quadtree Q, Ghost G, Axis A)
    Cur = containing cell of G
    Determine point G_D on diagonal of Cur along A.
    Find adjacent neighbour cells, Nbr_1 and Nbr_2, along axis A
    Axis B = rotation of A by 90 degrees
    Weights W_D = InterpolateDiagonal(Q, G_D, B)
    if Axis A does not intersect the centre of Nbr_1 **then**
        G_1 = ghost point on diagonal of Nbr_1
        Weights W_1 = InterpolateDiagonal(Q, G_1, B)
    else
        G_1 = centre of cell Nbr_1
    end if
    if Axis A does not intersect the centre of Nbr_2 **then**
        G_2 = ghost point on diagonal of Nbr_2
        Weights W_2 = InterpolateDiagonal(Q, G_2, B)
    else
        G_2 = centre of cell Nbr_2
    end if
    Weights W = quadratic interpolation at G from points G_1, G_2, G_D, and weights W_1, W_2, W_D.
    return W
end procedure

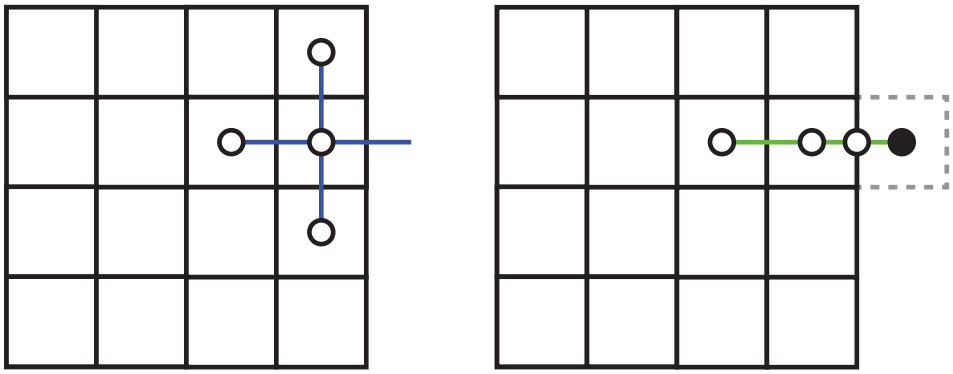


Figure 12: Left: At boundary cells, standard centred difference stencils cannot be applied because there is no appropriate neighbour degree of freedom. Right: The usual solution is to compute data for the missing ghost cell (black disk) through quadratic extrapolation based on two interior cells and the Dirichlet boundary value (empty circles).

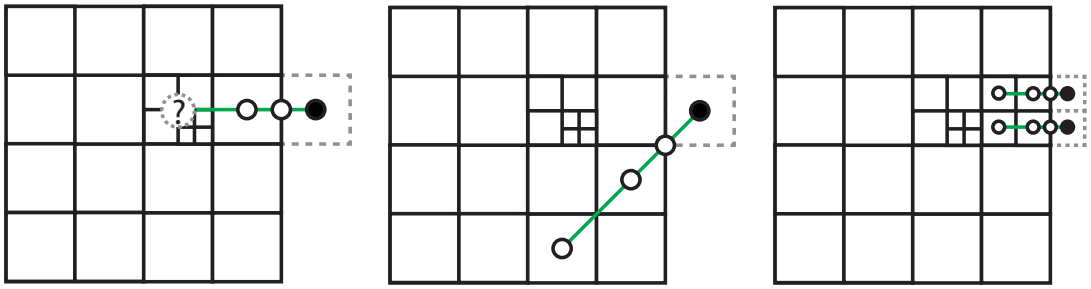


Figure 13: Left: Boundary cells lacking an interior neighbour at the same resolution make standard axis-aligned quadratic extrapolation difficult to apply. Center: One possible solution is diagonal extrapolation. Right: We apply a simpler solution: apply one refinement step on boundary cells that lack the necessary interior neighbour.

A second option is to simply perform a single refinement step on any boundary cell that lacks the required interior neighbour; the resulting child boundary cells will always have an inside neighbour cell at the same resolution (Figure 13, right). Note that this latter choice is different than the wasteful requirement of enforcing uniform resolution *along* the entire boundary mentioned previously; instead, this simply ensures that each cell has one matching resolution neighbour in the direction *normal to* the boundary. A high degree of adaptation parallel to the boundary is still preserved. We prefer this option due to the simplicity of its implementation, and we emphasize two of its features.

First, boundary cells can be checked and subdivided independently, and since our method does not require 2:1 grading, this boundary subdivision *does not* cause a cascade of refinements deeper into the interior. The one case where the refinement cannot be done truly independently is at corner cells: refining a cell adjacent to a corner cell may cause that corner cell to fail to satisfy the required condition in the perpendicular direction. Corner cells must therefore undergo the same check a second time (see Figure 14).

A second point to note is that the number of offending boundary cells is typically small so that relatively few divisions are actually needed in practice. For example, the randomly generated quadtree grid of Figure 19 features a substantial degree of adaptivity along boundaries, but requires subdividing only about one quarter (17/62) of the boundary cells; this in turn is a small fraction of the 838 cells in the entire original

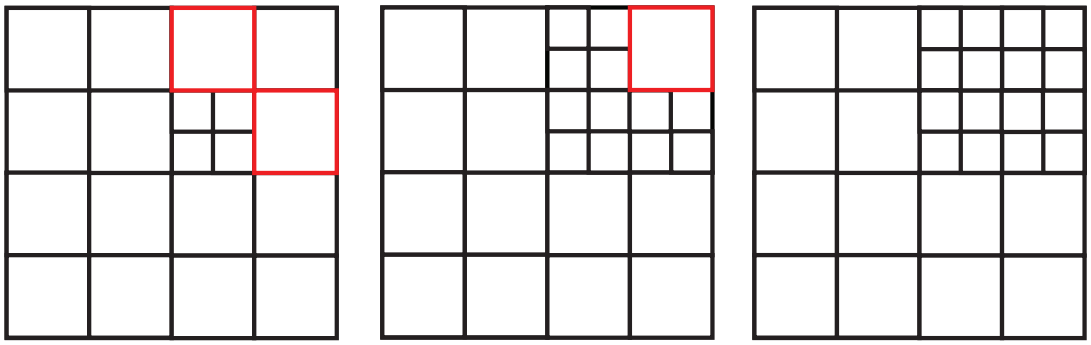


Figure 14: Left: The initial grid may have multiple boundary cells that do not have interior neighbours at the same resolution (red squares). Centre: Subdividing only the offending boundary cells *independently* corrects this in nearly all cases, but may introduce new violations at corners (red square). Right: A second pass on corner cells alone ensures all boundary cells satisfy the constraint.

grid.

4.2. Dirichlet Conditions on Interpolants at Boundaries

As noted previously, when the boundary is non-uniformly refined, quadratic interpolants used in the interior may end up intersecting the boundary. In the Dirichlet case, the value on the boundary, g_D, is known and can be used directly in constructing the interpolant, rather than using the value at a cell centre. See Figure 15, left.

4.3. Neumann Conditions on Boundary Cells

Neumann boundary conditions can be applied for cells incident on the boundary in the standard fashion with no modification whatsoever. We simply set the boundary flux to satisfy the specified gradient value, g_N.

4.4. Neumann Conditions on Interpolants at Boundaries

The more challenging task is to handle the case when a quadratic interpolant on the interior intersects a boundary with Neumann conditions. Unlike the Dirichlet case, we cannot simply use a boundary value directly, since we do not possess it. Furthermore, although the gradient value is given by the problem, it is the gradient *normal* to the boundary, whereas the line defining our interpolant meets the boundary at an angle.

Our solution is to extend the diagonal line of the interpolant beyond the boundary until it crosses the centre axis of the next interior cell (Figure 15, right). We will determine a ghost value at this position. We do so by constructing an axis-aligned quadratic interpolant using two interior cell centre values and the specified normal gradient at the boundary given by the Neumann condition, $\partial u/\partial n = g_N$. This additional quadratic allows us to extrapolate a ghost value which is in turn used to construct the originally required quadratic interpolant. The use of axis-aligned interpolation does require that each boundary cell have an interior neighbour of the same size, but this is easily enforced: it is precisely the same boundary refinement condition we introduced earlier to handle Dirichlet conditions.

Furthermore, our rule for refining boundary cells ensures that our quadratic interpolants never exit the domain boundary out of a convex corner cell, so there will always be an appropriate axis-aligned pair of interior cells from which to interpolate the necessary ghost value. For non-convex (re-entrant) corner geometries, the same guarantee can be provided by requiring that a two-layer patch of cells immediately surrounding the corner be uniformly refined; this can be straightforwardly enforced with a few additional refinement operations. An example of this scenario is provided in Section 5.5, in Figure 18.

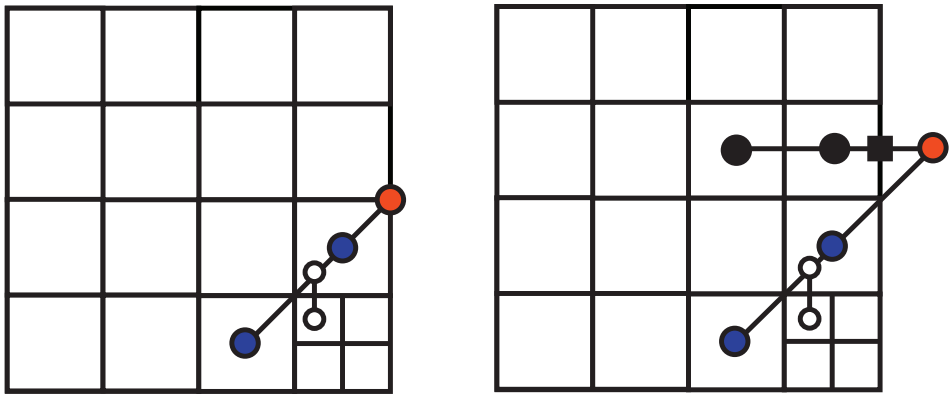


Figure 15: Left: When a required quadratic extrapolant intersects the boundary with Dirichlet conditions applied, we use the given boundary value (red disk) rather than a cell centre value. Right: For Neumann boundary conditions, we extend the extrapolant beyond the boundary until it intersects the axis of the adjacent boundary cell. We then construct a ghost cell value (red disk) using an axis-aligned quadratic extrapolant constructed from two interior cell centre values (black disks) and the gradient given by the Neumann boundary value (black square).

4.5. "Pure" Neumann boundary problems

In the case where the domain is entirely enclosed by Neumann boundaries, the solution to the discrete equations is not unique; there is a rank-one null space such that solutions which differ by a constant still satisfy the original equations. As noted by Min et al. [3], practical scenarios of this type usually indicate that only the gradient is of interest, such as for certain incompressible flows.

Min et al. found that treating the singular system by setting a single point in the domain to a fixed value as a Dirichlet condition led to inaccurate numerical gradients in a region around the chosen point [3]. We initially observed the same effect. However, we found that, at least in our setting, this is a numerical artifact caused by a failure of the discrete problem to satisfy the necessary discrete compatibility condition (see e.g., [39]). This can be corrected by first projecting out the null space component of the right-hand-side f. (Rescaling rows of the system by the areas of their associated cells ensures that the null space is the usual constant vector, $\mathbf{1}$.) This conveniently avoids the need for the two-pass approach suggested by Min et al.

4.6. A Hybrid Variant

As illustrated earlier in Figure 5, under repeated uniform refinement the more complex quadtree cases gradually cease to occur so that Martin's stencils are eventually the only ones needed. This suggests another possible approach, which is to view our new diagonal stencils as an upgrade to Martin's scheme. One can use Martin's axis-aligned interpolants wherever possible and introduce diagonal interpolants only in more complex configurations where Martin's axis-aligned interpolants are unavailable. We experimented with this approach and found that it typically gives comparable results in terms of absolute error and convergence behavior; our results include one such example.

4.7. Properties of the Linear System

Non-adaptive, regular grid finite volume methods for solving the Poisson problem lead to discrete divergence and gradient operators that possess a discrete orthogonality relationship; i.e., their matrix representations are (negative) transposes of one another. This property in turn leads to symmetric positive definite linear systems. Unfortunately, as is the case for previous methods that make use of quadratic interpolation, the orthogonality relationship between the discrete operators is lost and therefore the linear systems

resulting from our adaptive scheme are neither symmetric positive definite, nor even necessarily diagonally dominant. This rules out various common iterative solvers, direct solvers, and preconditioners which rely on these features, such as conjugate gradient, MINRES, direct Cholesky factorization, incomplete Cholesky preconditioning, etc. Nevertheless, many effective solvers exist to deal with non-symmetric systems: direct solvers based on LU factorization, Krylov solvers such as BiCGStab and GMRES, multigrid variants, etc. As discussed below, our numerical experiments make use of an LU-based direct solver in a black box fashion, although it would be an interesting future direction to explore the development of an effective multigrid scheme tailored to our specific discretization.

5. Numerical Results

We illustrate the effectiveness of our method by applying it to a selection of analytical test cases; these are drawn heavily from prior work by Min et al. [3] with a few additions of our own. To evaluate convergence, we compare the analytical solution to our numerical solutions achieved on a sequence of increasingly refined quadtrees generated by uniformly subdividing each cell into its four children. The initial quadtree pattern is prescribed in each problem statement (although in a practical application setting the refinement pattern would instead be dictated by a problem-dependent error criteria or refinement strategy). The uniform refinement used here preserves the resolution ratios across T-junctions, and is the typical approach used when evaluating order of convergence. In all cases, the quadtree shown in the figures is that *before* applying our extra boundary refinement step to support boundary conditions, unless otherwise noted. We observe results consistent with second order convergence of both solutions and gradients.

We implemented our method in C/C++. To solve the non-symmetric linear systems generated by our discretization, we used the Eigen library's "SparseLU" method [40], a direct solver based on a supernodal LU factorization technique adapted from the SuperLU package [41].

5.1. Poisson's equation with Dirichlet boundary conditions

We solve $\Delta u = f$ on the domain $\Omega = [0, \pi]^2$ with an exact solution of $u(x, y) = e^{-x-y}$ using Dirichlet boundary conditions (Min's example 9.1.1). Figure 16 shows the initial quadtree grid, and Table 2 lists the observed errors and convergence rates for the solution and its gradients. The column "Effective Resolution" indicates the resolution if the domain were a uniform grid of the same size as the smallest refined cell.

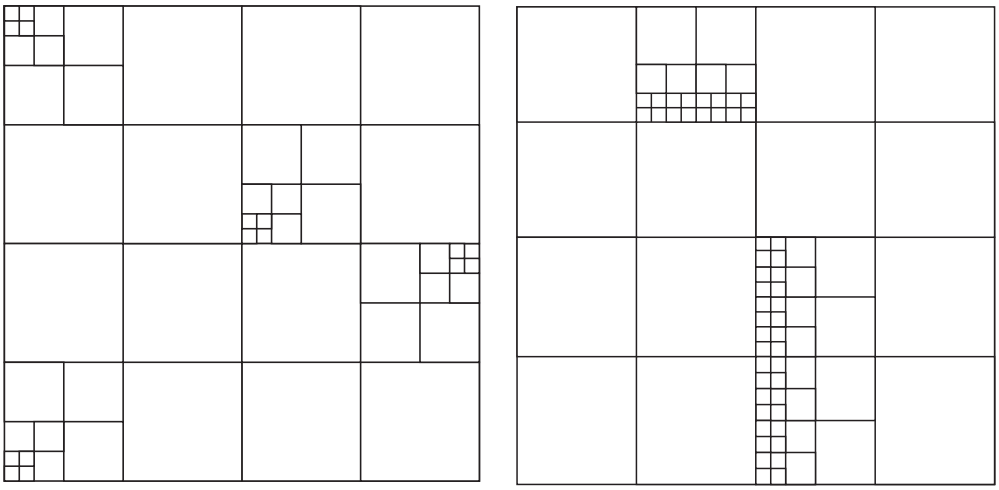


Figure 16: Left: The initial quadtree for the tests of Sections 5.1, 5.3, 5.6, 5.7, and 5.8. Right: The initial quadtree for the test of Section 5.2.

Table 2: Error analysis for Poisson with Dirichlet conditions (Section 5.1)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	2.74×10^{-3}	-	1.92×10^{-2}	-
64^2	1.10×10^{-3}	1.32	7.82×10^{-3}	1.30
128^2	2.99×10^{-4}	1.88	2.50×10^{-3}	1.65
256^2	7.52×10^{-5}	1.99	7.17×10^{-4}	1.80
512^2	1.95×10^{-5}	1.94	1.91×10^{-4}	1.91
1024^2	5.16×10^{-6}	1.92	4.92×10^{-5}	1.96
2048^2	1.33×10^{-6}	1.96	1.24×10^{-5}	1.98
4096^2	3.36×10^{-7}	1.98	3.13×10^{-6}	1.99

Table 3: Error analysis for Poisson with Neumann conditions (Section 5.2)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	3.47×10^{-2}	-	5.47×10^{-1}	-
64^2	9.20×10^{-3}	1.92	2.27×10^{-1}	1.27
128^2	2.80×10^{-3}	1.71	5.89×10^{-2}	1.94
256^2	7.55×10^{-4}	1.89	1.48×10^{-2}	1.99
512^2	1.94×10^{-4}	1.96	3.71×10^{-3}	2.00
1024^2	4.92×10^{-5}	1.98	9.29×10^{-4}	2.00
2048^2	1.24×10^{-5}	1.99	2.32×10^{-4}	2.00
4096^2	3.10×10^{-6}	2.00	5.81×10^{-5}	2.00

5.2. Poisson's equation with homogeneous Neumann boundary conditions

We solve $\Delta u = f$ on the domain $\Omega = [0, \pi]^2$ with an exact solution of $u(x, y) = \cos(x)\cos(y) - 1$ using Neumann boundary conditions (Min's example 9.1.5). Figure 16 shows the initial quadtree grid, and Table 3 gives the numerical results.

5.3. Poisson's equation with nonhomogeneous Neumann boundary conditions

We observed that the above scenario tests only homogeneous Neumann boundary conditions, $\partial u/\partial n = 0$. We therefore also performed the same test as in Section 5.1 with Neumann boundary conditions instead of Dirichlet, which involves non-zero boundary gradients. We use the same initial quadtree as before, shown in Figure 16, and the results are listed in Table 4.

Table 4: Error analysis for Poisson with nonhomogeneous Neumann conditions (Section 5.3)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	7.79×10^{-2}	-	1.75×10^{-1}	-
64^2	3.94×10^{-3}	4.31	8.48×10^{-3}	4.37
128^2	6.56×10^{-4}	2.58	2.40×10^{-3}	1.82
256^2	1.27×10^{-4}	2.37	6.78×10^{-4}	1.82
512^2	3.11×10^{-5}	2.03	1.88×10^{-4}	1.85
1024^2	7.70×10^{-6}	2.02	5.06×10^{-5}	1.89
2048^2	1.91×10^{-6}	2.01	1.34×10^{-5}	1.92
4096^2	4.77×10^{-7}	2.00	3.49×10^{-6}	1.94

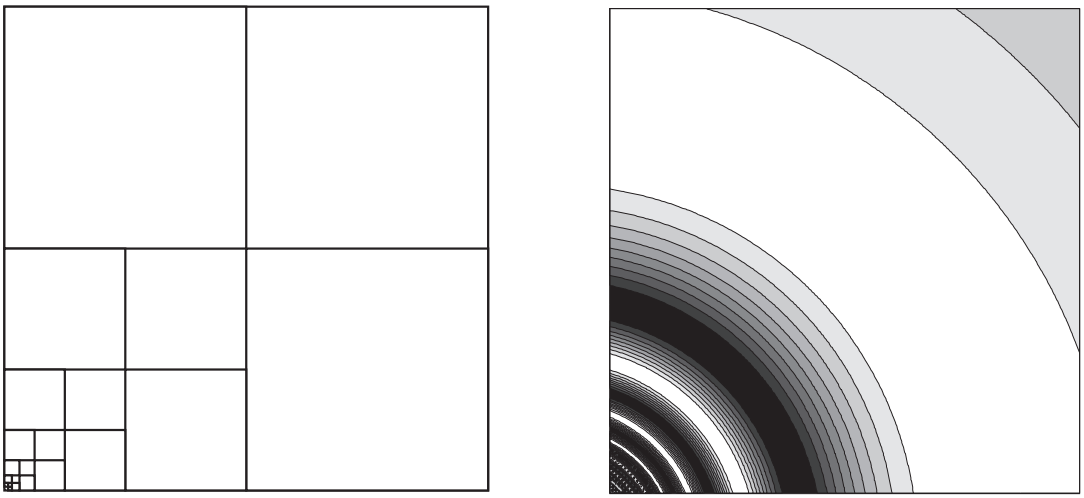


Figure 17: Left: The initial quadtree for the test of Section 5.4. Right: A contour plot of the exact solution, illustrating the sharp gradients present in the lower left corner.

5.4. Poisson's equation: Adaptive vs. Uniform Grids

To verify that adaptive grids confer an advantage over uniformly refined regular grids, we solve $\Delta u = f$ on the domain $\Omega = [0.1, 1]^2$ with an exact solution of $u(r, \theta) = \sin(1/r)$ using Dirichlet boundary conditions (Min's example 9.1.3). This problem has sharp gradients in the lower left corner of the domain, so our initial quadtree grid is refined in that corner as shown in Figure 17. Convergence is again examined by uniformly subdividing all cells, starting from this initial pattern. The adaptive grid results are shown in Table 5 and the uniform grid results are shown in Table 6. Under refinement both approaches ultimately converge at approximately second order, however in the adaptive case the absolute error is much smaller for comparable numbers of elements. For example, a gradient error of approximately 1.5×10^{-3} is achieved for the adaptive and uniform cases with about 120K and 1.4M cells, respectively, i.e., a full order of magnitude difference in number of cells.

5.5. Poisson's equation on a non-rectangular domain

To verify that our method is effective even in domains that are not rectangular, we solve $\Delta u = f$ with exact solution $u(x, y) = \sin(x)\sin(y)$ on an L-shaped domain (Min's example 9.1.4), shown in Figure 18, left. Min et al. appear not to have given the coordinates or dimensions of this domain, so we assume the lower-left corner to be the origin, $(0, 0)$, and the width of the domain to be π, in keeping with several other examples. The results are given in Table 7.

Similarly, the results for the same test performed with Neumann boundary conditions are given in Table 8. As discussed in section 4.4, we locally refine to a uniform resolution around the re-entrant corner, to ease the handling of Neumann boundary conditions (Figure 18, right). This does not preclude substantial adaptivity elsewhere on the domain boundary.

5.6. Variable coefficient Poisson equation with Dirichlet boundary conditions

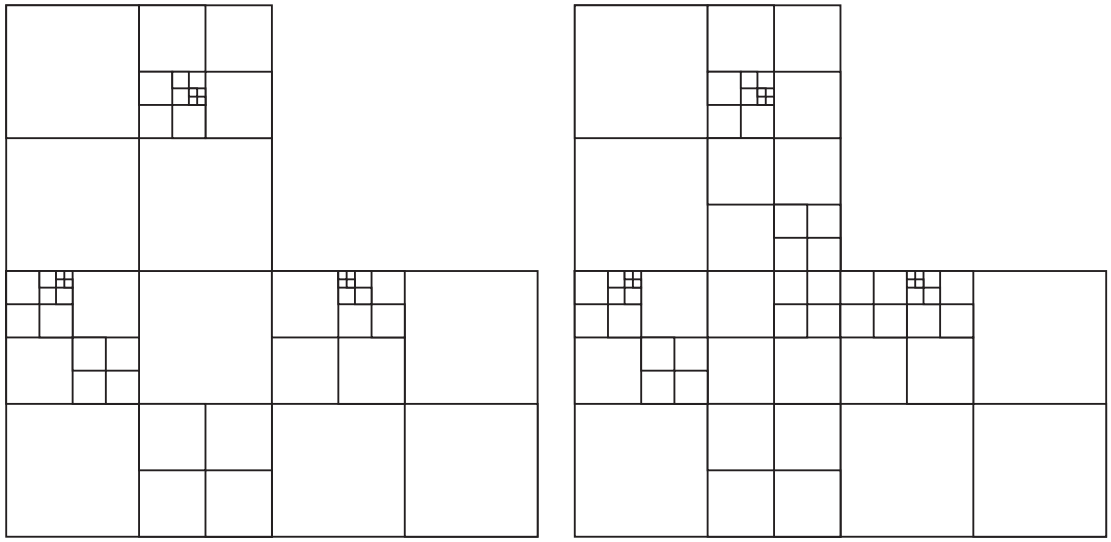
To examine the more general variable coefficient Poisson problem, we solve $\nabla \cdot (\beta \nabla u) = f$ on the same domain and quadtree structure as our original Laplace problem (Figure 16), with an exact solution of $u(x, y) = \sin(x) + \sin(y)$ and using $\beta(x, y) = \sin(x)\sin(y) + 2$ (Min's example 9.2.1). We first test Dirichlet boundary conditions, with the results shown in Table 9.

Table 5: Error analysis for adaptive Poisson problem (Section 5.4)

Number of cells	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
41	3.94×10^{-2}	-	1.83×10^{0}	-
117	1.08×10^{-2}	1.87	8.44×10^{-1}	1.12
469	3.54×10^{-3}	1.61	2.26×10^{-1}	1.90
1877	1.04×10^{-3}	1.76	7.54×10^{-2}	1.59
7509	2.94×10^{-4}	1.82	2.06×10^{-2}	1.87
30037	7.81×10^{-5}	1.91	5.51×10^{-3}	1.91
120149	2.01×10^{-5}	1.96	1.53×10^{-3}	1.85
480597	5.11×10^{-6}	1.98	4.10×10^{-4}	1.90

Table 6: Error analysis for uniform Poisson problem (Section 5.4)

Number of cells	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
85	1.84×10^{-1}	-	2.61×10^{0}	-
341	2.16×10^{-2}	3.09	3.02×10^{0}	-0.21
1365	1.13×10^{-2}	0.93	7.69×10^{-1}	1.97
5461	3.85×10^{-3}	1.55	2.44×10^{-1}	1.65
21845	9.01×10^{-4}	2.10	6.62×10^{-2}	1.88
87381	2.51×10^{-4}	1.84	1.71×10^{-2}	1.95
349525	6.63×10^{-5}	1.92	4.97×10^{-3}	1.78
1398101	1.70×10^{-5}	1.96	1.37×10^{-3}	1.86



(a) Base quadtree (Dirichlet case)

(b) After refining the non-convex corner (Neumann case)

Figure 18: The domain and initial quadtree for the tests of Section 5.5. The left figure shows the tree used for the Dirichlet case. The right figure shows the same tree after locally refining to ensure a two-layer patch of uniform cells surrounding the non-convex corner, to simplify treatment of Neumann boundary conditions.

Table 7: Error analysis for non-rectangular Poisson problem with Dirichlet boundaries (Section 5.5)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	1.45×10^{-2}	-	1.68×10^{-1}	-
64^2	5.87×10^{-3}	1.30	1.55×10^{-1}	0.12
128^2	1.03×10^{-3}	2.51	5.15×10^{-2}	1.59
256^2	2.29×10^{-4}	2.17	1.27×10^{-2}	2.02
512^2	6.06×10^{-5}	1.92	3.10×10^{-3}	2.03
1024^2	1.65×10^{-5}	1.88	7.68×10^{-4}	2.01
2048^2	4.29×10^{-6}	1.94	1.92×10^{-4}	2.00
4096^2	1.09×10^{-6}	1.97	4.78×10^{-5}	2.00

Table 8: Error analysis for non-rectangular Poisson problem with Neumann boundaries (Section 5.5)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	2.24×10^{-2}	-	6.59×10^{-1}	-
64^2	4.08×10^{-3}	2.46	1.81×10^{-1}	1.87
128^2	6.89×10^{-4}	2.56	5.07×10^{-2}	1.84
256^2	1.29×10^{-4}	2.42	1.26×10^{-2}	2.01
512^2	2.95×10^{-5}	2.13	3.08×10^{-3}	2.03
1024^2	7.85×10^{-6}	1.91	7.66×10^{-4}	2.01
2048^2	2.04×10^{-6}	1.95	1.91×10^{-4}	2.00
4096^2	5.19×10^{-7}	1.97	4.78×10^{-5}	2.00

Table 9: Error analysis for variable coefficient Poisson with Dirichlet conditions (Section 5.6)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	3.56×10^{-2}	-	5.68×10^{-2}	-
64^2	1.18×10^{-2}	1.60	1.94×10^{-2}	1.55
128^2	3.49×10^{-3}	1.75	6.06×10^{-3}	1.68
256^2	9.41×10^{-4}	1.89	1.47×10^{-3}	2.05
512^2	2.44×10^{-4}	1.95	3.69×10^{-4}	1.99
1024^2	6.21×10^{-5}	1.97	9.26×10^{-5}	2.00
2048^2	1.57×10^{-5}	1.99	2.32×10^{-5}	1.99
4096^2	3.93×10^{-6}	1.99	5.83×10^{-6}	2.00

Table 10: Error analysis for variable coefficient Poisson with Neumann conditions (Section 5.7)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	2.54×10^{-1}	-	9.73×10^{-1}	-
64^2	1.20×10^{-2}	4.40	2.12×10^{-2}	5.52
128^2	3.21×10^{-3}	1.91	5.34×10^{-3}	1.99
256^2	8.30×10^{-4}	1.95	1.31×10^{-3}	2.03
512^2	2.11×10^{-4}	1.97	3.28×10^{-4}	1.99
1024^2	5.33×10^{-5}	1.99	8.22×10^{-5}	2.00
2048^2	1.34×10^{-5}	1.99	2.06×10^{-5}	2.00
4096^2	3.36×10^{-6}	2.00	5.15×10^{-6}	2.00

Table 11: Error analysis for Poisson with hybrid discretization (Section 5.8)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
32^2	2.32×10^{-3}	-	1.89×10^{-2}	-
64^2	1.03×10^{-3}	1.17	7.83×10^{-3}	1.27
128^2	2.93×10^{-4}	1.82	2.51×10^{-3}	1.64
256^2	7.96×10^{-5}	1.88	7.18×10^{-4}	1.81
512^2	2.08×10^{-5}	1.94	1.92×10^{-4}	1.90
1024^2	5.32×10^{-6}	1.97	4.98×10^{-5}	1.95
2048^2	1.35×10^{-6}	1.98	1.27×10^{-5}	1.97
4096^2	3.39×10^{-7}	1.99	3.20×10^{-6}	1.99

5.7. Variable coefficient Poisson equation with Neumann boundary conditions

We solve the same problem again on the same domain, but using Neumann boundary conditions instead. The results are shown in Table 10.

5.8. Comparison with hybrid approach

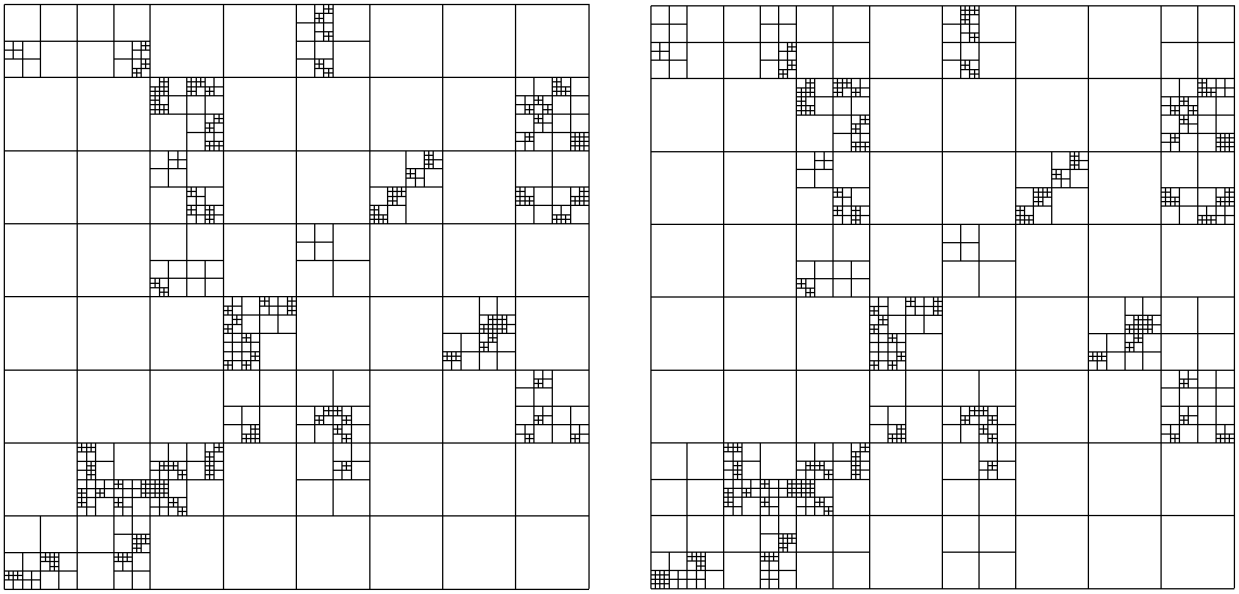
In our experience, the hybrid approach suggested in Section 4.6 tends to give comparable results to the purely diagonal-based approach. We illustrate one example by re-running the test of Section 5.1 using the hybrid discretization. Table 11 shows the results, which can be compared against the data in Table 2.

5.9. Poisson problem with complex base tree

We perform the same test problem as in Section 5.1 using a much more complex randomly generated tree structure, to demonstrate the generality of the method. The initial quadtree is shown in Figure 19 and features neighbouring cells which differ in resolution by up to 4 levels of the tree. This figure also illustrates that our boundary refinement rule requires relatively few extra subdivisions and allows for large resolution changes near boundaries. The numerical results are shown in Table 12. Runtime statistics for this computation are given in Table 13 to give a general sense for the scaling behaviour of our particular implementation.

5.10. Stability of projection

A common application of Poisson solvers is as a component of projection methods for solving the incompressible Navier-Stokes equations [2]. In this setting a key concern is the stability of the projection operation; repeated applications of a true projection operator should not affect the velocity after the first iteration. An advantage of cell-centred finite volume schemes is that this property is often preserved by construction in the discretization, whereas node-based finite difference schemes may offer only an *approximate* projection operation. In the context of non-graded trees, this is exemplified by the cell-centred approaches of Losasso



(a) Before Boundary Refinement **(b)** After Boundary Refinement

Figure 19: The more complex, random quadtree of Section 5.9, before and after applying the local refinements to support boundary conditions. Relatively few cells actually require refinement (17 out of 62 boundary cells in this example).

Table 12: Error analysis for Poisson with complex tree (Section 5.9)

Effective Resolution	$\|u - u_h\|_\infty$	Order	$\|\nabla u - \nabla u_h\|_\infty$	Order
128^2	1.88×10^{-3}	-	9.99×10^{-2}	-
256^2	5.14×10^{-4}	1.87	7.76×10^{-2}	0.36
512^2	9.44×10^{-5}	2.44	1.88×10^{-2}	2.04
1024^2	1.88×10^{-5}	2.33	4.67×10^{-3}	2.01
2048^2	4.09×10^{-6}	2.20	1.16×10^{-3}	2.01
4096^2	1.05×10^{-6}	1.96	2.90×10^{-4}	2.00

Table 13: Runtimes for Poisson with complex tree (Section 5.9)

Resolution	Tree Build (s)	Matrix Build (s)	System Solution (s)
128^2	2.09×10^{-3}	1.19×10^{-2}	7.17×10^{-3}
256^2	1.44×10^{-2}	2.56×10^{-2}	2.17×10^{-2}
512^2	6.14×10^{-2}	7.60×10^{-2}	8.81×10^{-2}
1024^2	3.17×10^{-1}	2.63×10^{-1}	4.12×10^{-1}
2048^2	1.21×10^{0}	9.39×10^{-1}	1.91×10^{0}
4096^2	7.24×10^{0}	3.92×10^{0}	1.06×10^{1}

et al. [14, 15] and the node-centred approaches of Min et al. [3, 20]; only the latter requires special treatment to enforce stability. We demonstrate below that our method preserves numerical stability by reproducing a test problem drawn from the work of Min and Gibou [20] (section 4.1), which performs repeated projection of a given vector field. The initial quadtree structure is shown in Figure 20, left. The domain is $\Omega = [0, \pi]^2$ with initial vector field given by

$$u^*(x, y) = \sin(x)\cos(y) + x(\pi - x)y^2\left(\frac{y}{3} - \frac{\pi}{2}\right), \tag{5}$$

$$v^*(x, y) = -\cos(x)\sin(y) + y(\pi - y)x^2\left(\frac{x}{3} - \frac{\pi}{2}\right). \tag{6}$$

$$\tag{7}$$

Our discretization can be used to solve for a new vector field $\mathbf{u}$ and scalar field ϕ satisfying

$$\mathbf{u} = \mathbf{u}^* - \nabla\phi, \tag{8}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{9}$$

The expected ϕ is given by $\phi = \left(\frac{x^3}{3} - \frac{\pi x^2}{2}\right)\left(\frac{y^3}{3} - \frac{\pi y^2}{2}\right)$. Figure 20, right, shows that despite applying many repeated projections to the velocity field above, the norm of the velocity error never increases, as expected.

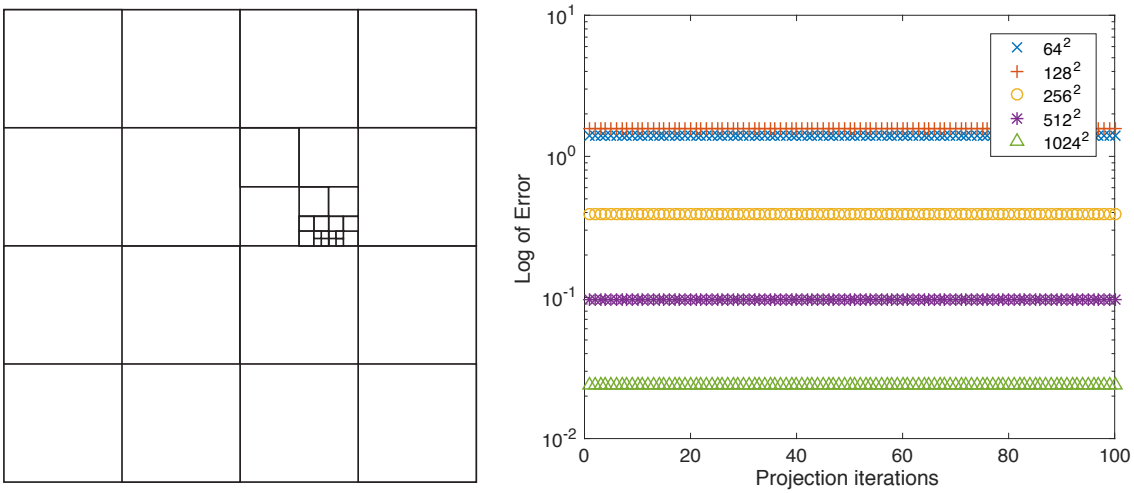


Figure 20: Left: The initial quadtree for the stability test of Section 5.10. Right: Plot of $\|\mathbf{u} - \mathbf{u}^*\|_\infty$ after repeated projection of the vector field.

We noted in Section 4.7 that the use of quadratic interpolation leads to discrete divergence and gradient operators that lack a proper orthogonality relationship. Since the standard approach to proving stability on regular grids relies on this orthogonality (e.g., [18, 42]) such a strategy is not available to us, as Minion explicitly notes [18]. Minion makes two further relevant points. First, stability can be proven for some simple refined grids using other techniques, but a proof for the general case is lacking. Second, in practice it nevertheless appears to be satisfied, as we also observe in our numerical results.

5.11. Recursion Depth

A useful question to ask is what depth of recursion is required to construct the quadratic stencils across complex T-junctions. For uniform grids it will naturally be zero. For strongly graded trees, as described in Section 3.6, only one quadratic per ghost point will ever be needed so the maximum depth is one. For the

Search Depth	Instances
1	424
2	458
3	252
4	34

Search Depth	Instances
1	910
2	1007
3	866
4	474
5	96
6	12

Table 14: Recursion depth statistics. Left: Data for the tree of Figure 19. Right: Data for the tree of Figure 21.

more general trees that we focus on, this number may be higher, but is nevertheless limited by the depth of the tree itself and the diversity of cell sizes in a particular region of the tree.

To explore this question experimentally, we can determine the maximum depth of recursion used for each ghost value in a particular tree geometry. We observed that the maximum depth of the search settles to no more than three under repeated uniform refinement. We attribute this to the particular nature of quadtree refinement; that is, arbitrary quadtrees ultimately converge to block-structured grids under uniform refinement as shown in Figure 5, and hence the number of different cell sizes in a region becomes limited. However, for initial arbitrary trees, which may have a very high degree of disparity in resolutions of nearby cells, the maximum search depth can be higher or lower. However, it is still bounded by the ratio of cell sizes in the tree.

Let us consider a few examples. The simple tree for the problem in Figure 16 has a maximum search depth of two, and an average depth (among all ghost values) of 1.39. The detailed breakdown is 35 searches of depth 1, and 22 searches of depth 2. The complex tree in Figure 19 has a maximum search depth of four (corresponding to nearby cells differing by up to four tree levels), and an average depth of 1.91. The detailed breakdown of search depths is given in Table 14, left. Finally, we constructed an even more extreme variant of the same random tree example going two levels deeper so that some neighbouring cells differ by a full six levels (Figure 21). We performed the same test, and found that the maximum search depth is six, the average depth is 2.37, and the detailed breakdown is given in Table 14, right. While such randomly structured trees with wildly differing levels of resolution may be uncommon in most applications, this stress test demonstrates that the recursion does indeed terminate relatively quickly even in such cases, rather than continuing to propagate indefinitely or to great depths.

Although our method requires recursive stencils which rely on interpolating data points that may be increasingly distant from the original ghost position, we did not observe any objectionable increase in error attributable to this fact. For example, the numerical errors using our proposed finite volume method are generally comparable in magnitude to those achieved by the finite difference scheme of Min et al. [3], which does *not* rely on recursive stencils.

5.12. Matrix Structure

The irregular neighbour relationships of quadtrees and the recursive interpolation necessary to construct ghost values will naturally lead to a greater number of nonzeros per row of the resulting matrix, as compared to the standard 5-point Laplacian stencil on uniform regular grids. However, this increased density will occur only near T-junctions. To put this to the test, we once again used the challenging scenario provided by the extremely irregular quadtree in Figure 21, and gathered statistics on the number of nonzeros per row of the matrix, shown in Figure 22. The most frequent number of nozeros per row is five, consistent with the standard 5-point stencil. While the vast majority of rows have under 20 entries per row, there is a long tail of rows with far larger numbers of nonzeros, up to a maximum of 257 in this case. We emphasize that this should be considered a stress-test of the method, and for less extreme trees this number is often much smaller; for example, with 2:1 face-grading of the same tree, the maximum number of nonzeros per row drops to 29. For the comparatively simple structure of Figure 16 the corresponding histogram of nonzeros per row is shown in Figure 23, with an upper bound of 19 entries per row.

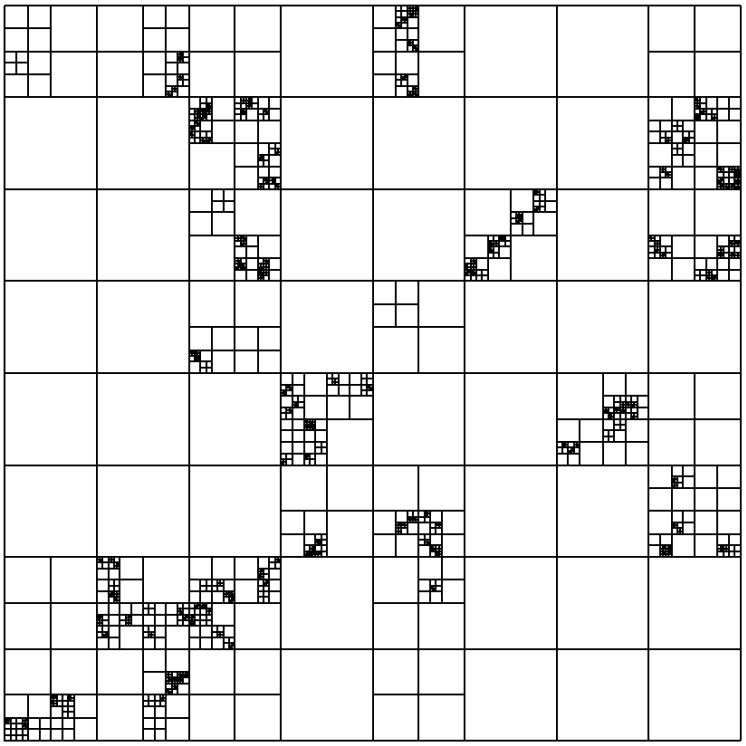


Figure 21: An even more complex quadtree with very large and random resolution jumps among neighbouring cells.

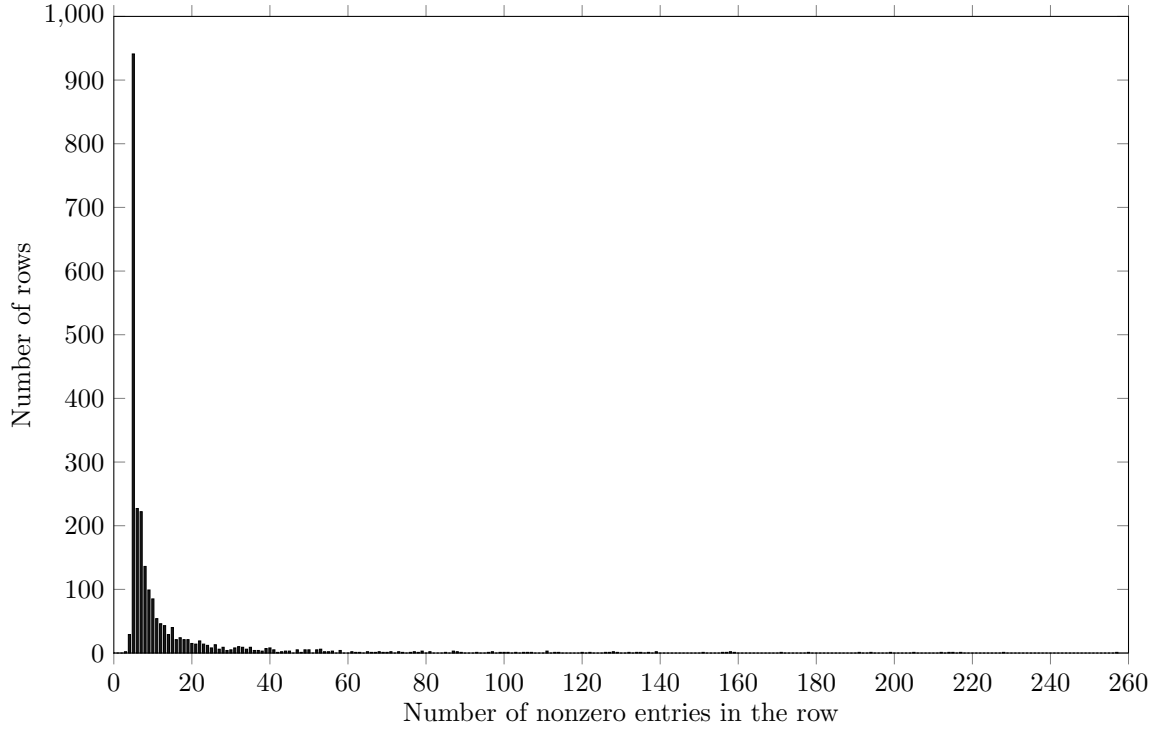


Figure 22: A histogram plot of the nonzero entries per row of the matrix in the very complex tree example of Figure 21. The peak occurs at 5, consistent with the standard 5-point stencil that applies in regular regions.

The matrix sparsity structure for the very complex tree is plotted in Figure 24. The overall coarse band structure correlates with the number of different cell sizes in the tree (e.g., in this case there are seven different cell sizes, and seven apparent bands counting from the main diagonal outwards).

5.13. Matrix Conditioning

To evaluate the effect of our discretization on the conditioning of the matrices, we estimated the condition numbers of the large sparse matrices generated by our method using MATLAB's **condest**, for the example of Section 5.1. Table 15 shows these values compared against the expected analytical condition numbers for a uniform grid of the same resolution as the finest cells in the adaptive domain, given by the classic expression $cond(A) \approx 4/\pi^2 h^2$ (see e.g., [43]). As can be seen from the data, the condition number for our adaptive approach consistently differs by a factor of approximately $1.7 - 1.8\times$ relative to the corresponding theoretical fine grid value. We observed this same behaviour when performing the analogous comparison for other scenarios; for example, on the complex tree example of Section 5.9 the same effect occurs but with a factor around $3\times$. This suggests that the (diagonal) quadratic interpolation mechanism has a fairly minor impact on the condition numbers of the resulting systems.

6. Conclusions

We have presented a novel adaptive cell-centred finite volume discretization of the variable coefficient Poisson problem which is unique among comparable schemes in providing second order accurate solutions *and gradients* on general *non-graded* quadtree grids. It can be considered a natural generalization of the cell-centred Poisson discretizations for block-structured grids originally introduced by Minion, Martin and

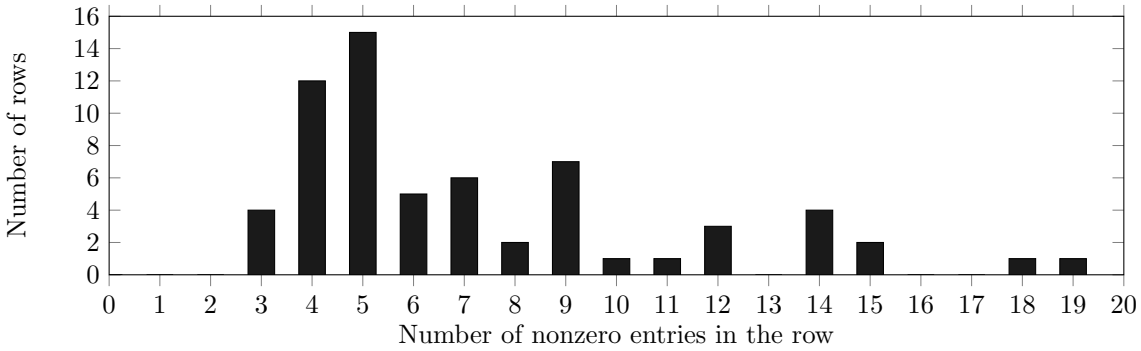


Figure 23: A histogram plot of the nonzero entries per row of the matrix in the fairly simple tree example of Figure 16.

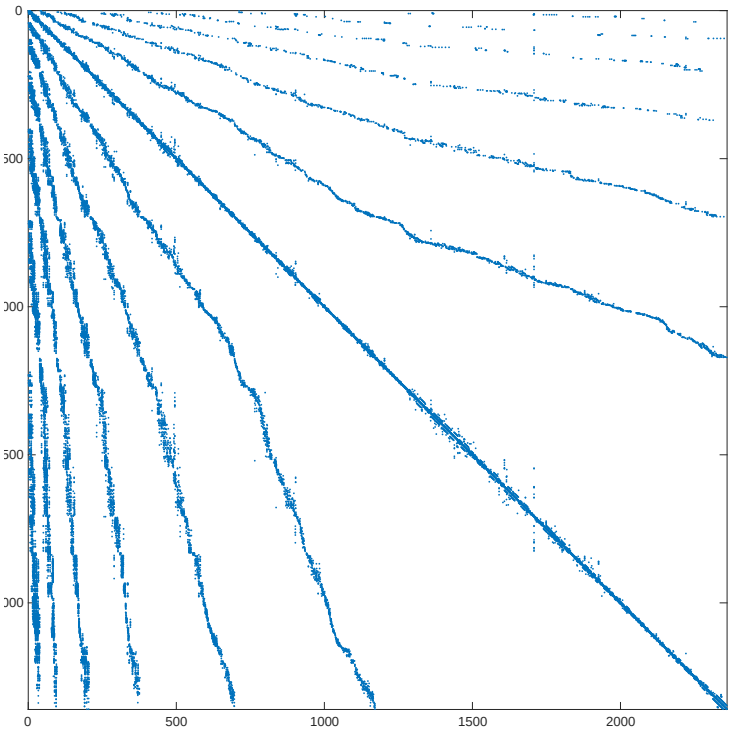


Figure 24: Matrix sparsity plot for the very complex tree example of Figure 21.

Effective Resolution	Uniform Grid Estimate	Observed Condition Number
32^2	4.205×10^1	7.171×10^1
64^2	1.682×10^2	2.815×10^2
128^2	6.728×10^2	1.193×10^3
256^2	2.691×10^3	4.713×10^3
512^2	1.076×10^4	1.887×10^4
1024^2	4.306×10^4	7.550×10^4
2048^2	1.722×10^5	3.020×10^5
4096^2	6.889×10^5	1.208×10^6

Table 15: A comparison of observed condition numbers against the approximate analytical expected condition number for a regular grid of the same effective resolution, for the problem of Section 5.1.

Cartwright [18, 6], and offers a cell-centred alternative to the node-centred quadtree scheme proposed by Min et al. [3].

While our method is highly flexible, the choice to use it will depend on the particular context. For compressible flows and other problems involving shocks, block-structured grids are considered superior to irregular trees as the former can reduce the effects of spurious shock reflections from T-junctions [14]. Where symmetric positive-definiteness is of paramount importance relative to preserving second order gradients, the method of Losasso et al. [15] remains preferable. For problems where a node-based finite difference discretization is a more natural fit than our cell-centred finite volume approach, the obvious choice is the prior work of Min et al. [3]. Nevertheless, we believe that our proposed method fills an important gap in the existing literature and will be an attractive choice in many settings; for example, in the case of incompressible flows arbitrary trees are viable, finite volume methods typically dominate, and accuracy of gradients is critical.

Our method could straightforwardly be combined with Neumann or Dirichlet boundary conditions on *irregular* domains while preserving second order solutions and gradients, by using the classic embedded boundary methods of Johansen and Colella [7] or Shortley and Weller [36], respectively. The only additional requirement would be that cells in a region near the boundary must be uniformly refined, so that the modified stencils arising near irregular boundaries do not interfere with the modified stencils used for T-junctions on the interior. In fact, Johansen and Colella [7] used Martin's block-structured refinement scheme alongside their irregular boundary approach in exactly this fashion, and Chen et al. [44] extended the node-based method of Min et al. [3] to irregular domains in much the same way.

We are interested in incorporating our Poisson discretization into the cell-centred non-graded Navier-Stokes solver recently presented by Guittet et al. [19]. That method relies on Losasso's cell-centred quadtree scheme (which yields only first order gradients) to perform the projection step; replacing that component with our scheme should offer improved accuracy of the velocity field.

As listed in Table 1, the method of Losasso et al. [15] offers less accurate gradients, but exhibits symmetric positive definiteness; this raises the question of whether it is possible to derive a cell-centred discretization with the same convergence properties as the present work (i.e., second order solutions and gradients), but that is further able to preserve matrix symmetry and positive definiteness. This echoes a two-decade-old comment of Minion [5, 18] who stated that "the author's efforts to produce a conservative second order divergence the adjoint of which in some inner product is a second order gradient have thus far failed." The machinery of mimetic finite difference methods may provide a promising direction in this vein [45], although attempts to extend it to achieve second order gradients thus far appear to require extra degrees of freedom [46].

Finally, although it is beyond the scope of the present work, a natural next step is to pursue a three dimensional generalization of our approach. Given the closely related recursive geometric structures of quadtrees and octrees, we conjecture that the same general strategy will be effective. On the other hand, the various geometric configurations are naturally somewhat more difficult to visualize and manipulate in

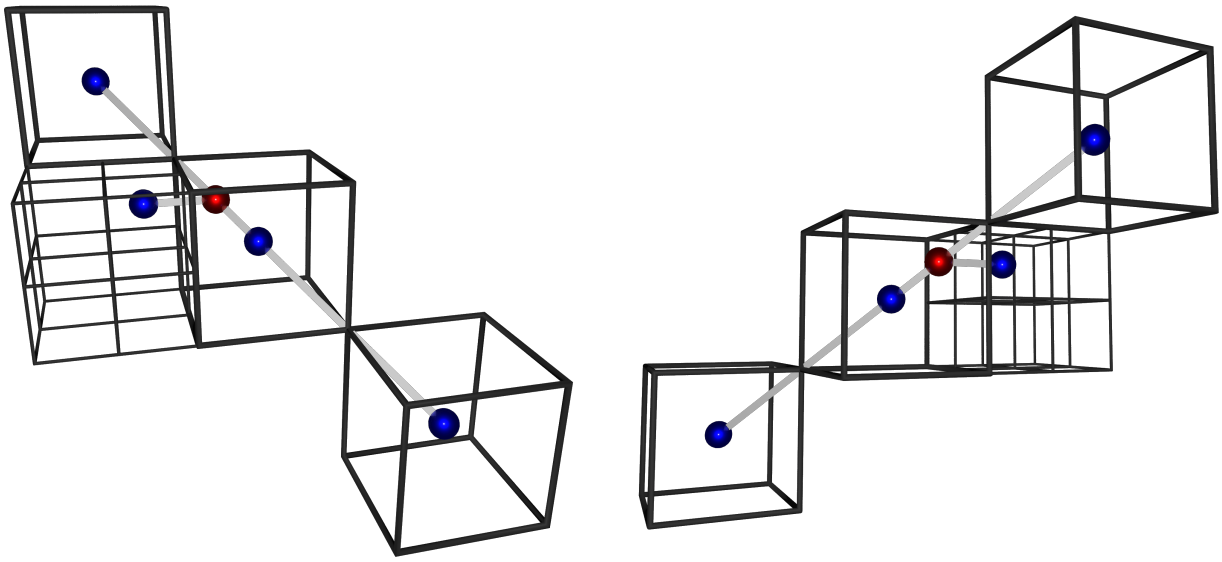


Figure 25: The simplest diagonal stencil in three dimensions, from two viewpoints. The ghost point is shown in red, with the other relevant sample points shown in blue.

3D. A first step is illustrated in Figure 25, which shows two views of the diagonal quadratic stencil for a particular graded T-junction case in 3D. One can see that the necessary stencil is placed at a 45 degree angle relative to *all three* Cartesian axes, as opposed to the two axes present in 2D. Given a sufficiently strong grading condition, this stencil (along with its symmetric counterparts) will be adequate to solve 3D problems. However, finding the full set of 3D stencils necessary to treat general *non-graded* octrees, as well as determining the practical feasibility of such an implementation and evaluating its accuracy, remains to be investigated.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Grant RGPIN-04360-2014). I am also thankful to the members of the *Computational Motion Group* at Waterloo for discussions that helped to inspire and improve this work, and to Robert Bridson for suggestions related to compatibility conditions.

Bibliography

[1] C. Grossmann, H.-G. Roos, M. Stynes, Numerical treatment of partial differential equations, Springer-Verlag, 2007.
[2] J. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer Methods in Applied Mechanics and Engineering 195 (44-47) (2006) 6011–6045.
[3] C. Min, F. Gibou, H. D. Ceniceros, A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids, J. Comp. Phys. 218 (1) (2006) 123–140.
[4] P. Colella, D. T. Graves, J. N. Johnson, N. D. Keen, T. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, B. Van Straalen, Chombo software package for AMR applications design document, Tech. rep., Lawrence Berkeley National Laboratory (2013).
[5] M. L. Minion, Two methods for the study of vortex patch evolution on locally refined grids, Ph.D. thesis, University of California, Berkeley (1994).
[6] D. F. Martin, K. L. Cartwright, Solving Poisson's equation using adaptive mesh refinement, Tech. rep., EECS Department, University of California, Berkeley (1996).

[7] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comp. Phys. 147 (1) (1998) 60–85.
[8] D. F. Martin, P. Colella, D. T. Graves, A cell-centered adaptive projection method for the incompressible Navier-Stokes equations in three dimensions, J. Comp. Phys. 227 (3) (2008) 1863–1886.
[9] A. Pletzer, B. Jamroz, R. Crockett, S. Sides, Compact cell-centered discretization stencils at fine-coarse block structured grid interfaces, J. Comp. Phys. 260 (2014) 25–36.
[10] Q. Liu, A stable and accurate projection method on a locally refined staggered mesh, International Journal for Numerical Methods in Fluids 67 (1) (2011) 74–92.
[11] W. Henshaw, A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids, J. Comp. Phys. 113 (1) (1994) 13–25.
[12] R. E. English, L. Qiu, Y. Yu, R. Fedkiw, An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids, J. Comp. Phys. 254 (2013) 107–154.
[13] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comp. Phys. 190 (2) (2003) 572–600.
[14] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Trans. Graph. (SIGGRAPH) 23 (3) (2004) 457–462.
[15] F. Losasso, R. Fedkiw, S. Osher, Spatially adaptive techniques for level set methods and incompressible flow, Computers & Fluids 35 (10) (2005) 995–1010.
[16] R. Finkel, J. L. Bentley, Quad Trees: A data structure for retrieval on composite keys, Acta Informatica 4 (1) (1974) 1–9.
[17] D. Meagher, Geometric modeling using octree encoding, Computer Graphics and Image Processing 19 (2) (1982) 129–147.
[18] M. L. Minion, A projection method for locally refined grids, J. Comp. Phys. 127 (1) (1996) 158–178.
[19] A. Guittet, M. Theillard, F. Gibou, A stable projection method for the incompressible Navier-Stokes equations on arbitrary geometries and adaptive Quad/Octrees, J. Comp. Phys. 292 (2015) 215–238.
[20] C. Min, F. Gibou, A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids, J. Comp. Phys. 219 (2) (2006) 912–929.
[21] A. Gupta, A finite element for transition from a finite grid to a coarse grid, Int. J. Numer. Methods Eng. 12 (1) (1978) 35–45.
[22] A. Tabarraei, N. Sukumar, Adaptive computations on conforming quadtree meshes, Finite elements in Analysis and Design 41 (7) (2005) 686–702.
[23] S. Natarajan, E. T. Ooi, H. Man, C. Song, Finite element computations on quadtree meshes: strain smoothing and semi-analytical formulation, Int. J. Adv. Eng. Sci. Appl. Math 7 (3) (2015) 124–133.
[24] M. Barad, P. Colella, A fourth-order accurate local refinement method for Poisson's equation, J. Comp. Phys. 209 (1) (2005) 1–18.
[25] B. Seibold, Minimal positive stencils in meshfree finite difference methods for the Poisson equation, Computer Methods in Applied Mechanics and Engineering 198 (3) (2008) 592–601.
[26] M. A. Olshanskii, K. M. Terekhov, Y. V. Vassilevski, An octree-based solver for the incompressible Navier-Stokes equations with enhanced stability and low dissipation, Computers & Fluids 84 (2013) 231–246.
[27] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, J. Comp. Phys. 225 (1) (2007) 300–321.
[28] H. Chen, C. Min, F. Gibou, A numerical scheme for the Stefan problem on adaptive Cartesian grids with supralinear convergence rate, J. Comp. Phys. 228 (16) (2009) 5803–5818.
[29] J. Papac, A. Helgadottir, C. Ratsch, Gibou, A level set approach for diffusion and Stefan-type problems with Robin boundary conditions on quadtree/octree adaptive Cartesian grids, J. Comp. Phys. 233 (2013) 241–261.
[30] M. Theillard, L. F. Djodom, J.-L. Vié, F. Gibou, A second-order sharp numerical method for solving the linear elasticity equations on irregular domains and adaptive grids - Application to shape optimization, J. Comp. Phys. 233 (2013) 430–448.
[31] M. Mirzadeh, M. Theillard, Gibou, A second-order discretization of the nonlinear Poisson-Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids, J. Comp. Phys. 230 (5) (2011) 2125–2140.
[32] Á. Helgadóttir, F. Gibou, A Poisson-Boltzmann solver on irregular domains with Neumann or Robin boundary conditions on non-graded adaptive grid, J. Comp. Phys. 230 (10) (2011) 3830–3848.
[33] M. Mirzadeh, F. Gibou, A conservative discretization of the Poisson-Nernst-Planck equations on adaptive Cartesian grids, J. Comp. Phys. 274 (2014) 633–653.
[34] D. F. Martin, P. Colella, A cell-centered adaptive projection method for the incompressible Euler equations, J. Comp. Phys. 163 (2) (2000) 271–312.
[35] G. Chesshire, W. Henshaw, Composite overlapping meshes for the solution of partial differential equations, J. Comp. Phys. 90 (1990) 1–64.
[36] G. H. Shortley, R. Weller, The numerical solution of Laplace's equation, Journal of Applied Physics 9 (5) (1938) 334.
[37] F. Gibou, R. Fedkiw, L.-T. Cheng, M. Kang, A second order accurate symmetric discretization of the Poisson equation on irregular domains, J. Comp. Phys. 176 (1) (2002) 205–227.
[38] Y. T. Ng, H. Chen, C. Min, F. Gibou, Guidelines for Poisson solvers on irregular domains with Dirichlet boundary conditions using the Ghost Fluid Method, J. Sci. Comput. 41 (2) (2009) 300–320.
[39] A. Stanoyevitch, Introduction to numerical ordinary and partial differential equations using Matlab, John Wiley & Sons, Ltd., 2011.
[40] G. Guennebaud, B. Jacob, Others, Eigen v3 (2010).
URL `http://eigen.tuxfamily.org`
[41] X. S. Li, An overview of SuperLU: Algorithms, Implementation and User Interface, TOMS 31 (3) (2005) 302–325.

[42] Y. T. Ng, C. Min, F. Gibou, An efficient fluid-solid coupling algorithm for single-phase flows, J. Comp. Phys. 228 (23) (2009) 8807–8829.
[43] J. W. Demmel, Applied numerical linear algebra, SIAM, 1997.
[44] H. Chen, C. Min, F. Gibou, A supra-convergent finite difference scheme for the Poisson and heat equations on irregular domains and non-graded adaptive Cartesian grids, J. Sci. Comput. 31 (1) (2007) 19–60.
[45] F. Brezzi, K. Lipnikov, V. Simoncini, A family of mimetic finite difference methods on polygonal and polyhedral meshes, Mathematical Models and Methods in Applied Sciences 15 (10) (2005) 1533–1551.
[46] L. B. da Veiga, G. Manzini, A higher-order formulation of the mimetic finite difference method, SIAM J. Sci. Comput. 31 (1) (2008) 732–760.