ARTICLE IN PRESS

Journal of Computational Physics xxx (2011) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions

Jeffrey Lee Hellrung Jr. [a,*], Luming Wang [b], Eftychios Sifakis [c], Joseph M. Teran [a]

[a] University of California, Los Angeles, Department of Mathematics, 520 Portola Plaza, Math Sciences Building 6363, Los Angeles, CA 90095, United States
[b] University of California, Berkeley, Department of Mathematics, 844 Evans Hall, Berkeley, CA 94720, United States
[c] University of Wisconsin, Madison, Department of Computer Sciences, 1210 W. Dayton St., Madison, WI 53706, United States

ARTICLE INFO

Article history:
Received 14 March 2011
Received in revised form 11 November 2011
Accepted 16 November 2011
Available online xxxx

Keywords:
Elliptic interface problems
Embedded interface methods
Virtual node methods
Variational methods
Multigrid methods

ABSTRACT

We present a numerical method for the variable coefficient Poisson equation in three-dimensional irregular domains and with interfacial discontinuities. The discretization embeds the domain and interface into a uniform Cartesian grid augmented with virtual degrees of freedom to provide accurate treatment of jump and boundary conditions. The matrix associated with the discretization is symmetric positive definite and equal to the standard 7-point finite difference Poisson stencil away from embedded interfaces and boundaries. Numerical evidence suggests second order accuracy in the L^∞-norm. Our approach improves the treatment of Dirichlet and jump constraints in the recent work of Bedrossian et al. [1] and introduces innovations necessary in three dimensions. Specifically, we construct new constraint-based Lagrange multiplier spaces that significantly improve the conditioning of the associated linear system of equations; we provide a method for cell-local polyhedral approximation to the zero isocontour surface of a level set needed for three-dimensional embedding; and we show that the new Lagrange multiplier spaces naturally lead to a class of easy-to-implement multigrid methods that achieve near optimal efficiency, as shown by numerical examples. For the specific case of a continuous Poisson coefficient in interface problems, we provide an expansive treatment of the construction of a particular solution that satisfies the value jump and flux jump constraints. As in [1], this is used in a discontinuity removal technique that yields the standard 7-point stencil across the interface and only requires a modification to the right-hand side of the linear system.

1. Introduction

Elliptic interface problems such as

$$-\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x}\in\Omega\setminus\Gamma; \tag{1}$$
$$[u] = a(\mathbf{x}), \quad \mathbf{x}\in\Gamma; \tag{2}$$
$$[\beta\nabla u\cdot\hat{\mathbf{n}}] = b(\mathbf{x}), \quad \mathbf{x}\in\Gamma; \tag{3}$$
$$u = p(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega_d; \tag{4}$$
$$\beta\nabla u\cdot\hat{\mathbf{n}} = q(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega_n; \tag{5}$$

* Corresponding author.
E-mail addresses: jhellrun@math.ucla.edu (J.L. Hellrung Jr.), lwang@math.berkeley.edu (L. Wang), sifakis@cs.wisc.edu (E. Sifakis), jteran@math.ucla.edu (J.M. Teran).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.11.023

have a wide variety of applications in physics and engineering, and naturally arise when two dissimilar materials interact across a thin interface. Common examples include immiscible, incompressible fluids in contact and phase change problems. The interface Γ is generally a co-dimension one closed curve (dimension 2) or surface (dimension 3) that divides the domain into an interior region Ω^- and an exterior region Ω^+ such that $\Omega = \Omega^- \sqcup \Omega^+ \sqcup \Gamma \subset \mathbb{R}^d$ ($d = 2$ or 3, typically). The scalar coefficient field β and the source term f can exhibit discontinuities across Γ, but have smooth restrictions β^σ, f^σ to Ω^σ, $\sigma \in \{-,+\}$. We let $\hat{\mathbf{n}}(\mathbf{x})$ denote the outward unit normal, whether to Ω^- at a point $\mathbf{x} \in \Gamma$ or to Ω at a point $\mathbf{x} \in \partial\Omega$; and define $[v](\mathbf{x}) := v^+(\mathbf{x}) - v^-(\mathbf{x}) := \lim_{\epsilon \to 0^+} v(\mathbf{x} + \epsilon\hat{\mathbf{n}}(\mathbf{x})) - \lim_{\epsilon \to 0^+} v(\mathbf{x} - \epsilon\hat{\mathbf{n}}(\mathbf{x}))$ as the *jump* of the quantity v across the interface Γ. The relevant physics generally determine the jumps in the solution (2) and in the flux (3), as well as the boundary conditions on $\partial\Omega$. Unless stated otherwise, we assume the surfaces Γ, $\partial\Omega$ are smooth.

Due to irregular geometry of the boundary and/or interface in many physical phenomena, a natural approach to the numerical approximation is the *finite element method* (FEM) with unstructured meshes that conform to the geometry of Γ and $\partial\Omega$ [2–9]. However, meshing complex interface geometries can prove difficult and time-consuming when the interface frequently changes shape, especially in 3 dimensions. Also, many numerical methods, such as geometric multigrid methods, do not naturally apply to unstructured meshes. These concerns are largely circumvented with the use of *embedded* (or *immersed*) methods that approximate solutions to (1)–(3) on Cartesian grids or structured meshes that do not conform to the interface. Despite advances in this direction, embedded methods that retain higher order accuracy in L^∞ often are limited to 2 dimensions and introduce relatively difficult linear algebra problems and complex implementations that sometimes require significant effort to adapt to general applications.

Recently, however, Bedrossian et al. [1] introduced a second order virtual node method for solving the elliptic interface problem (1)–(5) in 2 dimensions. The discretization presented in [1] is easy to implement and yields a symmetric positive definite sparse linear system for both interface problems and boundary value problems on irregular domains. In summary, this virtual node method employs a uniform Cartesian grid with duplicated Cartesian bilinear elements along the interface. These duplicated elements introduce additional *virtual* nodes or degrees of freedom to accurately capture the lack of regularity in the solution. The method is variational to define stencils symmetrically, and a different discretization is used depending on proximity to embedded features, allowing for the retention of the standard 5-point finite difference stencil away from embedded boundaries and interfaces. Langrange multipliers are used to enforce embedded Dirichlet conditions (4) and embedded jump conditions (2), and the choice of Lagrange multiplier space admits a symmetric positive definite discretization. In the special case when β is smooth, a discontinuity removal technique allows the use of the standard 5-point Poisson stencil even across the embedded interface.

The feature set of this virtual node approach is very powerful. In the present work, we improve many aspects of [1] and provide key modifications necessary to extend the method to 3 dimensions. Within the context of embedded Dirichlet and embedded interface discretizations, we present a novel and flexible algorithm to define the discrete Lagrange multiplier space. This algorithm gives more control on the conditioning of the resulting linear system and specifically addresses the conditioning issues (see Appendix C) we found in the straightforward extension of [1] to 3 dimensions. We also give an expanded treatment of the discontinuity removal technique, detailing an algorithm for the construction of a scalar function satisfying the jump conditions (2) and (3). Specific to the 3-dimensional implementation, we describe an algorithm for creating a polyhedral representation of cell-local interface/boundary geometry and quadrature rules suitable for these polyhedral surfaces. Finally, we present a family of multigrid algorithms that solve (1)–(5) with near-optimal multigrid efficiency.

The remainder of the paper proceeds as follows. We review existing embedded methods and related multigrid algorithms in Section 2. Section 3 presents our numerical discretizations for embedded Neumann (Section 3.2), embedded Dirichlet (Section 3.3), and embedded interface problems (Section 3.4). We outline our new constraint aggregation algorithm as it applies to our embedded Dirichlet discretization in Section 3.3.2, and detail the special case in embedded interface problems of smooth β in Section 3.4.1. Section 4 explains the components of our multigrid algorithms for all discretization types. We use numerical examples to demonstrate the accuracy of our discretization and the performance of our multigrid solvers in Section 5, and we conclude with a short summary and discussion in Section 6. We include an appendix with some additional miscellaneous details.

2. Existing methods

The *Immersed Interfaced Method* (IIM) is perhaps the most popular finite difference method for approximating (1)–(3) to second order accuracy. LeVeque and Li first proposed the IIM for approximating elliptic interface problems in [10] and the term now applies to a widely researched and extensively applied class of finite difference methods [11–17]. See [18] and the references therein for a complete exposition of the method and its numerous applications, and [19] for justification of the general IIM approach. Using generalized Taylor expansions, the original IIM adaptively modifies the stencil to obtain $\mathcal{O}(h)$ truncation error along the interface. For smooth β, this reduces to the standard 5-point or 7-point finite difference stencil, but otherwise results in an asymmetric discretization that follows from locally solving constrained optimization problems that enforce a discrete maximum principle [20]. The IIM also generally requires the evaluation of higher order jump conditions and surface derivatives along the interface. This can lead to difficulty in implementation, especially in 3 dimensions [21,18,15,17]. Chen and Strain described a new approach to the IIM, called the *Piecewise-polynomial Interface*

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

Method (PIM), in [22] that does not require the derivation of additional jump conditions and accurately treats complex interfaces. Various other attempts have been made [23–28,18] to improve the efficiency and reduce the complexity of the IIM.

Extrapolation-based finite difference schemes such as [29–34] introduce fictitious points along coordinate axes and use the known jump conditions to determine their values. The *Ghost Fluid Method* (GFM), such as that presented by Liu et al. in [29], exemplifies such methods. For 2- and 3-dimensional interface problems, the GFM neglects the tangential flux terms $[\beta \nabla u \cdot \hat{t}]$ when determining fictitious values, yielding a symmetric positive definite system and a resulting method which is first order accurate [29,35]. However, the GFM is capable of achieving up to fourth order accuracy for irregular domain problems [30,31]. The GFM is similar to our approach in spirit. We also incorporate similar ideas from the *Virtual Node Algorithm* (VNA) [36–38]. Various other approaches attain higher order accuracy by accounting for the tangential flux in other ways, often sacrificing simplicity and symmetry of discretization in the process. For instance, the *Coupling Interface Method* (CIM) [34] extends the GFM to higher order by using a second order extension at most grid points but reverting to a first order method at grid points where the second order extension cannot be applied. The method couples jump conditions in different directions to express the tangential derivatives, and the use of one-sided differences results in an asymmetric discretization. Similarly, the *Matched Interface and Boundary (MIB) method* [33] uses higher order extrapolations of the solution matched with higher order one-sided discretizations of the jump conditions to determine the values at fictitious points. The MIB method accounts for non-zero $[\beta \nabla u \cdot \hat{t}]$ by differentiating the given jump conditions using one-sided interpolations. This widens the stencil in several directions that depend on the local geometry, and results in an asymmetric discretization. The work of [39] extended the MIB method to handle high curvature geometry, the work of [40] provide a 3-dimensional version, and more recent progress is given in [41]. Pan et al. in [42] derived symmetric finite difference formulas (in 1 and 2 dimensions) within the MIB framework. In [43,44] Hou et al. also use techniques seemingly inspired by the analysis of the original GFM approach done in [29,35]. They develop a second order variational GFM by altering finite element interpolating functions to capture the jump conditions in the solution. Their approach is remarkably robust to non-smooth interface geometry (especially [44]), but results in an asymmetric discretization in the general case. The recent works of [45,46] treated the cases of Robin and Neumann boundary conditions by altering the 5-point stencil along the boundary using a finite volume-like approach. This results in a symmetric positive definite system.

Ideas similar to the extrapolation-based finite difference schemes have also seen extensive use in the FEM community, for instance in fictitious domain methods [47–50], the *Discontinuous Galerkin (DG) method* [51,52], the *eXtended Finite Element Method* (XFEM) [53–61] and [62],[1] and other non-conforming finite element methods [64–76]. Fictitious domain methods handle embedded features by including every element that intersects the feature into the discretization. This naturally introduces "virtual nodes" or "ghost nodes" into the resulting discretization. The XFEM enriches the standard finite element basis with additional discontinuous basis functions, thereby introducing new degrees of freedom. These basis functions exist only at the nodes of elements that intersect the embedded interface and usually are the standard basis elements multiplied by a generalized Heaviside function. The methods of [65,67,68,37,72,76] introduce a related virtual node concept to provide the additional degrees of freedom required to represent the discontinuities. The most straightforward implementation of this virtual node concept [67,68,72] yields a representation equivalent to the standard Heaviside enrichment of the XFEM. However, this approach generalizes to the slightly richer representations of [36,38,76] that attain more geometric detail, particularly when dealing with coarse grids and non-smooth interfaces. Moreover, virtual node representations are considered more geometrically intuitive and easier to incorporate into existing FEM code [68,72,76] than traditional Heaviside enrichment.

The solution spaces of these FEM approaches generally do not satisfy the embedded boundary or interface conditions. Thus, these methods impose linear constraints with either penalty methods or Lagrange multipliers to enforce the conditions in some weak sense. For example, see [47,69,72,49,61] and the references therein. When using Lagrange multipliers, the Ladyzhenskaya–Babuška-Brezzi inf-sup conditions place stringent limitations on the types of constraints that will retain optimal convergence rates of the approximation spaces [77–79,59,69,51]. Such inf-sup restrictions generally limit the strength of the Lagrange multiplier space relative to the solution approximation space. For certain elements, designing the proper approximation spaces is a non-trivial task [57,59]. Moreover, the use of Lagrange multipliers requires the solution of an indefinite saddle point system that can potentially introduce significant cost. Applying stabilization through a consistent penalty method, such as Nitsche's method, presents an alternative approach [67,69,70,72,73,75]. However, these can have adverse effects on conditioning and require the determination of the stabilization parameters. Instead of using Lagrange multipliers or stabilization, the methods of [66,43,58,80,71,44,74] alter the basis functions to either satisfy the constraints directly, or simplify the process of doing so. In this regard, such methods represent the finite element analogues of the IIM.

The method of [81] offers a finite volume approach to embedded domain problems. Like some fictitious domain methods, XFEM, and our virtual node method, this method uses partially empty cells along the boundary. However, the one-sided quadratic interpolations used to compute the fluxes along the boundary yield an asymmetric system. See [82,83] for a more recent 3-dimensional version applied to Poisson's equation and the heat equation. In [84], Oevermann and Klein proposed a second order finite volume method for interface problems, and simplified and extended their method to 3-dimensions in [85]. In an approach similar to ours, any Cartesian cell that intersects the interface yields a distinct multilinear representation of the solution. The jump conditions are then built into the difference stencil by locally solving constrained

[1] See [63] for corrections to IIM convergence estimates.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

overdetermined systems. An asymptotic technique resolves the problem of vanishing cell volumes, though it requires specific treatment for each possible cell geometry. The resulting system is asymmetric for the general case of $[\beta] \neq 0$.

When $[\beta] \neq 0$ the majority of these second order methods do not retain a symmetric positive definite system. While the FEM approaches that use stabilization do retain a symmetric positive definite system [72], generally the finite element methods that use Lagrange multipliers, such as [54], result in a symmetric indefinite system. Although we use Lagrange multipliers, we present a simple method of reducing the indefinite system to a symmetric positive definite system using a null space method. On the other hand, when the coefficient β is smooth across the interface, methods such as the original IIM achieve second order accuracy by only altering the right-hand side of the system. For this case, we present a method that uses the virtual node framework that also retains the original left-hand side.

Several of the above works employ multigrid methods to solve the resulting linear systems. Black-box multigrid solvers, either of a purely algebraic variety [21,84,34,85] or of a more geometric variety [20], are often efficient alternatives to, or may be combined with, Krylov solvers [22,86]. However, less general multigrid algorithms specially tuned to the particular discretization method may outperform a black-box multigrid solver; see, for example, [28,86]. Some methods lend themselves to using relatively straightforward extensions of standard geometric multigrid techniques, including both mortar finite element methods [7,9] and embedded methods [48,81–83], usually with special attention being paid near irregular features. Many of the works describing IIM-based discretizations [26–28,22] utilize a multigrid solver with a grid hierarchy defined geometrically but incorporate algebraic techniques in the remaining components (coarse-grid operators and grid transfer operators). In [87] Wan and Liu discuss the transfer operators near embedded features in a geometric multigrid method for irregular domain discretizations in general. In contrast to the multigrid approaches in many of the preceding works on embedded discretizations, our multigrid algorithms define the grid hierarchy, coarse-grid operators, and grid transfer operators geometrically, hence allow for efficient implementations that have lower memory requirements and increased parallelizability.

3. Discretization

Our numerical discretizations for Neumann, Dirichlet, and interface problems make use of an embedding of the domain or interface within a uniform Cartesian grid. We thus first outline this embedding procedure and the associated notation. We subsequently describe our embedded Neumann discretization, and we will then see how an alteration of our treatment of the boundary conditions in embedded Neumann problems yields our discretization for embedded Dirichlet problems. Finally, we will show how a natural combination of our embedded Neumann and embedded Dirichlet discretizations allows us to deal with interfacial discontinuities.

3.1. Domain and interface embedding and integration

Let us first consider the treatment of the domain Ω for embedded Neumann and embedded Dirichlet problems. We embed the domain Ω into a non-conforming, uniform Cartesian grid $\mathcal{G}^h$ with grid-spacing $(\Delta x, \Delta y, \Delta z)$. (Note that to simplify the convergence analysis, our numerical examples assume $\Delta x = \Delta y = \Delta z =: h$.) We include all Cartesian grid cells c_i that intersect Ω in the discretization, and refer to this set $\mathcal{C}^h = \{c_i \in \mathcal{G}^h : c_i \cap \Omega \neq \emptyset\}$ as the *computational domain* (see Fig. 1). Also, we define the set of all cells that intersect the boundary as $\mathcal{C}^h_{\partial\Omega} = \{c_i \in \mathcal{C}^h : c_i \cap \partial\Omega \neq \emptyset\}$ and refer to these as *boundary (grid) cells*. Note that a boundary cell may be regarded as partially empty, since only a portion of the cell lies within Ω. We refer to this region of a boundary cell c_i that lies in the domain Ω, $c_i \cap \Omega$, as the *material region* of the cell, and use the terms *material node* and *virtual node* to describe the Cartesian grid vertices lying inside and outside Ω, respectively. We refer to the set of grid vertices spanned by the computation domain as $\mathcal{N}^h$, and specifically the material nodes as $\mathcal{N}^h_m$ and the virtual nodes as $\mathcal{N}^h_v$. See Fig. 1 for a diagram labeling the grid vertices along a typical boundary. For embedded Neumann and embedded Dirichlet problems, we identify each grid vertex, material or virtual, as a degree of freedom.

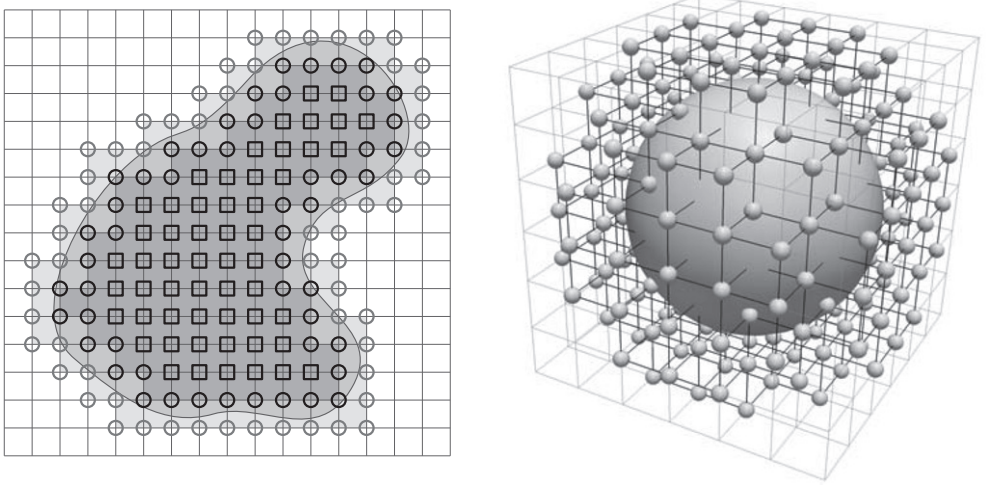
In the course of the discretization, for each boundary cell $c_i \in \mathcal{C}^h_{\partial\Omega}$, we will need to evaluate integrals over the following domains:

- the material volume within a cell, $c_i \cap \Omega$;
- the boundary of the material volume within a cell, $\partial(c_i \cap \Omega)$; and
- the boundary of Ω within a cell, $c_i \cap \partial\Omega$ (which is contained within $\partial(c_i \cap \Omega)$).

In all cases, the integrand is polynomial (or locally approximated by a polynomial). We evaluate these integrals using polyhedral representations $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$ approximating $\partial(c_i \cap \Omega)$ and $c_i \cap \partial\Omega$, respectively. We use the term *polyhedral representation* to convey an analogous meaning as polygonalizing a curve in 2 dimensions, but we essentially regard $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$ simply as collections of polygons. For implementation purposes, to maximize data structure reuse, it is convenient for $\mathcal{P}^{c_i}_{\partial\Omega} \subset \mathcal{P}^{c_i}$, i.e., all polygons in $\mathcal{P}^{c_i}_{\partial\Omega}$ are also members of $\mathcal{P}^{c_i}$. See Fig. 2.

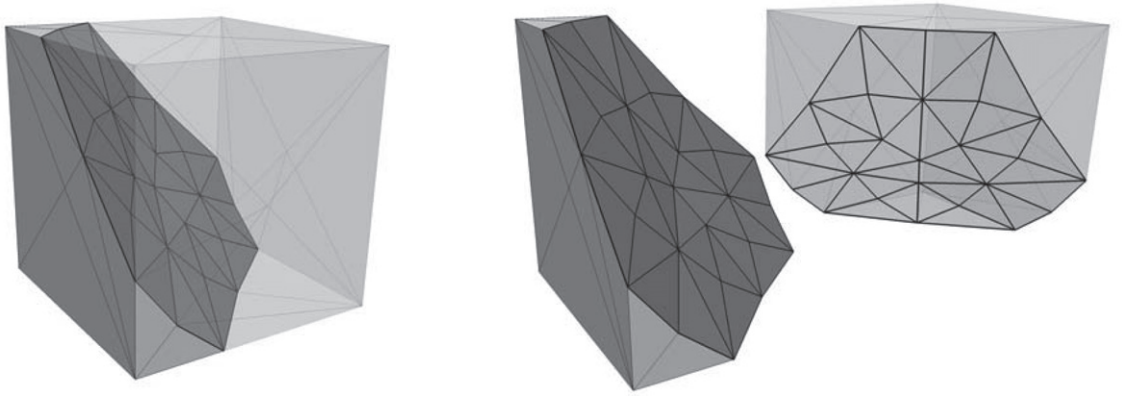
We employ the divergence theorem to transform volume integrals over $c_i \cap \Omega$ into surface integrals over $\partial(c_i \cap \Omega)$ (cf. [88]). Such transformations are non-unique, but constructing a simple one is straightforward given the polynomial nature of the integrand. For example,

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Embedding in 2 dimensions (b) Embedding in 3 dimensions

Fig. 1. Example domain embeddings for Neumann and Dirichlet problems. Subfigure (a) shows an example in 2 dimensions to clearly depict the various classes of grid cells and vertices: shaded grid cells comprise the computational domain ($\mathcal{C}^h$), with lighter-shaded grid cells on the boundary ($\mathcal{C}^h_{\partial\Omega}$); grid vertices surrounded by gray circles represent virtual degrees of freedom ($\mathcal{N}^h_v$); grid vertices surrounded by black circles represent material degrees of freedom ($\mathcal{N}^h_m$) incident to a boundary grid cell; and grid vertices surrounded by squares represent material degrees of freedom ($\mathcal{N}^h_m$) incident only to non-boundary grid cells. Subfigure (b) shows an example in 3 dimensions.



(a) Boundary grid cell with a polyhedralization of a portion of $\partial\Omega$ embedded inside (b) The two halves of the boundary grid cell after division along $\partial\Omega$

Fig. 2. A grid cell c_i with an example boundary dividing it. The left half of the cell in (b) corresponds to $c_i \cap \Omega$, the material region of the cell. (b) Shows the polyhedralization $\mathcal{P}^{c_i}$ of the material region of the cell, where the shaded triangles highlight $\mathcal{P}^{c_i}_{\partial\Omega} \subset \mathcal{P}^{c_i}$, the polyhedralization just of the portion of $\partial\Omega$ passing through c_i.

$$\int_{c_i\cap\Omega} x^p y^q z^r d\mathbf{x} = \int_{c_i\cap\Omega} \frac{1}{p+1}\nabla\cdot\left(x^{p+1}y^q z^r, 0, 0\right) d\mathbf{x} = \int_{\partial(c_i\cap\Omega)} \frac{1}{p+1}\left(x^{p+1}y^q z^r, 0, 0\right)\cdot\hat{\mathbf{n}}(\mathbf{x}) d\mathbf{S}(\mathbf{x}).$$

We decompose surface integrals over $\partial(c_i \cap \Omega)$ and $c_i \cap \partial\Omega$ into a sum of integrals over the component polygons of $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$, respectively. For example, given a vector-valued function $\mathbf{h}(\mathbf{x})$,

$$\int_{\partial(c_i\cap\Omega)} \mathbf{h}(\mathbf{x})\cdot\hat{\mathbf{n}}(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \sum_{g\in\mathcal{P}^{c_i}} \int_g \mathbf{h}(\mathbf{x})\cdot\hat{\mathbf{n}}_g d\mathbf{S}(\mathbf{x}).$$

Note that over each polygon $g \in \mathcal{P}^{c_i}$, the unit normal $\hat{\mathbf{n}}_g$ is constant, hence $\mathbf{h}(\mathbf{x})\cdot\hat{\mathbf{n}}_g$ restricted to g is a polynomial in $\mathbf{x}$ (assuming that the components of $\mathbf{h}$ are polynomials to begin with). To evaluate these polygon-local surface integrals, one could make a change of variables into a localized coordinate system and again apply the divergence theorem. However, the polynomial integrand may have degree as high as 5, and this change of variables requires a computationally intensive expansion

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

of a composition of the integrand with the coordinate transformation. We found it simpler to triangulate each polygon and use a Gaussian quadrature rule over each component triangle. As the polygons in our implementation are limited to triangles and convex quadrilaterals (see Section 3.1.1 below), such a triangulation is trivial. To maximize efficiency while ensuring the quadrature is exact, we use a quadrature rule of order equal to the degree of the polynomial integrand. For specific quadrature rules up to order 5, we refer the reader to Appendix A.

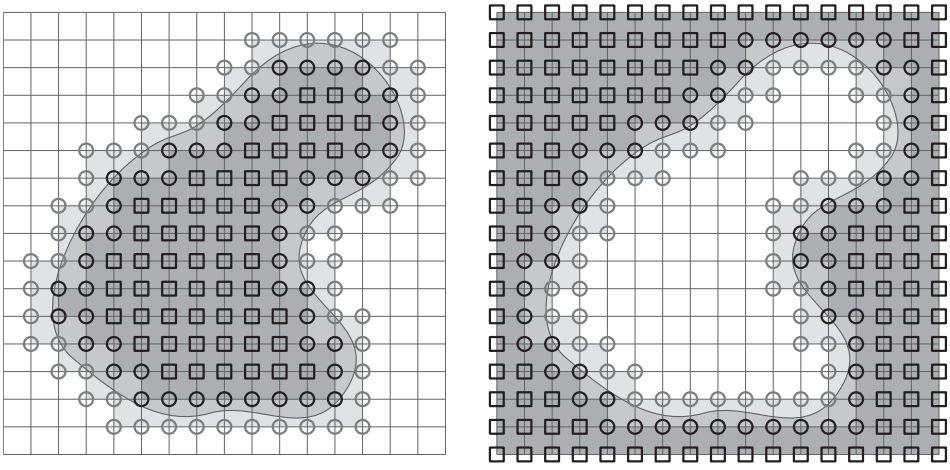
For embedded interface problems, we embed the interface Γ into $\mathcal{G}^h$ in a completely analogous way as for the domain boundary $\partial\Omega$ in embedded Neumann and embedded Dirichlet problems. We likewise use the notation $\mathcal{C}_\Gamma^h = \{c_i \in \mathcal{G}^h : c_i \cap \Gamma \neq \emptyset\}$ and the term *interfacial (grid) cells* to refer to the set of cells through which the interface passes. As we will see in Section 3.4, our interface discretization is based on an embedded domain discretization, as described above, in each of Ω^- and Ω^+. This naturally introduces an *interior computational domain* $\mathcal{C}^{h,-}$ and *exterior computational domain* $\mathcal{C}^{h,+}$, where $\mathcal{C}^{h,\sigma} = \{c_i \in \mathcal{G}^h : c_i \cap \Omega^\sigma \neq \emptyset\}$. Note that $\mathcal{C}^{h,-}$ and $\mathcal{C}^{h,+}$ are disjoint save for $\mathcal{C}_\Gamma^h$, where each Cartesian grid cell and the associated degrees of freedom, material and virtual, are duplicated. See Fig. 3.

We will often speak generically about both our interface discretization and our Dirichlet and/or Neumann discretizations. Due to the similarities in the embedding of $\partial\Omega$ (for Neumann and Dirichlet problems) and of Γ (for interface problems) into the background grid $\mathcal{G}^h$, and to avoid cluttering the exposition with too many "boundary/interface" terms, we will occasionally simply use the term *embedded feature* to refer both to the embedded domain boundary $\partial\Omega$ in Neumann and Dirichlet problems and to the embedded interface Γ in interface problems.

3.1.1. Embedded feature polyhedralization

We define all of the embedded domain boundaries $\partial\Omega$ and embedded interfaces Γ in the numerical examples in Section 5 analytically and implicitly as the zero isocontour of a level set function. This Eulerian representation ensures that we can always resolve embedded features to a resolution comparable to the background grid $\mathcal{G}^h$. Note that the embedding procedure and integration techniques described above require an explicit polyhedral representation of the embedding within each boundary/interfacial grid cell. Thus, one must create some polyhedral approximation, per boundary/interfacial grid cell, of the implicitly defined embedding. Since it is relatively easy to divide a tetrahedron along a plane approximating the level set surface given the level set function values at the tetrahedron's vertices, we symmetrically partition each boundary/interfacial grid cell into 24 congruent tetrahedra and accordingly divide each tetrahedron. The union of these dividing surfaces (triangles and quadrilaterals) within each tetrahedron compose the polyhedral representation of the embedded boundary or embedded interface. In 2 dimensions, the analogous procedure would be to partition each square grid cell into 4 triangles and divide each triangle by a line according to the level set function values at the triangle's vertices. This polyhedralization procedure is similar to that described in [88]. See Fig. 4.

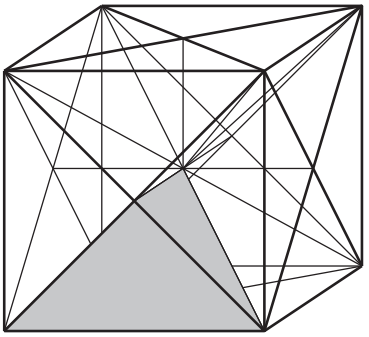
The procedure described above may produce a *sliver* polyhedron (a polyhedron with large aspect ratio) when dividing a given tetrahedron; likewise, the polygonal representation of the embedded surface may contain some sliver polygons. We note that the aspect ratio of such primitives has no *direct* bearing on the conditioning of the discretization. The quantities



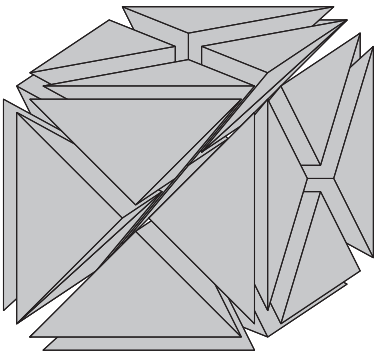
(a) Embedding for Ω^- (b) Embedding for Ω^+

Fig. 3. An example interface embedding in 2 dimensions, showing the separate domain embeddings for Ω^- and Ω^+. Grid cells and grid vertices are labelled as in Fig. 1: shaded grid cells comprise the interior (Ω^-, (a)) and exterior (Ω^+, (b)) computational domains, with the lighter-shaded grid cells on the interface; grid vertices surrounded by gray circles represent virtual degrees of freedom; grid vertices surrounded by black circles represent material degrees of freedom incident to an interfacial grid cell; and grid vertices surrounded by squares represent material degrees of freedom incident only to non-interfacial grid cells. Notice how all interfacial grid cells and circled grid vertices are effectively duplicated between the grids embedding the interior and exterior domains. Also note that each grid vertex on an interfacial grid cell is duplicated into precisely one material degree of freedom and one virtual degree of freedom.

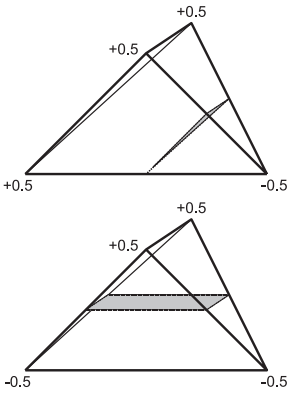
Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



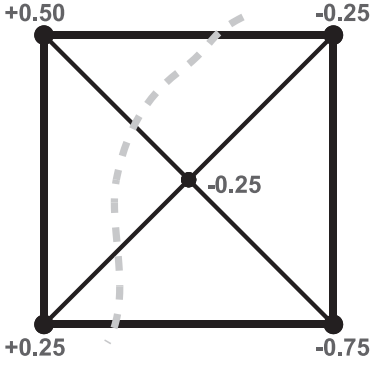
(a) Grid cell partitioned into 24 congruent tetrahedrons (wireframe)



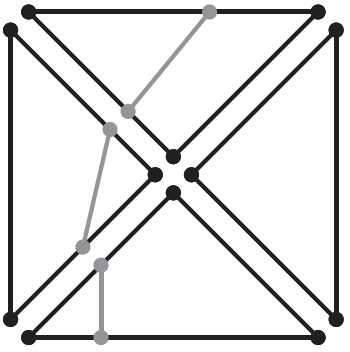
(b) Grid cell partitioned into 24 congruent tetrahedrons (separated)



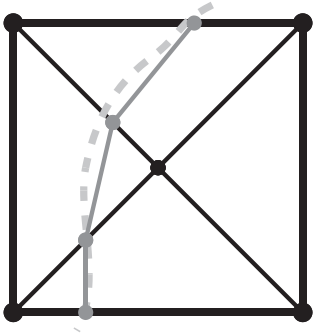
(c) Typical divisions of a tetrahedron given the level set function values at its vertices



(d) Grid cell (dimension 2) partitioned into 4 congruent triangles, with a typical level set curve (light gray; $\partial\Omega$ or Γ) and level set function values at the vertices of the triangles.



(e) The division of each triangle according to the level set function values at its vertices.



(f) The polygonal representation (gray) of the level set curve is the union of the dividing segments within each triangle.

Fig. 4. We approximate an embedded boundary or embedded interface implicitly defined by a level set function with a polyhedral representation computed by partitioning each boundary/interfacial grid cell into 24 congruent tetrahedra, as in (a) and (b); and subsequently dividing each tetrahedron according to the level set function values at its vertices, e.g., as in (c). The union of the dividing triangles and quadrilaterals within each divided tetrahedron compose the polyhedral representation of the embedded boundary or embedded interface. In 2 dimensions, the analogous procedure would be to partition each square grid cell into 4 triangles, as in (d), and divide each triangle according to the level set values at its vertices, as in (e). The union of the dividing segments within each triangle compose the polygonal representation of the embedded boundary or interface, as in (f).

of actual relevance to conditioning are the measures of the material volume and the embedded surface within a boundary/interfacial grid cell. Unlike the conditioning issues associated with sliver elements in a conforming mesh, however, our method allows conditioning issues caused by vanishing material volume measures within a grid cell to be addressed via Jacobi preconditioning, as we discuss at the end of Section 3.2. Further, our constraint aggregation method described in Section 3.3.2 fully alleviates any conditioning issues caused by vanishing embedded surface measures within a grid cell (which are only relevant within the context of discretizing the Dirichlet boundary conditions (4) and the value jump interface conditions (2)). See also [51] for a more detailed discussion on the advantages, with respect to conditioning, of using embedded domain methods over conforming mesh methods such as locally boundary-fitting remeshing schemes.

3.2. Embedded Neumann

Our discretization of embedded Neumann problems is a generalization of the 2-dimensional method given by Bedrossian et al. [1], and is similar to some XFEM approaches, e.g. [54], as well as the early work of Almgren et al. in [48]. We discretize the embedded Neumann problem,

$$\begin{aligned} -\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x}\in\Omega; \\ \beta(\mathbf{x})\nabla u(\mathbf{x})\cdot\hat{\mathbf{n}} &= q(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega; \end{aligned} \tag{6}$$

using the energy minimization form of (6):

over all $u \in \mathbf{H}^1(\Omega)$, minimize

$$E(u) := e(u) - (f,u)_\Omega - (q,u)_{\partial\Omega} := \int_\Omega \frac{1}{2}\nabla u \cdot \beta \nabla u d\mathbf{x} - \int_\Omega f u d\mathbf{x} - \int_{\partial\Omega} q u d\mathbf{S}(\mathbf{x}). \tag{7}$$

We choose to discretize the energy minimization problem because this straightforwardly yields a symmetric system; it naturally incorporates the Neumann boundary conditions into the right-hand side of the system; and it provides the necessary setting to ensure accuracy of the discretization near the boundary. We define the solution space $\mathbf{V}^h \subset \mathbf{H}^1(\Omega)$ as the space of continuous functions that are trilinear over the material region of each cell $c_k \in \mathcal{C}^h$. For $u^h \in \mathbf{V}^h$, we write $u^h(\mathbf{x}) = \sum_{i=1}^n u_i N_i(\mathbf{x})$ for $\vec{u} = (u_1, \ldots, u_n)^t \in \mathbf{R}^n$. Here $N_i(\mathbf{x})$ is the standard piecewise trilinear interpolation basis function associated with grid vertex i; and n denotes the number of degrees of freedom in the discretization, equal to the number of grid vertices that compose the cells of $\mathcal{C}^h$.

Using the above representation of $u^h \in \mathbf{V}^h$, we define a discrete energy $E^h(u^h)$ approximating $E(u^h)$. Although we could discretize the energy directly from the piecewise trilinear representation of u^h, this would result in a 27-point stencil everywhere, even away from the boundary. To retain the standard second order 7-point stencil away from the boundary we use different discretizations of the energy over $\mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}$ and over $\mathcal{C}^h_{\partial\Omega}$,

$$E^h(u^h) := \left(\sum_{c_k \in \mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}} e^{c_k}(u^h)\right) + \left(\sum_{c_k \in \mathcal{C}^h_{\partial\Omega}} \tilde{e}^{c_k}(u^h)\right) - \left(\sum_{c_k \in \mathcal{C}^h} (f,u^h)^{c_k}_\Omega\right) - \left(\sum_{c_k \in \mathcal{C}^h_{\partial\Omega}} (q,u^h)^{c_k}_{\partial\Omega}\right), \tag{8}$$

where the superscripts denote restriction to cell c_k. For cells $c_k \in \mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}$ that *do not* intersect the boundary, we define $e^{c_k}(u^h)$ as

$$e^{c_k}(u^h) := \frac{1}{2}\bar{\beta}\Delta x \Delta y \Delta z \Big((D_x u^h)^2 + (D_y u^h)^2 + (D_z u^h)^2 \Big).$$

Here $\bar{\beta}$ denotes a cell average of β; and $(D_x u^h)^2$ denotes the average of the squared finite difference approximations of $\partial_x u^h$ over the 4 x-oriented edges in the cell:

$$(D_x u^h)^2 := \frac{1}{4}\sum_{s,t\in\{0,1\}} \left(\frac{u_{i+1,j+s,k+t} - u_{i,j+s,k+t}}{\Delta x}\right)^2,$$

where $\{u_{p,q,r}\}$ denote the degrees of freedom at the 8 corners of the cell. $(D_y u^h)^2$ and $(D_z u^h)^2$ likewise denote approximations to $(\partial_y u^h)^2$ and $(\partial_z u^h)^2$, respectively. On the other hand, for cells $c_k \in \mathcal{C}^h_{\partial\Omega}$ that *do* intersect the boundary, we use the Cartesian trilinear representation of u^h to define $\tilde{e}^{c_k}(u^h)$. If we let $\mathcal{N}^h_{c_k}$ denote the indices of the 8 vertices at the corners of the cell c_k, and let $\{N_i : i \in \mathcal{N}^h_{c_k}\}$ denote the corresponding trilinear basis functions, then this yields the discretization

$$\tilde{e}^{c_k}(u^h) := \frac{1}{2}\sum_{i,j\in\mathcal{N}^h_{c_k}} \left(\bar{\beta}\int_{c_k\cap\Omega} \nabla N_i \cdot \nabla N_j d\mathbf{x}\right) u_i u_j. \tag{9}$$

Note that $\nabla N_i \cdot \nabla N_j$ is a 4th-degree polynomial, hence we can evaluate these integrals as described in Section 3.1. Like the integrals, the cell average of β, $\bar{\beta}$, is computed only over the material region of the cell, $c_k \cap \Omega$.

We discretize the remaining forms cell-wise, as:

$$(f,u^h)^{c_k}_\Omega := \sum_{i\in\mathcal{N}^h_{c_k}} \left(\bar{f}\int_{c_k\cap\Omega} N_i d\mathbf{x}\right) u_i;$$

$$(q,u^h)^{c_k}_{\partial\Omega} := \sum_{i\in\mathcal{N}^h_{c_k}} \left(\bar{q}\int_{c_k\cap\partial\Omega} N_i d\mathbf{S}(\mathbf{x})\right) u_i.$$

Similar to $\bar{\beta}$, $\bar{f}$ is the average source over $c_k \cap \Omega$, and $\bar{q}$ is the average normal flux over $c_k \cap \partial\Omega$. Again, all integrals above have polynomial integrands, hence we can evaluate these integrals as described in Section 3.1. See Appendix B for details on how we computed $\bar{\beta}$, $\bar{f}$, and $\bar{q}$ for the numerical examples in Section 5.

Lastly, we minimize the discrete energy (8) by solving the linear system

$$\begin{aligned} &A\vec{u} = \vec{f}, \\ &A_{ij} := \frac{\partial^2}{\partial u_i \partial u_j} E^h(u^h), \\ &f_i := \frac{\partial}{\partial u_i}\left((f,u^h)_\Omega + (q,u^h)_{\partial\Omega}\right) \end{aligned} \tag{10}$$

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

for the vector $\vec{u}$. We use the standard FEM term *stiffness matrix* to refer to the matrix A, and it is clear from the derivation that A is symmetric and positive semi-definite. Indeed, its null space is spanned by the vector $\vec{u} = (1, 1, \ldots, 1)^t$ corresponding to $u^h \equiv 1$.

With this approach, our definition of the energy (9) results in a slightly denser stencil near the boundary, as all 8 degrees of freedom in a cell couple together if $\partial\Omega$ passes through that cell. See Fig. 5 for a graphical depiction of the stencil definitions and the sparsity pattern of the stiffness matrix.

The symmetric system (10) readily lends itself to black-box solvers such as (preconditioned) conjugate gradient. However, conditioning of the stiffness matrix may deteriorate when a cell has a very small material volume measure, as we first mentioned in Section 3.1.1. This arises from the increasing irrelevance of virtual nodes far from the boundary (see, for example, the (4,12) grid vertex in Fig. 5). The respective row and column in A and the corresponding entry in $\vec{f}$ all approach zero simultaneously. We found that simple Jacobi preconditioning (and, in extreme cases, outright elimination of degrees of freedom; see Section 5 for explanation) mitigates these conditioning issues as in [1]. Note however that our multigrid solver described in Section 4 naturally suffers no such adverse effects from A's conditioning.

3.3. Embedded Dirichlet

Following the progression in [1], we extend our embedded Neumann approach to solve embedded Dirichlet problems,

$$\begin{aligned} -\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x}\in\Omega, \\ u(\mathbf{x}) &= p(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega, \end{aligned} \tag{11}$$

within our virtual node framework. We will show how a further extension will naturally yield a discretization for embedded interface problems, resulting in a method that encapsulates all types of boundary conditions in a unified framework.

For the embedded Dirichlet case, we use the constrained minimization problem:

over all $u \in \mathbf{H}^1(\Omega)$, minimize

$$E(u) := e(u) - (f,u)_\Omega \quad \text{such that,} \tag{12}$$

$$(u,\mu)_{\partial\Omega} = (p,\mu)_{\partial\Omega} \quad \forall \mu \in \mathbf{H}^{-1/2}(\partial\Omega). \tag{13}$$

where $e(\cdot)$, $(\cdot,\cdot)_\Omega$, and $(\cdot,\cdot)_{\partial\Omega}$ are as in (7).

We discretize the energy (12) exactly as in the Neumann case, so the only difference comes in discretizing the constraints (13). We proceed by selecting a finite-dimensional subspace (the discrete Lagrange multiplier space) $\mathbf{\Lambda}^h \subset \mathbf{H}^{-1/2}(\partial\Omega)$, and enforce (13) for all $\mu^h \in \mathbf{\Lambda}^h$. Not all plausible choices of $\mathbf{\Lambda}^h$ will yield an acceptably accurate approximation, as, in general, $(\mathbf{\Lambda}^h, \mathbf{V}^h)$ must satisfy an inf-sup stability criterion to retain the optimal convergence rates of the approximation spaces [78]. One possible choice for $\mathbf{\Lambda}^h$, which we shall refer to as Λ_1^h and is used in, for instance, [62,69] defines μ^h as piecewise constant over each Cartesian grid cell c_i intersecting the boundary $\partial\Omega$ (see Fig. 6). In other words, we define $\mu^h \in \Lambda_1^h$ as

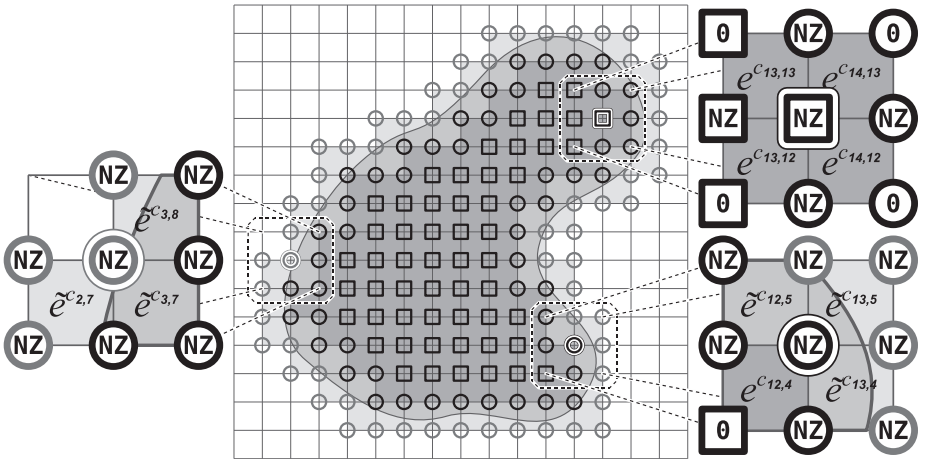
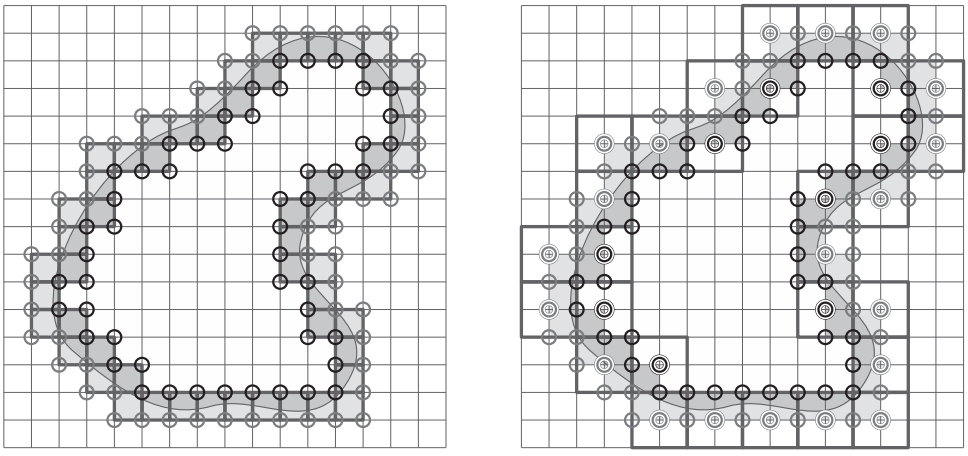


Fig. 5. Illustration in 2 dimensions of the stiffness matrix (A) stencils for various grid vertices. The stencil for a degree of freedom indicates where the nonzero (NZ) entries are of the row (or column) in A corresponding to the degree of freedom. Squared grid vertices have the standard finite difference Poisson stencil (a 5-point stencil in 2 dimensions; a 7-point stencil in 3 dimensions), which naturally arises through the use of e^{c_k} to discretize the energy (7). Circled grid vertices (both black and gray) will generally have a denser stencil (up to a 9-point stencil in 2 dimensions; up to a 27-point stencil in 3 dimensions), due to the use of $\tilde{e}^{c_k}$.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Schematic of functions in Λ_1^h, with single-wide constraints C_1

(b) Schematic of functions in Λ_2^h, with double-wide constraints C_2

Fig. 6. Schematics of two discretizations Λ^h of the Lagrange multiplier space $\mathbf{H}^{-1/2}(\partial\Omega)$ in 2 dimensions used in (13). (a) Shows a schematic of functions in Λ_1^h, which are piecewise constant over $\mathcal{C}_{\partial\Omega}^h \cap \partial\Omega$. (b) Shows a schematic of functions in Λ_2^h, which are piecewise constant over $\mathcal{C}_{\partial\Omega}^{2h} \cap \partial\Omega$ (using the doubly-coarse grid $\mathcal{G}^{2h}$). Note that the center grid vertex (highlighted) in each doubly-coarse boundary grid cell is an independent degree of freedom with respect to C_2, the constraints induced by Λ_2^h. That is, the center grid vertex in a doubly-coarse boundary grid cell participates only in the constraint corresponding to that cell.

$$\mu^h(\mathbf{x}) := \sum_{c_i \in \mathcal{C}_{\partial\Omega}^h} \mu_i \chi_{c_i \cap \partial\Omega}(\mathbf{x}),$$

where the characteristic functions $\chi_{c_i \cap \partial\Omega}$ are given by

$$\chi_{c_i \cap \partial\Omega}(\mathbf{x}) := \begin{cases} 1, & \mathbf{x} \in c_i \cap \partial\Omega, \\ 0, & \mathbf{x} \notin c_i \cap \partial\Omega, \end{cases}. \tag{14}$$

With this choice of discrete Langrange multiplier space, satisfying (13) for all $\mu^h \in \Lambda^h = \Lambda_1^h$ yields a system of sparse linear constraints $B\vec{u} = \vec{p}$ on the coefficient vector $\vec{u}$ of the approximate solution u^h. Each row of the matrix B corresponds to a cell $c_i \in \mathcal{C}_{\partial\Omega}^h$ and enforces the condition

$$\int_{c_i \cap \partial\Omega} u^h(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \int_{c_i \cap \partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{15}$$

Therefore, if $\mathcal{C}_{\partial\Omega}^h = \{c_1, \ldots c, c_m\}$ and $\vec{u} \in \mathbb{R}^n$, then $\vec{p} \in \mathbb{R}^m$, $B \in \mathbb{R}^{m \times n}$, and

$$B_{ij} := \int_{c_i \cap \partial\Omega} N_j(\mathbf{x}) d\mathbf{S}(\mathbf{x}) \tag{16}$$

for each Cartesian trilinear basis function $N_j(\mathbf{x})$. Since only 8 of these basis functions are supported over a given $c_i \cap \partial\Omega$, each row of B contains precisely 8 nonzero entries. The corresponding entry in $\vec{p}$ is

$$p_i := \int_{c_i \cap \partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{17}$$

As before, we evaluate these integrals as described in Section 3.1 (using a suitable polynomial approximation for $p(\mathbf{x})$ in each grid cell or a suitable quadrature rule to evaluate (17)). Discretizing (12) and (13) thus gives rise to the quadratic program:

$$\begin{aligned} &\text{minimize over } \vec{u} \in \mathbb{R}^n \\ &E^h(u^h) := e(u^h) - (f, u^h)_\Omega := \frac{1}{2}\vec{u}^t A\vec{u} - \vec{f}^t\vec{u} \\ &\text{subject to } B\vec{u} = \vec{p}. \end{aligned} \tag{18}$$

The matrix A is exactly as for the embedded Neumann case described in Section 3.2, as is the vector $\vec{f}$ excepting the contribution of the Neumann constraint q (see (10)). This minimization problem may equivalently be expressed as a saddle point system, introducing a Lagrange multiplier $\vec{\lambda}$:

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{p} \end{pmatrix}. \tag{19}$$

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

3.3.1. Null space method and fundamental basis of constraint system

As is done in [1], we solve (18)/(19) using a null space method, which efficiently transforms our problem into a symmetric positive definite linear system. This affords us a wide variety of solution techniques, including black-box solvers such as (preconditioned) conjugate gradient; and a large class of preconditioners, such as incomplete Cholesky (which we use for many of the numerical examples in Section 5). This derived symmetric positive definite system also readily lends itself as a basis for a multigrid smoother such as Gauss–Seidel (as presented in Section 4). For these reasons, our null space approach has significant advantages over alternative approaches such as Schur's complement reduction, direct methods applied to the saddle point system (19), stationary methods such as Uzawa's method, penalty methods, or Krylov methods applied to (19). Those aforementioned approaches which are iterative typically require solving a linear system at each iteration and/or have slow convergence properties. Direct methods tend to be too computationally expensive and memory intensive when applied to large systems. Preconditioning saddle point systems such as (19) is much less well-developed than preconditioning symmetric positive definite systems; hence, applying a Krylov method to (19) is much less appealing than applying a Krylov method to an equivalent symmetric positive definite system. For a more complete survey of the advantages and disadvantages of these and other approaches, see [89].

The null space method requires the construction of a matrix Z whose columns span the null space of B and a vector $\vec{c} \in \mathbb{R}^n$ satisfying the discretized constraints (i.e., $B\vec{c} = \vec{p}$). Our solution $\vec{u}$ to (18) or (19) may then be expressed as $\vec{u} = \vec{c} + Z\vec{v}$ for some $\vec{v}$, and substituting this expression for u into (19) (and eliminating $\vec{\lambda}$ via left multiplication by Z^t) yields the system $Z^tAZ\vec{v} = Z^t(\vec{f} - A\vec{c})$ for $\vec{v}$. As noted in Section 3.2, the null space of A is spanned by the vector $(1, 1, \ldots, 1)^t \in \mathbb{R}^n$, and the entries of B are all non-negative, so $\ker(A) \cap \ker(B) = \{\vec{0}\}$. Therefore, Z^tAZ is non-singular and, specifically, symmetric positive definite. We have thus transformed (18)/(19) into a symmetric positive definite system for $\vec{v}$. We obtain $\vec{u}$ by setting $\vec{u} = \vec{c} + Z\vec{v}$.

We now address the determination of Z. Obtaining Z through a QR factorization or a SVD is likely to be computationally expensive and, moreover, produce a dense Z. A *fundamental basis* presents an alternative to numerical factorization [89]. The matrix B is full rank if and only if an ordering of the degrees of freedom exists so that B may be expressed as $B = (B_m|B_{n-m})$ for some $m \times m$ non-singular matrix B_m. Any such ordering gives the corresponding fundamental basis

$$Z = \begin{pmatrix} -B_m^{-1}B_{n-m} \\ I_{n-m} \end{pmatrix}. \tag{20}$$

Clearly, $BZ = 0$ and the vector $\vec{c} = \begin{pmatrix} B_m^{-1}\vec{p} \\ 0 \end{pmatrix}$ satisfies $B\vec{c} = \vec{p}$. Therefore, if we can solve systems of the form

$$B_m\vec{x} = \vec{d}, \tag{21}$$

efficiently, we can store the factors B_m, B_{n-m}, and A sparsely and compute the action of Z^tAZ readily (e.g., for use in conjugate gradient). Note that, regardless of the choice of B_m, the symmetric positive definite stencil defined by Z^tAZ coincides with the standard 7-point stencil for all degrees of freedom sufficiently far from the boundary.

3.3.2. Aggregation of single-wide constraints

Unfortunately, as discussed in [1], the choice of Λ_1^h (the space of functions that are piecewise constant over each boundary grid cell) as the discrete Lagrange multiplier space approximating $\mathbf{H}^{-1/2}(\partial\Omega)$ makes it difficult (if not impossible) to determine an ordering of the degrees of freedom that gives a well-conditioned and easily invertible B_m. Bedrossian et al. [1] gives an ordering of the degrees of freedom and of the constraints that yields an upper-triangular B_m; however, although the resulting system (21) can theoretically be efficiently solved by back-substitution, in practice such a solution procedure introduces prohibitively large numerical errors for anything but the smallest grids.

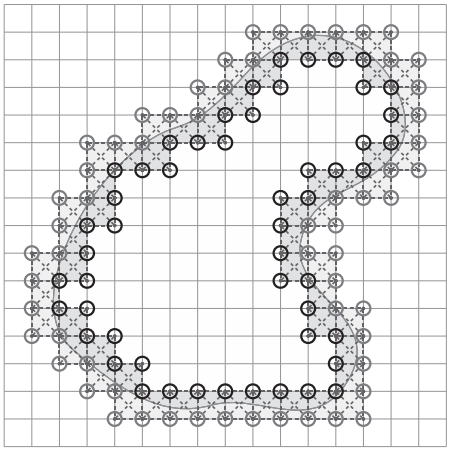
As in [1], we remedy this by using an alternative approximation to $\mathbf{H}^{-1/2}(\partial\Omega)$ that induces a different set of linear constraints. To motivate our approach, suppose we define a set of m linear constraints (other than those induced by Λ_1^h) such that each constraint contains an *independent* degree of freedom, a degree of freedom which participates only in that one constraint. Observe, then, that ordering these m independent degrees of freedom first, in matching order with their associated constraints, yields a *diagonal* B_m, which is trivial to invert. As the constraints induced by Λ_1^h generally have an insufficient number of independent degrees of freedom, we thus aim to manufacture an alternative discrete Lagrange multiplier space such that the induced set of constraints admits such a set of independent degrees of freedom that gives a diagonal B_m. For example, Bedrossian et al. [1] uses $\Lambda_1^{2h} =: \Lambda_2^h$ (the set of scalar piecewise constant functions over the cells of the doubly-coarse grid $\mathcal{G}^{2h}$; see Fig. 6) as an approximation to $\mathbf{H}^{-1/2}(\partial\Omega)$, leading to what may be described as *double-wide constraints*. Each double-wide constraint encompasses a 2×2 (in 2 dimensions) or $2 \times 2 \times 2$ (in 3 dimensions) block of cells. The center vertex in such a block of cells always participates only in that one constraint, hence these center vertices correspond to independent degrees of freedom. Double-wide constraints are acceptable for problems in 2 dimensions, as investigated by Bedrossian et al. [1]; however, the structural rigidity of Λ_2^h presents conditioning issues in 3 dimensions (see Appendix C for a specific example). One of our major contributions is a more general, flexible approach toward constructing constraints which gives greater control on conditioning, and for which the double-wide constraints induced by Λ_2^h will be a special case.

The key idea is that rather than first defining the set of constraints and then selecting an independent degree of freedom from each constraint, we will first select the set of independent degrees of freedom and then subsequently build a single constraint equation around each independent degree of freedom. To this end, let C_1 denote the set of *single-wide constraints* (15) induced by Λ_1^h, as described above; and let G denote the adjacency graph induced by C_1, as depicted in Fig. 7(a). That is,

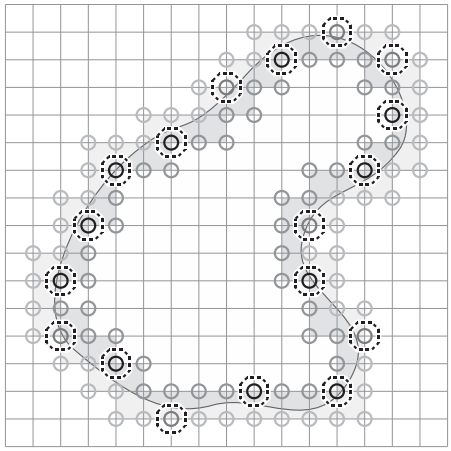
Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

two degrees of freedom are *adjacent* in G if they simultaneously participate in some single-wide constraint; or, in more geometric terms, two grid vertices are adjacent in G if they share a common incident boundary grid cell. Choose $m_a < m$ degrees of freedom which constitute an *independent set* $\mathcal{I}$ with respect to G. In other words, no two degrees of freedom in $\mathcal{I}$ will simultaneously participate in the same single-wide constraint. An example of such an independent set is given in Fig. 7(b). Now associate each of the m single-wide constraints in C_1 to one of these independent degrees of freedom in $\mathcal{I}$, with the provision that, if a constraint contains an independent degree of freedom, it must be associated with that independent degree of freedom. (This latter requirement is conflict-free, as any single-wide constraint in C_1 will contain at most one independent degree of freedom, by construction.) Thus, for those single-wide constraints containing an independent degree of freedom, this association is precisely determined. However, some single-wide constraints will contain no independent degree of freedom, so some additional heuristic must be used to determine this association. See Figs. 7(c) and (d) for an example association of each single-wide constraint to an independent degree of freedom.

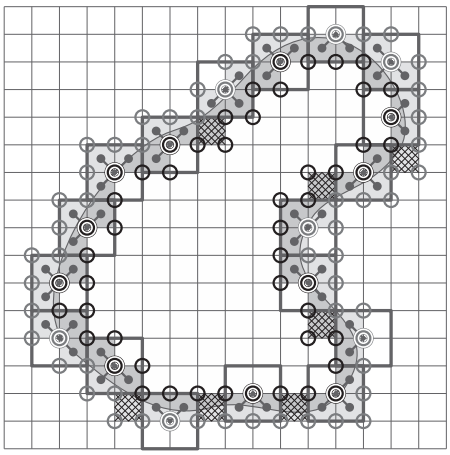
Let $\mathcal{I} = \{d_1, \ldots c, d_{m_a}\}$ denote the independent set of degrees of freedom; and let $C_{d_i} \subset C_1$ denote the set of single-wide constraints associated with independent degree of freedom d_i, such that $\bigsqcup_i C_{d_i} = C_1$. We then form the following m_a *aggregate constraint* equations:



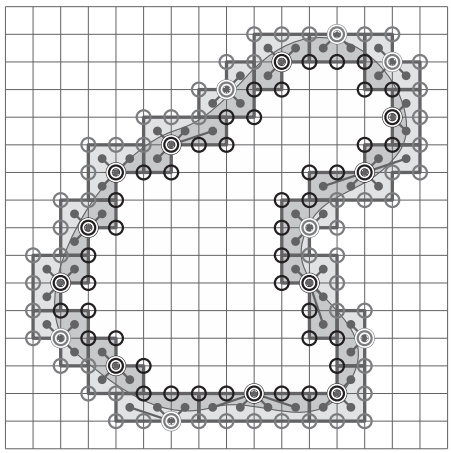
(a) Adjacency graph G induced by the set of single-wide constraints C_1. Boundary grid vertices are adjacent with respect to G if they share a common incident boundary grid cell.



(b) Example selection of an independent set $\mathcal{I}$ of degrees of freedom. No two independent boundary grid vertices share a common incident boundary grid cell; equivalently, no two independent degrees of freedom simultaneously participate in the same single-wide constraint.



(c) Associating single-wide constraints to a participating independent degree of freedom. Some constraints (identified by cross-hatching) contain no independent degree of freedom; one must resort to some additional heuristic to associate these constraints.



(d) Associating the remaining single-wide constraints to a nearby independent degree of freedom (using some implementation-defined heuristic) and the final set of aggregate constraints.

Fig. 7. Illustrated progression of the constraint aggregation described in Section 3.3.2.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

$$\sum_{c_k \in C_{d_i}} \int_{c_k} u^h(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \sum_{c_k \in C_{d_i}} \int_{c_k} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}), \tag{22}$$

where $c_k \in C_{d_i}$ denotes that cell c_k corresponds to a single-wide constraint associated with independent degree of freedom d_i. Effectively, the single-wide constraint equations in C_1 associated to a given independent degree of freedom are summed into a single aggregate constraint equation. Likewise, the corresponding discrete Lagrange multiplier space Λ_a^h is spanned by sums of the basis functions $\chi_{c_k \cap \partial\Omega}$ of Λ_1^h from (14):

$$\mu^h(\mathbf{x}) := \sum_{d_i \in \mathcal{I}} \mu_i \sum_{c_k \in C_{d_i}} \chi_{c_k \cap \partial\Omega}.$$

Now let B and $\vec{p}$ denote the matrix and right-hand side of the system of aggregate constraints (22):

$$B_{ij} := \sum_{c_k \in C_{d_i}} \int_{c_k \cap \partial\Omega} N_j(\mathbf{x}) d\mathbf{S}(\mathbf{x}), \quad p_i := \sum_{c_k \in C_{d_i}} \int_{c_k \cap \partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{23}$$

Clearly, by construction, this set of aggregate constraints admits an ordering of the degrees of freedom to give a diagonal B_{m_a}: just order the independent degrees of freedom first.

In summary, the above procedure aggregates the single-wide constraints C_1 to yield an alternative set of constraints C_a which admits an ordering of the degrees of freedom to give a diagonal B_{m_a}. We have thus far described this constraint aggregation in very general terms, and there indeed remains a great deal of flexibility, particularly in how one chooses the set of independent degrees of freedom. For example, selecting all degrees of freedom which exist in the doubly-coarse grid $\mathcal{G}^{2h}$ as independent degrees of freedom leads to the double-wide constraints C_2 mentioned earlier. For simplicity, in the following discussion, we consider only strategies which select independent degrees of freedom one at a time and greedily, noting that alternative approaches could very well yield equal or superior results. Such a constraint aggregation implementation may be described by the following parameters.

- One should decide how the degrees of freedom should be ordered or prioritized for consideration for inclusion in the independent set.
- We need some condition on which to terminate the further selection of independent degrees of freedom.
- Once we have selected the set of independent degrees of freedom, we must associate an independent degree of freedom to each otherwise unassociated single-wide constraint (a constraint containing no independent degree of freedom).

For purposes of selecting independent degrees of freedom, we found that weighting degrees of freedom by the sum of the their coefficients across all single-wide constraints (i.e., the weight of the jth degree of freedom is $\sum_i B_{ij}$) gives good results. Thus, in each iteration, we select, for inclusion in the independent set, the degree of freedom with the largest weight, taking care to exclude degrees of freedom adjacent to previously selected independent degrees of freedom. The motivation for using $\sum_i B_{ij}$ as the weight for the jth degree of freedom is an attempt to maximize the diagonal entries in B_{m_a} and ultimately improve the conditioning of the Z^tAZ system. An alternative weighting that seemed to give acceptable results was $\max_i B_{ij}$. We found that additionally limiting the independent degrees of freedom to *only virtual* degrees of freedom resulted in a vastly more efficient boundary smoother in our multigrid algorithm; see Section 4.

Now, given a degree of freedom weighting scheme like above, one may freeze the independent set once all remaining eligible degrees of freedom (those not adjacent to previously selected independent degrees of freedom) have a weight below some threshold. Alternatively, one may freeze the independent set once all the subsequently induced aggregate constraints (given the current set of independent degrees of freedom and some grid-cell-to-independent-degree-of-freedom association heuristic) satisfy some geometric bound. For example, one may terminate the further selection of independent degrees of freedom once the current set of independent degrees of freedom induces a set of aggregate constraints which each lie within a $4 \times 4 \times 4$ block of grid cells centered on the corresponding independent degree of freedom.

Finally, to minimize the geometric extent of the aggregate constraints, we associate an otherwise unassociated single-wide constraint to the geometrically closest independent degree of freedom, breaking ties by preferring higher-weighted degrees of freedom.

Algorithm 1 outlines an example implementation of the constraint aggregation algorithm described above. We followed this specific implementation of the constraint aggregation algorithm for the numerical examples given in Section 5.3. In this implementation, we select an independent set of virtual degrees of freedom prioritized by the sum of their associated coefficients over all single-wide constraints; and we terminate the further selection of independent degrees of freedom once all boundary grid cells are within some $4 \times 4 \times 4$ block of grid cells centered on an independent degree of freedom (Fig. 8 explains this termination condition graphically). Together with the rule associating single-wide constraints to the geometrically closest independent degree of freedom, this termination condition ensures that all aggregate constraints fit within a $4 \times 4 \times 4$ block of grid cells centered on an independent degree of freedom, thus limiting the geometric extent of an aggregate constraint.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

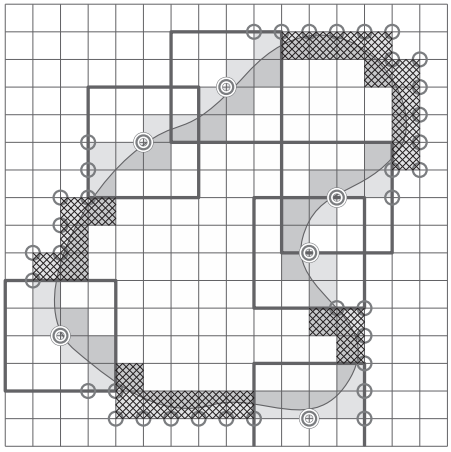


Fig. 8. A graphical representation (in 2 dimensions) of a plausible state of Algorithm 1 after the selection of 6 independent degrees of freedom (highlighted). Some degrees of freedom have been removed to indicate their ineligibility as subsequently selected independent degrees of freedom: material degrees of freedom, by definition of Algorithm 1, are never selected as independent degrees of freedom (this vastly improved the performance of our boundary smoother in our multigrid algorithm; see Section 4); and those virtual degrees of freedom adjacent to one of the 6 previously selected independent degrees cannot now be selected as independent degrees of freedom, simply by the definition of independence. Further, we distinguish between *covered* boundary grid cells, which lie within some 4×4 block of cells (shown as the dark gray outlined squares) around an independent degree of freedom; and the remaining *uncovered* boundary grid cells (denoted by cross-hatching). Once all boundary grid cells are covered, Algorithm 1 terminates further selection of independent degrees of freedom.

Algorithm 1. Constraint aggregration algorithm for embedded Dirichlet discretizations.

Reorder the degrees of freedom such that virtual degrees of freedom (VDOFs) are enumerated first and $w_1 > w_2 > \ldots m$, where $w_j = \sum_i B_{ij}$ for VDOF j and B_{ij} is as in (16).
let $\mathcal{I} \leftarrow \emptyset$ {$\mathcal{I}$ denotes the set of independent degrees of freedom (IDOFs)}
{only iterate over VDOFs}
for $j = 1, 2, \ldots,$ **do**
  {Use an acceleration structure (e.g., an explicit set or bit set data structure) to make the following query efficient.}
  if VDOF j is adjacent to some IDOF in $\mathcal{I}$ **then**
    continue
  end if
  $\mathcal{I} \leftarrow \mathcal{I} \sqcup \{j\}$ {add VDOF j to the set of IDOFs}
  {Use an acceleration structure (e.g., an associative array data structure) to make the following query efficient.}
  if each boundary grid cell is within some $4 \times 4 \times 4$ block of grid cells centered on an IDOF in $\mathcal{I}$ (see Fig. 8)
  then
    break
  end if
end for
Associate each boundary grid cell to the geometrically closest IDOF in $\mathcal{I}$, breaking ties by preferring IDOFs with higher weights (w_j). Let C_j denote the set of boundary grid cells associated to IDOF j.
for all $j \in \mathcal{I}$ **do**
  Sum the single-wide constraint equations associated with the boundary grid cells in C_j to form a new aggregate constraint equation.
end for

We conclude this section with some remarks regarding the discrete Lagrange multiplier space $\mathbf{\Lambda}^h$. Generally speaking, using a richer discrete Lagrange multiplier space (one that better approximates $\mathbf{H}^{-1/2}(\partial\Omega)$) results in a smaller error in the approximate solution u^h. Within the context of single-wide constraint aggregation, roughly speaking, one can increase the richness of $\mathbf{\Lambda}_a^h$ (the discrete Lagrange multiplier space associated with the aggregate constraints) by choosing more independent degrees of freedom. In some sense, then, the discrete Lagrange multiplier space $\mathbf{\Lambda}_2^h$ associated with the double-wide constraints represents the richest possible discrete Lagrange multiplier space one may obtain within this constraint aggregation framework, as its set of independent degrees of freedom is maximal. However, as shown in Appendix C, use of double-wide constraints leads to a relatively poorly conditioned $Z^T A Z$ system in 3 dimensions, and this behavior is

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

characteristic of selecting too many independent degrees of freedom, some of which may be poorly supported and lead to poor conditioning. We feel that our criterion in Algorithm 1 to terminate further selection of independent degrees of freedom strikes a balance between maintaining second order accuracy and ensuring reasonable conditioning in the Z^TAZ system.

In addition to the relationship among the richness of $\mathbf{\Lambda}^h$, the error in the approximate solution u^h, and (for $\mathbf{\Lambda}_a^h$ in particular) the conditioning of the Z^TAZ system, it is also necessary, in order to obtain optimal convergence rates, for $\mathbf{\Lambda}^h$ and the approximation space to $\mathbf{H}^1(\Omega)$, $\mathbf{V}^h$, to satisfy an inf-sup stability condition uniformly in grid resolution [78]. This ultimately has the effect of limiting the richness of $\mathbf{\Lambda}^h$. Fortunately, based primarily on numerical evidence (see, for example [62,69]), it is generally accepted that the pairing $\left(\mathbf{V}^h, \mathbf{\Lambda}_1^h\right)$ satisfies an inf-sup stability condition, where we use the discrete Lagrange multiplier space $\mathbf{\Lambda}_1^h$ associated with the single-wide constraints. More explicitly, we assume the existence of γ_0, $h_0 > 0$ such that, for all $h \in (0, h_0]$,

$$\inf_{\mu^h \in \Lambda_1^h} \sup_{v^h \in \mathbf{V}^h} \alpha(\mu^h, v^h) \geqslant \gamma_0,$$

where $\alpha : \mathbf{H}^{-1/2}(\partial\Omega) \times \mathbf{H}^1(\Omega) \to \mathbb{R}$ is

$$\alpha(\mu^h, v^h) := \frac{(\mu^h, Tv^h)_{\partial\Omega}}{\|\mu^h\|_{-1/2,\partial\Omega}\|v^h\|_{1,\Omega}}$$

and $T : \mathbf{H}^1(\Omega) \to L^2(\partial\Omega)$ is the trace operator on Ω. Now if $\mathbf{\Lambda}_a^h$ is the discrete Lagrange multiplier space associated with any set of aggregate constraints, then $\mathbf{\Lambda}_a^h$ is a *subspace* of $\mathbf{\Lambda}_1^h$, hence

$$\inf_{\mu^h \in \Lambda_a^h} \sup_{v^h \in \mathbf{V}^h} \alpha(\mu^h, v^h) \geqslant \inf_{\mu^h \in \Lambda_1^h} \sup_{v^h \in \mathbf{V}^h} \alpha(\mu^h, v^h) \geqslant \gamma_0,$$

and we see that $\left(\mathbf{V}^h, \mathbf{\Lambda}_a^h\right)$ satisfies an inf-sup stability condition as well. (The same argument is used in [1] to show that, specifically, $\left(\mathbf{V}^h, \mathbf{\Lambda}_2^h\right)$ is inf-sup stable.) Generally speaking, if $(\mathbf{V}^h, \mathbf{\Lambda}^h)$ satisfies an inf-sup stability condition, then pairing $\mathbf{V}^h$ with any coarsening (i.e., subspace) of $\mathbf{\Lambda}^h$ will be inf-sup stable as well.

3.4. Embedded interface

To handle the full elliptic interface problem (1)–(3), we combine our embedded Neumann and embedded Dirichlet approaches in a straightforward way. We consider the equivalent minimization form of the problem (1)–(3):

$$\begin{aligned}
&\text{over all } u \in \mathbf{V} := \{u : u^{\pm} \in \mathbf{H}^1(\Omega^{\pm})\}, \text{ minimize} \\
&E(u) := e(u) - (f, u)_{\Omega} - (b, \bar{u})_{\Gamma} := \int_{\Omega^+ \sqcup \Omega^-} \frac{1}{2}\nabla u \cdot \beta \nabla u d\mathbf{x} - \int_{\Omega} f u d\mathbf{x} - \int_{\Gamma} b\bar{u} d\mathbf{S}(\mathbf{x}), \qquad (24) \\
&\text{such that} \quad ([u], \mu)_{\Gamma} = (a, \mu)_{\Gamma} \quad \forall \mu \in \mathbf{H}^{-1/2}(\Gamma). \qquad (25)
\end{aligned}$$

Here $\bar{u}(\mathbf{x})|_{\Gamma} = (u^+ + u^-)/2$. As before, we define discretizations of $\mathbf{V}$ and $\mathbf{H}^{-1/2}(\Gamma)$ and then construct the resulting discrete saddle point problem. To define $\mathbf{V}^h \subset \mathbf{V}$, we separately discretize $\mathbf{H}^1(\Omega^+)$ and $\mathbf{H}^1(\Omega^-)$ using the same virtual node representation used to discretize the embedded Neumann and embedded Dirichlet problems, employing the duplicated grid described in Section 3.1 and depicted in Fig. 3. This discretization yields the block diagonal stiffness matrix for the interface problem,

$$A = \begin{pmatrix} A^+ & 0 \\ 0 & A^- \end{pmatrix}, \qquad (26)$$

where A^+ is the stiffness matrix associated with the embedded Neumann problem on Ω^+ and A^- is the stiffness matrix associated with the embedded Neumann problem on Ω^-, as described in Section 3.2.

As for the embedded Dirichlet problem, we first discretize the continuous constraint equations (25) via $\mathbf{\Lambda}_1^h$ into single-wide constraint equations,

$$\int_{c_k \cap \Gamma} [u^h] d\mathbf{S}(\mathbf{x}) = \int_{c_k \cap \Gamma} a d\mathbf{S}(\mathbf{x}), \qquad (27)$$

and then aggregate these single-wide constraints (27), as described in Section 3.3.2:

$$\sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} [u^h] d\mathbf{S}(\mathbf{x}) = \sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} a d\mathbf{S}(\mathbf{x}). \qquad (28)$$

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

Note that we described the constraint aggregation procedure in Section 3.3.2 with Dirichlet constraints in mind, but aggregating single-wide interface constraints is completely analogous. Regarding the specific implementation in Algorithm 1, one would use the weights $w_j = \left|\sum_i B_{ij}\right|$ to account for negative single-wide constraint coefficients for interior degrees of freedom.

Using the aggregate constraints in (28) results in the block interface constraint matrix $B = (B^+ | -B^-)$, where B^+,B^- are, respectively, the constraint matrices associated with the embedded Dirichlet problems on the exterior and interior of the interface. In other words,

$$B_{ij} = \text{sign}(j) \sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} N_j d\mathbf{S}(\mathbf{x}), \tag{29}$$

where $\text{sign}(j) := +1$ if the j^{th} degree of freedom is associated with $u^{h,+}$ and $\text{sign}(j) := -1$ if the jth degree of freedom is associated with $u^{h,-}$. These discretization choices give the saddle point problem

$$\begin{pmatrix} A^+ & 0 & (B^+)^t \\ 0 & A^- & (-B^-)^t \\ B^+ & -B^- & 0 \end{pmatrix} \begin{pmatrix} \vec{u}^+ \\ \vec{u}^- \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f}^+ \\ \vec{f}^- \\ \vec{a} \end{pmatrix}, \tag{30}$$

where $\vec{u}^+$ contains the degrees of freedom associated with the exterior discretization and $\vec{u}^-$ contains the degrees of freedom associated with the interior discretization. We once again solve this saddle point system using the null space method described in Section 3.3.1 by ordering the independent degrees of freedom first to obtain a diagonal B_{m_a}. Observe that we may restrict independent degrees of freedom to only virtual degrees of freedom, as every material degree of freedom has a geometrically co-located virtual degree of freedom that is indistiguishable as far as adjacency and weight (up to a sign change) is concerned. We have found that such a restriction results in a better-conditioned system. Contrast this observation with the Dirichlet case, where each material degree of freedom does *not* have an equivalent (as far as the constraint system is concerned) virtual degree of freedom, and hence the decision to allow or disallow the selection of material degrees of freedom as independent degrees of freedom has a much bigger impact on the final set of aggregate constraints.

3.4.1. Discontinuity removal

In general, our proposed method requires the solution of the symmetric positive definite system Z^tAZ. However, if the coefficient β is smooth, the IIM and similar methods achieve uniform second order accuracy without altering the standard Poisson finite difference stencil (the 5-point stencil in 2 dimensions or the 7-point stencil in 3 dimensions). In this section, we demonstrate how the virtual node framework similarly allows the use of the standard Poisson stencil when β is smooth.

Suppose $d(\mathbf{x}) \in \mathbf{V}$ is constructed to satisfy the jump conditions (2), (3) and $u(\mathbf{x})$ is the exact solution. Then since $[\beta] = 0$, the difference $w(\mathbf{x}) := u(\mathbf{x}) - d(\mathbf{x})$ satisfies $\beta[\nabla w \cdot \hat{\mathbf{n}}] = [\beta \nabla w \cdot \hat{\mathbf{n}}] = 0$ and $[w] = 0$. Since w satisfies homogeneous jump conditions $[\nabla w \cdot \hat{\mathbf{n}}] = 0$ and $[w] = 0$, we do not require virtual degrees of freedom to capture any discontinuities across Γ. In this manner, solving for w presents an appealing alternative as the presence of virtual nodes no longer adversely affects the subsequent linear algebra problem. Therefore, when $[\beta] = 0$ we recover an approximation to (2) and (3) by separately discretizing w and d and then setting $u = w + d$.

We discretize w over the unduplicated grid $\mathcal{G}^h$ using $\mathbf{H}^1(\Omega)$ Cartesian piecewise trilinear elements. Consequently, if the grid $\mathcal{G}^h$ contains r material degrees of freedom, then $\vec{w} \in \mathbb{R}^r$ contains the coefficients in terms of the trilinear basis. We discretize u and d using the full virtual node basis $\mathbf{V}^h$ as they possess lower regularity across Γ. With these choices, we can represent the coefficient vector $\vec{u} \in \mathbb{R}^n$ $(n > r)$ of the approximate solution u^h in the basis of $\mathbf{V}^h$ as $\vec{u} = \vec{d} + T\vec{w}$, where the matrix $T \in \mathbb{R}^{n \times r}$ is an embedding of the trilinear basis into the virtual node basis. We define this transformation by a simple identification of virtual and material nodes, as a function $v^h \in \mathbf{V}^h$ satisfies homogeneous jump conditions if and only if the value of the function v^h at a virtual node equals its value at the geometrically co-located material node. Thus, T maps the value at a given vertex in the unduplicated grid to each of its copies, material or virtual, in the duplicated grid. To be a little more explicit, assume that we order the degrees of freedom such that

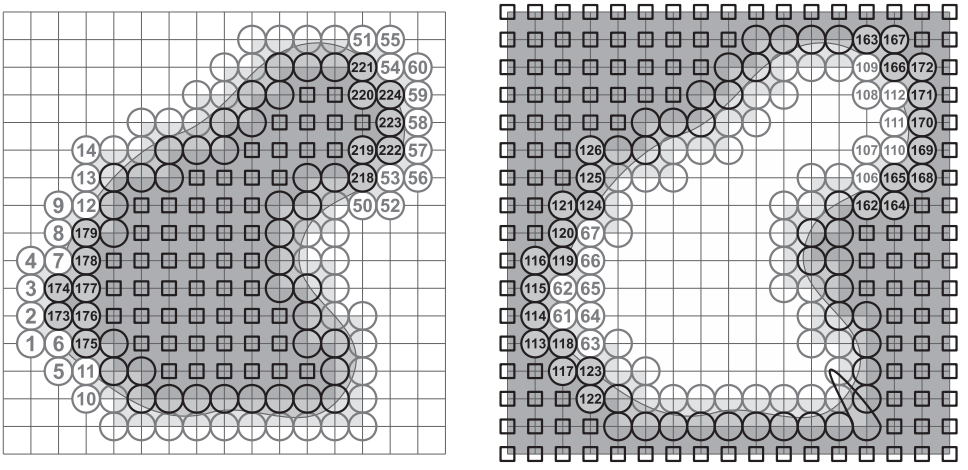
$$\vec{u} = (u_1, u_2, \ldots c, u_s, u_{s+1}, u_{s+2}, \ldots c, u_{2s}, u_{2s+1}, \ldots c, u_n)^t.$$

Here, $\{u_k\}_{k=1}^s$ represent the $s := n - r$ coefficients of the virtual degrees of freedom; $\{u_{s+k}\}_{k=1}^s$ represent the coefficients of the material degrees of freedom respectively co-located with $\{u_k\}_{k=1}^s$; and the remaining coefficients $\{u_k\}_{k=2s+1}^n$ correspond to degrees of freedom lying outside any interfacial grid cells. See Fig. 9 for an illustration of this ordering. Then T would take the form

$$T = \begin{pmatrix} I_s & 0 \\ I_s & 0 \\ 0 & I_{n-2s} \end{pmatrix}. \tag{31}$$

Regardless of the ordering of the degrees of freedom, each column of T corresponds to a material node in the grid and each row of T corresponds to either a material node or a virtual node. The column of T corresponding to material node j has a 1 in the row corresponding to material node j; a 1 in the row corresponding to j's geometrically co-located virtual node, if it exists (e.g., one of the first s columns in (31) above); and zeros everywhere else.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Interior discretization (b) Exterior discretization

Fig. 9. Example enumeration of the interfacial degrees of freedom (circled) such that T has the representation (31). Only the indices of a few select interfacial degrees of freedom are shown. Here, we enumerate the $s = 112$ virtual degrees of freedom lexicographically, beginning with the interior discretization. The interior discretization has 60 virtual degrees of freedom (indexed 1–60) and 52 interfacial material degrees of freedom (indexed 173–224); likewise, the exterior discretization has 52 virtual degrees of freedom (indexed 61–112) and 60 interfacial material degrees of freedom (indexed 113–172). Notice how the index to an interfacial material degree of freedom is offset from the index of its co-located virtual degree of freedom by exactly $s = 112$. The remaining non-interfacial degrees of freedom (squared) are enumerated starting with index $2s + 1 = 225$.

Determining $\vec{w}$ now proceeds in a manner analogous to the null space method used to solve (18): we wish to minimize the energy over all vectors of the form $\vec{u} = \vec{d} + T\vec{w}$. For the sake of discussion, suppose we discretize the energy (24) using the Cartesian trilinear representation everywhere in the domain. Then substituting the expression $\vec{u} = \vec{d} + T\vec{w}$ into the energy (24) gives

$$E^h(\vec{u}) := \frac{1}{2}\vec{u}^t A\vec{u} - \vec{f}^t\vec{u} = \frac{1}{2}\vec{w}^t T^t AT\vec{w} - \vec{f}^t T\vec{w} + \vec{w}^t T^t A\vec{d} + \frac{1}{2}\vec{d}^t A\vec{d} - \vec{f}^t\vec{d},$$

which, in turn, implicitly defines an energy over only the material degrees of freedom $\vec{w} \in \mathbb{R}^r$. Differentiation with respect to w_i thus leads to the linear system

$$T^t AT\vec{w} = T^t\left(\vec{f} - A\vec{d}\right), \quad \vec{u} = \vec{d} + T\vec{w}.$$

It is not hard to show that the matrix T^tAT is a straightforward, trilinear discretization over the material degrees of freedom, i.e., a 27-point second order approximation to the (variable coefficient) Laplacian. Thus, we may replace the T^tAT operator with the standard 7-point Poisson stencil Δ^h_β, only introducing a second order deviation in $\vec{w}$:

$$\Delta^h_\beta\vec{w} = T^t\left(\vec{f} - A\vec{d}\right), \quad \vec{u} = \vec{d} + T\vec{w}. \tag{32}$$

This approach allows the application of efficient black-box solvers for Δ^h_β, and the discontinuity along the interface only enters into the right-hand side of (32).

We now discuss the approximation of d, the particular solution satisfying the jump conditions (2) and (3). Observe that, without loss of generality, we may assume that d is supported only near the interface, as the jump constraints are localized to the interface. Further, we may assume that d vanishes entirely on, say, the exterior region Ω^+, as the jump constraints only involve *differences* between exterior and interior values. This latter assumption allows us to express the jump constraints on d as direct constraints on d^-:

$$-d^- = [d] = a, \quad -\beta\nabla d^- \cdot \hat{\mathbf{n}} = \beta[\nabla d \cdot \hat{\mathbf{n}}] = b.$$

The corresponding discretized single-wide constraints on d^h, the $\mathbf{V}^h$-approximation to d, are thus

$$\int_{c_i\cap\Gamma} d^{h,-}d\mathbf{S}(\mathbf{x}) = -\int_{c_i\cap\Gamma} ad\mathbf{S}(\mathbf{x}), \quad \int_{c_i\cap\Gamma} \beta\nabla d^{h,-}\cdot\hat{\mathbf{n}}d\mathbf{S}(\mathbf{x}) = -\int_{c_i\cap\Gamma} bd\mathbf{S}(\mathbf{x})$$

for each grid cell $c_i \in \mathcal{C}^h_\Gamma$ intersecting the interface Γ; and $d^{h,+} \equiv 0$. This gives a sparse linear system for the coefficient vector $\vec{d}$ of d^h where only interior interfacial degrees of freedom participate:

$$\sum_{j\in\mathcal{N}^{h,-}_{c_i}}\left(\int_{c_i\cap\Gamma} N_j d\mathbf{S}(\mathbf{x})\right)d_j = -\int_{c_i\cap\Gamma} ad\mathbf{S}(\mathbf{x}); \tag{33}$$

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

$$\sum_{j \in \mathcal{N}_{c_i}^{h,-}} \left(\int_{c_i \cap \Gamma} \beta \nabla N_j \cdot \hat{\mathbf{n}} d\mathbf{S}(\mathbf{x}) \right) d_j = - \int_{c_i \cap \Gamma} b d\mathbf{S}(\mathbf{x}); \tag{34}$$

where $\mathcal{N}_{c_i}^{h,-}$ denotes the indices of the 8 interior degrees of freedom geometrically located at the corners of cell c_i. This system has $2m$ rows, where $m = |\mathcal{C}_\Gamma^h|$ is the number of interfacial grid cells; and it has one column for each interior interfacial degree of freedom. Thus, unfortunately, this system will not only be asymmetric, but will generally be overdetermined as well. Hence, one should take some care when computing an approximate solution.

Algorithm 2 gives one approach to constructing $\vec{d}$ which we found works well. The algorithm locally constructs a trilinear function v which approximately satisfies the constraints (33), (34) within a $3 \times 3 \times 3$-cell neighborhood centered on an interfacial grid cell $c_i \in \mathcal{C}_\Gamma^h$. We then evaluate v at the grid vertices of c_i to obtain values for the corresponding entries in $\vec{d}$. This procedure may give an interfacial degree of freedom multiple values, from more than one local construction; we average these values together, as explained below.

We alert the reader to two subtle but important details of Algorithm 2. First, most, if not all, of these local constructions amount to a least-squares solution to a small overdetermined system of linear equations. In order to achieve second order convergence in u, we found it necessary to scale the constraints (34) on $\nabla d^{h,-}$ by $h^{1+\gamma}$ (for some γ between 0 and 1), which places more emphasis on satisfying the constraints on $d^{h,-}$ than on satisfying the constraints on $\nabla d^{h,-}$ in the least-squares solves. We found $\gamma = 1/3$ gave the best convergence rate for Example 5.5 over the range of tested resolutions. We suggest further research is necessary to determine the optimal scaling of the $\nabla d^{h,-}$-constraints in general, both theoretically and empirically.

Second, as mentioned above, more than one local construction may yield a value for $\vec{d}$ at a given interfacial degree of freedom; indeed, the number of such local constructions around a degree of freedom equals the number of incident interfacial grid cells. We compute a final value for $\vec{d}$ at this degree of freedom by taking a *weighted* average of the values yielded by the various local constructions, with weights equal to the surface area of Γ within the interface grid cell around which the local construction is based. Other weightings of the various local construction contributions could very well give equal or better results.

Algorithm 2. Construct an approximate d satisfying (2) and (3)

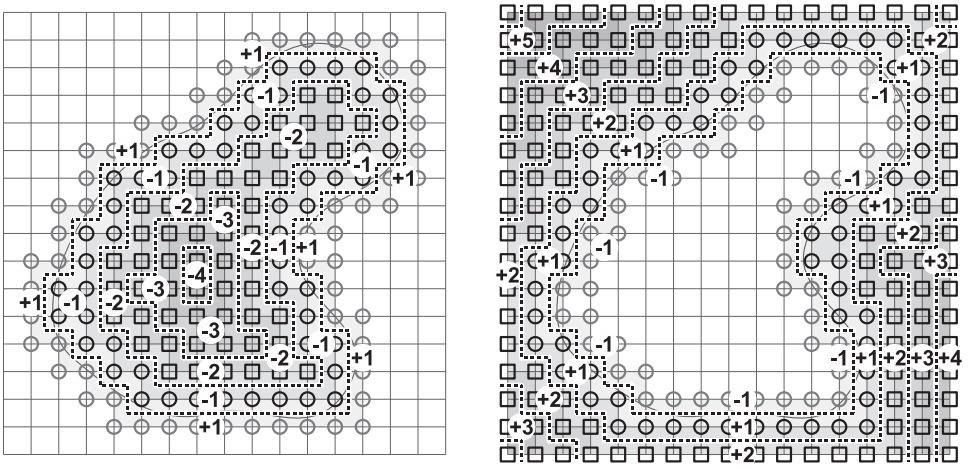
{I and J below denote multi-indices, i.e., triples of linear indices, over the unduplicated grid $\mathcal{G}^h$.}
$\vec{c} \leftarrow \vec{0}$
$w_J \leftarrow 0$ for each interior grid vertex J incident to an interfacial grid cell {the weight sum for degree of freedom J}
for all $c_I \in \mathcal{C}_\Gamma^h$ **do**
  let $S := \left\{ c_J \in \mathcal{C}_\Gamma^h : \|I - J\|_\infty \leqslant 1 \right\}$
  assert ($|S| \leqslant 27$ and $c_I \in S$)
  Construct a trilinear function v (i.e., solve for 8 coefficients) satisfying the constraints (33), (34) on $d^{h,-}$ and $\nabla d^{h,-}$ defined over the cells in S (2 constraints per cell). If $2|S| < 8$, choose v to have minimum 2-norm (for some appropriate 2-norm on the trilinear coefficients); if $2|S| > 8$, choose v to minimize the 2-norm of the residual of the constraint equations after scaling the $\nabla d^{h,-}$-constraint equations by $h^{1+\gamma}$.
  let $w := \int_{c_I \cap \Gamma} dS(\mathbf{x})$ {the local weight for the degrees of freedom at the corners of c_I}
  for $i = 1, \ldots, 8$ **do**
    let J denote the index of the ith grid vertex incident to c_I (say, lexicographically)
    $d_J^- \leftarrow d_J^- + w \cdot v(\mathbf{x}_J)$ {$\mathbf{x}_J$ denotes the spacial coordinates of grid vertex J}
    $w_J \leftarrow w_J + w$
  end for
end for
for all interfacial grid vertex indices J **do**
  $d_J^- \leftarrow d_J^- / w_J$
{average the multiple contributions to the value of d_J^-}
end for

Note that the computational cost of computing $\vec{d}$ in the above fashion is proportional to the number of interfacial degrees of freedom, hence contributes negligibly to the overall cost of computing $\vec{u}$.

4. Multigrid

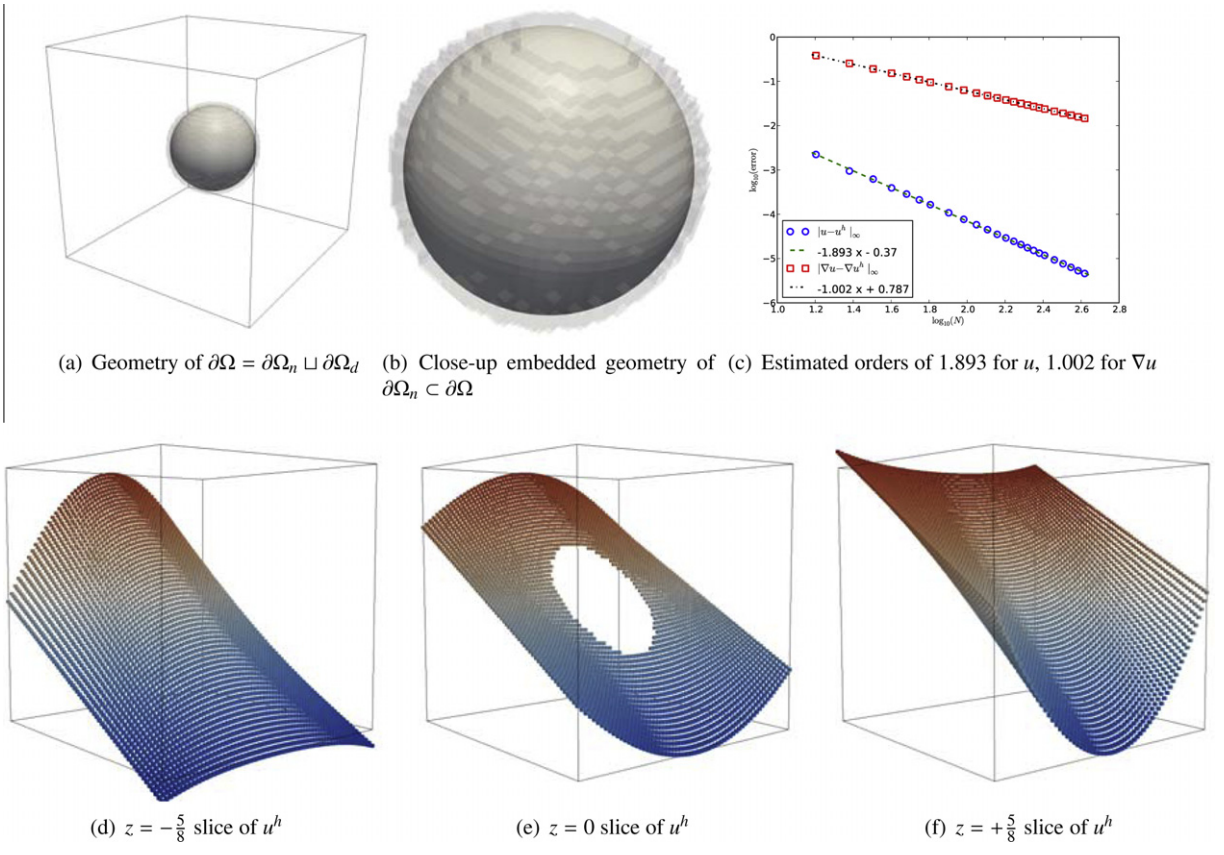
One of our primary contributions is a collection of geometric multigrid algorithms to solve the linear systems arising from the discretizations of the Neumann, Dirichlet, and interface Poisson problems described in Section 3. Multigrid methods are well-known to be more efficient than standard iterative Krylov solvers (such as conjugate gradient), as a multigrid solver can

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Ω (Neumann, Dirichlet) or Ω^- (interface) discretization (b) Ω^+ discretization

Fig. 10. Partitioning the degrees of freedom according to their grid-distance from the embedded boundary or embedded interface.



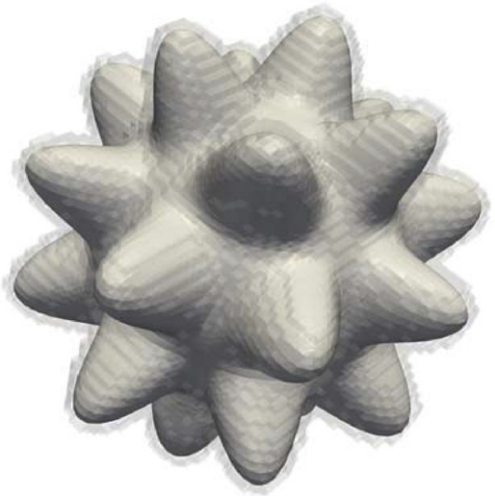
(a) Geometry of $\partial\Omega = \partial\Omega_n \sqcup \partial\Omega_d$ (b) Close-up embedded geometry of $\partial\Omega_n \subset \partial\Omega$ (c) Estimated orders of 1.893 for u, 1.002 for ∇u

(d) $z = -\frac{5}{8}$ slice of u^h (e) $z = 0$ slice of u^h (f) $z = +\frac{5}{8}$ slice of u^h

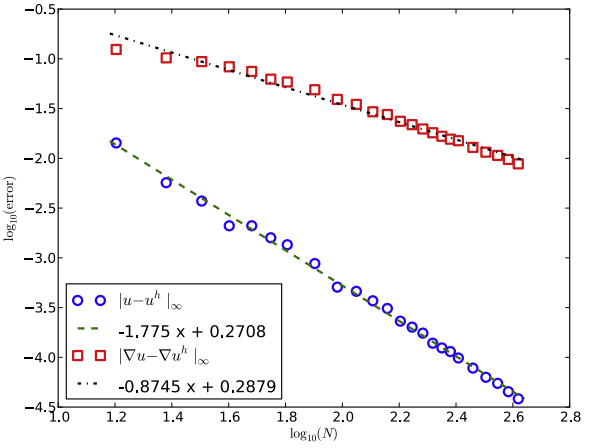
Fig. 11. for Example Section 5.1: geometry of $\partial\Omega$ at $N = 32$, convergence plot of the errors, and z-slices of u^h at $N = 32$. The black wireframe box in (c)-(e) is $\{(x,y) \in [-1,+1]^2\} \times [-1,+1]$.

often operate in $\mathcal{O}(\#\text{ of degrees of freedom})$ time (or nearly so). Additionally, our multigrid solvers are geometric in nature, hence allow implementations with low memory requirements and scalable parallelizability.

We will begin the exposition with a discussion of the grid hierarchy, followed by details regarding the smoothing and transfer operators. Since our multigrid algorithms for the Neumann, Dirichlet, and interface discretizations share the same general principles, we will discuss our multigrid algorithms within the context of all three discretizations simultaneously, noting important differences as they arise. We emphasize that the constraint aggregation described in Section 3.3.2 plays

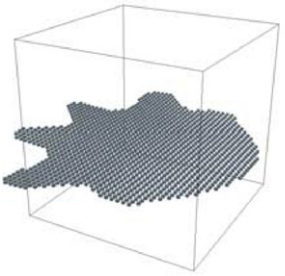


(a) Embedded geometry of $\partial\Omega_n = \partial\Omega$

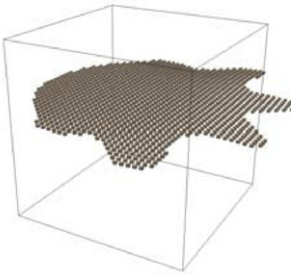


(b) Estimated orders of 1.775 for u, 0.875 for ∇u

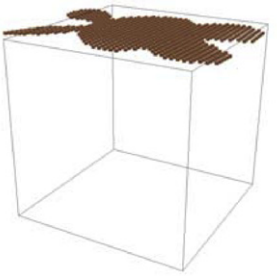
(c) $z = -\frac{1}{2}$ slice of u^h



(d) $z = -\frac{5}{32}$ slice of u^h



(e) $z = +\frac{5}{32}$ slice of u^h



(f) $z = +\frac{1}{2}$ slice of u^h

Fig. 12. for Example Section 5.2: geometry of $\partial\Omega_n$ at $N = 64$, convergence plot of the errors, and z-slices of u^h at $N = 64$. The black wireframe box in (c)–(f) is $\{(x,y) \in [-1/2, +1/2]^2\} \times [-1/2, +1/2]$.

an integral role in our multigrid algorithms for embedded Dirichlet and embedded interface problems, as we base our boundary/interface-local smoother on the Z^tAZ symmetric positive definite system.

We note that we follow standard geometric multigrid principles away from embedded features, and thus our primary focus is the nontrivial treatment of the multigrid components around the embedded features of the discretization. In order to minimize peripheral complexity, we assume that β (for Neumann and Dirichlet problems) or β^+, β^- (interface problems) are constant.

4.1. Discretization

As is characteristic of geometric multigrid methods, we discretize our problem (as desribed in Section 3) within each of a hierarchy of Cartesian grids $\mathcal{G}^h, \mathcal{G}^{2h}, \ldots c$, with the cell resolutions between successive grids in the hierarchy differing by a factor of 2. Thus, with each level in the hierarchy, we associate

- Cartesian grids $\mathcal{G}^h, \mathcal{G}^{2h}, \ldots,$, with the domain embedded as described in Section 3.1;
- Poisson operators A^h, $A^{2h}, \ldots$, (10)/(26); and
- solution and right-hand side vectors $\vec{u}^h, \vec{u}^{2h}, \ldots c$ and $\vec{f}^h, \vec{f}^{2h}, \ldots c$.

For Dirichlet and interface problems, we also associate the aggregated constraint matrices B^h, $B^{2h}, \ldots, c$ (23)/ (29). To simplify the discretization, we assume the constraint aggregation on a given level is independent of the aggregation on other levels. That is, we make no attempt to ensure coherency or geometric consistency between the sets of constraints on successive levels. However, as a result, the constructions of the multigrid components near embedded features require special consideration, as will be explained below. Note that the presence of the aggregate constraints on each level allows one to easily form the Z^tAZ system as described in Section 3.3, and, as we will see, it is this system that we base our smoothing operator on.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

We emphasize that, in spirit, for Dirichlet and interface problems, we are applying multigrid to the saddle point system (19) or (30). In theory, then, we should additionally associate a Lagrange multiplier vector $\vec{\lambda}^h, \vec{\lambda}^{2h}, \ldots$, at each level of the hierarchy. However, we have designed our multigrid algorithms in such a way that, in practice, it is unnecessary (and, indeed, impractical) to explicitly operate on and store these Lagrange multiplier vectors. Instead, we ensure the (aggregate) constraint equations at each level are always satisfied, hence there is no need to restrict $\vec{\lambda}$-residuals or prolongate $\vec{\lambda}$-corrections; and our smoothing operator is based on the Z^tAZ system, which means we can smooth the error in $\vec{u}$ without making any explicit reference to $\vec{\lambda}$.

4.2. Smoothing operator

In describing our smoothing operator, it will be useful to distinguish between non-boundary/non-interfacial and boundary/interfacial degrees of freedom. The former are squared and the latter are circled in Figs. 1 and 3. Non-boundary/non-interfacial degrees of freedom possess the standard 7-point stencil, as depicted in Fig. 5, even within the Z^tAZ systems arising from Dirichlet and interface problems. Thus, on these degrees of freedom, one may apply standard smoothers appropriate for symmetric positive definite systems, such as weighted Jacobi, Gauss–Seidel, Red–Black Gauss–Seidel, etc.

Although the boundary discretization for Neumann problems produces a denser stencil than the standard 7-point stencil, it is still (at least) semidefinite, hence one may still apply standard smoothers to these degrees of freedom as well. However, the embedded boundary/interface discretization for Dirichlet and interface problems is indefinite (recall that we are, in spirit, operating on the saddle point system (19) or (30)), so the standard smoothers mentioned above are not options. Alternative smoothers *might* work, such as Kaczmarz or box smoothers, but they will generally be slower, and they require the use of the Lagrange multiplier $\vec{\lambda}$. We choose instead to apply a standard smoother, such as Gauss–Seidel, on the symmetric positive definite Z^tAZ system (which coincides with the Poisson operator away from embedded features). Note that the Z^tAZ system operates on all the degrees of freedom *except* the independent degrees of freedom. In effect, we have *eliminated* the independent degrees of freedom from the system, such that each update of a boundary/interfacial non-independent degree of freedom in the Z^tAZ system during, say, a Gauss–Seidel step induces an update of one or more (eliminated) independent degrees of freedom to ensure the solution remains in the null space of the constraint system. Thus, if our initial guess at the finest level satisfies the constraints (e.g., $\vec{c} = \left(\left(B_{m_a}^{-1}\vec{p}\right)^t \; 0\right)^t$), then future corrections via smoothing will keep the approximation in the solution space of the constraint system.

As we will see, to avoid complexity, we do not use specialized transfer operators near embedded features, as is done in [26–28,22,87]. However, the incoherency between the feature embeddings and constraint aggregations within successive discretization levels, as well as the absence of the $\vec{\lambda}$ vectors, precludes the successful use of standard transfer operators near these embedded features. We address this by devoting extra smoothing effort around embedded features to drive the corresponding residuals close to zero and propagate non-boundary/non-interface corrections toward embedded features. Thus, a full smoothing sweep will generally consist of a few boundary/interface-local Gauss–Seidel sweeps, followed by a single Gauss–Seidel sweep over all degrees of freedom, and ending with a few more boundary/interface-local Gauss–Seidel sweeps. One can use numerical experimentation to determine exactly how many boundary/interface-local sweeps are necessary, and our experiments indicate that Neumann and Dirichlet problems need only a half dozen or fewer additional boundary/interface-local smoothing sweeps on either side of the smoothing sweep over all degrees of freedom; interface problems seem to need somewhat more additional boundary/interface-local smoothing sweeps. Fortunately, for large resolutions, the single smoothing sweep over all degrees of freedom will dominate the work expended on these boundary/interface-local smoothing sweeps. For complete results, we refer the reader to Section 5.6.

4.3. Transfer operators

Our multigrid algorithms use standard prolongation and restriction operators away from embedded features. We prolongate a coarse-grid correction $\vec{u}^{2h}$ to the fine-grid solution $\vec{u}^h$ via trilinear interpolation: $\vec{u}^h \leftarrow \vec{u}^h + P\vec{u}^{2h}$. We restrict a fine-grid residual in $\vec{u}^h$ to the coarse-grid right-hand side via the scaled adjoint operator $R := 8P^t$.

Often, in the presence of embedded features, one considers introducing specialized transfer operators near these features [26–28,22,87]. As stated above, we have opted to avoid this complexity. However, we cannot rely on the standard transfer operators by themselves to correctly restrict fine-grid residuals and prolongate coarse-grid corrections near embedded features. Thus, we expend extra smoothing effort to ensure the fine-grid residuals near embedded features are close to zero prior to restriction. Indeed, for Dirichlet and interface problems, we restrict identically zero residuals *from all* fine-grid equations corresponding to boundary/interfacial degrees of freedom, which correspond to precisely those rows in the saddle point system involving $\vec{\lambda}$. We additionally only restrict *to a strict subset* of the coarse-grid equations (e.g. only those corresponding to material degrees of freedom, or only those corresponding to non-boundary/non-interfacial degrees of freedom). Further, we prolongate zero values *from* virtual degrees of freedom in the coarse-grid correction, and again expend extra smoothing effort to propagate toward embedded features the more reliable coarse-grid corrections away from the embedded features.

For Dirichlet and interface problems, recall again that, in spirit, we are applying our multigrid algorithms on the saddle point systems, hence obstensibly we should be restricting residuals from the constraint equations as well. However, by

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

smoothing via the Z^tAZ system and (implicitly) propagating all updates to the independent degrees of freedom, we ensure the constraint equations are always satisfied exactly, i.e., have zero residual.

4.4. Details

The preceding sections gave an overview of the general strategy for our multigrid algorithms, and here we only provide some additional details of our implementation of these ideas. Our primary goal is not necessarily to develop the most efficient implementation, but rather to provide a simple reference implementation which can provide a baseline for future research.

We use lexicographically ordered Gauss–Seidel iterations in all phases of our smoothers. The empirical convergence rates we obtain in our numerical examples in Section 5.6 indicate that the Gauss–Seidel method is a sufficiently good smoother away from embedded features. Technically, the pre-restriction and post-prolongation smoothing sweeps serve difference purposes, so one could tailor the details of each to perform optimally for their respective purpose. For simplicity, however, we use identical pre-restriction and post-prolongation smoothing sweeps. Furthermore, we always buttress the Gauss–Seidel sweep over all degrees of freedom with equal numbers of boundary/interface-local Gauss–Seidel sweeps on either side. We refer to this number at the finest level as the *number of boundary smoothing sweeps* (NBSS; Neumann, Dirichlet) and *number of interface smoothing sweeps* (NISS; interface). At each successively coarser level, we increase the number of boundary/interface smoothing sweeps by a factor of 2 (see Algorithms 3 and 4). Since the number of degrees of freedom in a neighborhood of an embedded feature scales as N^2 for a grid resolution of, say, $N \times N \times N$, this increase in the number of boundary/interface smoothing sweeps at coarser levels does not change the overall complexity of our algorithms. Furthermore, we found it significantly improved our v-cycle convergence rates with negligible additional cost per v-cycle.

For the boundary/interface-local Gauss–Seidel sweeps, we iterate over all degrees of freedom within a fixed L^∞-grid-distance of a boundary/interfacial degree of freedom. See Fig. 10 for an example assignment to all degrees of freedom of the (discrete) signed grid-distance to the embedded boundary or embedded interface. We use the terms *boundary smoothing region width* (BSRW; Neumann, Dirichlet) and *interface smoothing region width* (ISRW; interface) to refer to this distance defining the boundary/interface-local region we apply extra Gauss–Seidel sweeps to. Thus, a BSRW/ISRW of 1 refers to all boundary/interfacial degrees of freedom, while a BSRW/ISRW of 2 refers to all degrees of freedom within an L^∞-grid-distance of 1 from a boundary/interfacial degree of freedom.

Within an interface-local Gauss–Seidel sweep, we found it necessary to relax co-located interior and exterior degrees of freedom consecutively. In other words, co-located pairs of degrees of freedom resulting from a single grid vertex duplication should be relaxed one after the other. To be clear, an interface-local Gauss–Seidel sweep which iterates over *only all* interior degrees of freedom followed by *only all* exterior degrees of freedom (or vice versa) *fails* to reduce the residuals around the interface within a reasonable number of iterations.

For completeness, we provide pseudocode for a multigrid v-cycle for Neumann problems (Algorithm 3) and Dirichlet problems (Algorithm 4) (the pseudocode for interface problems would be nearly identical to that for Dirichlet problems, so we omit it). In these algorithm listings, L denotes the number of levels, with the finest level indexed as 1; and we index all variables associated with a given level with the level index (as opposed to $h,2h,\ldots$, as we had been doing above).

Algorithm 3. Multigrid v-cycle algorithm for Neumann problems.

initialize Poisson operators $A^1,\ldots,A^L$ at all levels as described in Section 3.2; allocate space for solution vectors $\vec{u}^1,\ldots,\vec{u}^L$ and right-hand side vectors $\vec{f}^1,\ldots,\vec{f}^L$
set $\vec{f}^1 \leftarrow \vec{f}$ from (10)
set $\vec{u}^1$ as some convenient initial guess satisfying any (grid-aligned) Dirichlet conditions
for $\ell = 1,\ldots,L-1$ **do**
  perform a full smoothing sweep on $A^\ell\vec{u}^\ell = \vec{f}^\ell$ {Section 4.2, with $2^{\ell-1}$ NBSS boundary-local Gauss–Seidel sweeps on each side of a Gauss–Seidel sweep over all degrees of freedom}
  restrict the fine-grid residual $\vec{r}^\ell := \vec{f}^\ell - A^\ell\vec{u}^\ell$ to the coarse-grid right-hand side: $\vec{f}^{\ell+1} \leftarrow R^\ell\vec{r}^\ell$ {Section 4.3; only restrict *to* coarse-grid material equations}
  set $\vec{u}^{\ell+1} \leftarrow \vec{0}$
end for
solve $A^L\vec{u}^L = \vec{f}^L$ exactly {using a sufficient number of Gauss–Seidel iterations, for example}
for $\ell = L-1,\ldots,1$ **do**
  prolongate the coarse-grid correction $\vec{u}^{\ell+1}$ to the fine-grid solution: $\vec{u}^\ell \leftarrow \vec{u}^\ell + P^{\ell+1}\vec{u}^{\ell+1}$ {Section 4.3; prolongate zeros at coarse-grid virtual degrees of freedom}
  perform a full smoothing sweep on $A^\ell\vec{u}^\ell = \vec{f}^\ell$ {Section 4.2, with $2^{\ell-1}$NBSS boundary-local Gauss–Seidel sweeps on each side of a Gauss–Seidel sweep over all degrees of freedom}
end for

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

Algorithm 4. Multigrid v-cycle algorithm for Dirichlet problems.

initialize Poisson operators $A^1,\ldots,A^L$ and aggregated constraint matrices $B^1,\ldots,B^L$ (and/or fundamental basis matrices $Z^1,\ldots,Z^L$) at all levels as described in Section 3.3; allocate space for solution vectors $\vec{u}^1,\ldots,\vec{u}^L$ and right-hand side vectors $\vec{f}^1,\ldots,\vec{f}^L$

set $\vec{f}^1 \leftarrow \vec{f}$ from (10) (without the q contribution, of course)

let $\vec{c} := \begin{pmatrix} B_{m_a}^{-1}\vec{p} \\ 0 \end{pmatrix}$ {$\vec{c}$ satisfies the embedded Dirichlet constraints}

set $\vec{f}^1 \leftarrow Z^t\left(\vec{f}^1 - A^1\vec{c}\right)$ (note: we implicitly identify the domains and codomains of Z and Z^t)

set $\vec{u}^1$ as some convenient initial guess satisfying any grid-aligned Dirichlet conditions

for $\ell = 1,\ldots,L-1$ **do**

  perform a full smoothing sweep on $\left(Z^\ell\right)^t A^\ell Z^\ell \vec{u}^\ell = \vec{f}^\ell$ {Section 4.2, with $2^{\ell-1}$ NBSS boundary-local Gauss–Seidel sweeps on each side of a Gauss–Seidel sweep over all degrees of freedom; be sure to update independent degrees of freedom as necessary to maintain $\vec{u}^\ell$ in the null space of the embedded Dirichlet constraints}

  restrict the fine-grid residual $\vec{r}^\ell := \vec{f}^\ell - (Z^\ell)^t A^\ell Z^\ell \vec{u}^\ell$ to the coarse-grid right-hand side: $\vec{f}^{\ell+1} \leftarrow R^\ell \vec{r}^\ell$ {Section 4.3; restrict zero values *from* fine-grid boundary degrees of freedom, and only restrict *to* coarse-grid non-boundary equations}

  set $\vec{u}^{\ell+1} \leftarrow \vec{0}$

end for

solve $\left(Z^L\right)^t A^L Z^L \vec{u}^L = \vec{f}^L$ exactly {using a sufficient number of Gauss–Seidel iterations, for example}

for $\ell = L-1,\ldots,1$ **do**

  prolongate the coarse-grid correction $\vec{u}^{\ell+1}$ to the fine-grid solution: $\vec{u}^\ell \leftarrow \vec{u}^\ell + P^{\ell+1}\vec{u}^{\ell+1}$ {Section 4.3; prolongate zeros at coarse-grid virtual degrees of freedom}

  perform a full smoothing sweep on $(Z^\ell)^t A^\ell Z^\ell \vec{u}^\ell = \vec{f}^\ell$ {Section 4.2, with $2^{\ell-1}$ NBSS boundary-local Gauss–Seidel sweeps on each side of a Gauss–Seidel sweep over all degrees of freedom; be sure to update independent degrees of freedom as necessary to maintain $\vec{u}^\ell$ in the null space of the embedded Dirichlet constraints}

end for

set $\vec{u}^1 \leftarrow \vec{c} + Z\vec{u}^1$ (note: we implicitly identify the domains and codomains of Z and Z^t)

5. Numerical examples

We now present some numerical examples demonstrating the second order accuracy of our methods for embedded Neumann, embedded Dirichlet, and embedded interface problems, including an example utilizing discontinuity removal for an interface problem with smooth β across the interface. We will additionally present some examples demonstrating the efficiency of our geometric multigrid algorithms.

We discretized our examples on a variety of $N \times N \times N$-cell grids (up to 416^3 for Neumann, Dirichlet, and interface problems with smooth β; up to 320^3 for interface problems with discontinuous β) within the box $[-1,+1]^3$. For each example below, we give a graphic depicting the embedded boundary or interface; a few plots showing typical slices of the discrete approximation u^h, e.g., plots of $u^h(x,y,z_0)$ against (x,y) with $z = z_0$ fixed; and log–log plots of the errors in the discrete approximation $\|u - u^h\|_\infty$ and the gradient of the discrete approximation $\|\nabla u - \nabla u^h\|_\infty$ against the resolution N, which demonstrate second order convergence in u and first order convergence in ∇u. We compute $\|u - u^h\|_\infty$ as the maximum absolute difference between the analytic solution u and the discrete approximation u^h over all material degrees of freedom. We compute $\|\nabla u - \nabla u^h\|_\infty$ as the maximum L^∞-norm between ∇u and ∇u^h over, again, all material degrees of freedom. Note that, strictly speaking, ∇u^h is discontinuous across grid cell faces, and specifically around grid vertices. Thus, we evaluate ∇u^h at a grid vertex by averaging its limits when approached from each of the (up to 8) non-boundary/non-interface incident grid cells (using the trilinear representation of u^h within each incident grid cell). We restrict this averaging to only non-boundary/non-interfacial grid cells to ensure we use only material degrees of freedom in the evaluation of $\nabla\, u^h$.

Occasionally, at higher resolutions, a degree of freedom is so poorly supported that catastrophic cancellation and/or round-off error dominates in the integration calculations (Section 3.1) associated with the degree of freedom. For all the examples below, we eliminate a *virtual* degree of freedom i from the linear system whenever $A_{ii} \leqslant 1 \times 10^{-12}\max_j A_{jj}$, i.e., when its corresponding diagonal entry in the striffness matrix is vanishingly small. We found this elimination to be occasionally necessary to improve the solve times and/or reduce the error in the approximate solution. An alternative solution to this problem of poorly supported degrees of freedom is to perturb the boundary or interface away from grid vertices lying

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

too close (via a perturbation of the level set function values), thus attempting to give sufficient support to all degrees of freedom.

5.1. Embedded Neumann Example 1

Our first two examples apply our method to the embedded Neumann problem:

$$-\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x}\in\Omega;$$
$$\beta\nabla u\cdot\hat{\mathbf{n}} = q(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega_n.$$

This first example uses $\beta(x,y,z) = 2 + y^2 + xz$ and sets f and q according to the exact solution $u(x,y,z) = x\cos y + y^2\sin z$. The domain is given by $\Omega = \{\mathbf{x} : 0.4 < \|\mathbf{x}\|_2 \text{ and } \|\mathbf{x}\|_\infty < 1\}$, with Neumann conditions applied to the embedded portion of the boundary $\partial\Omega_n = \{\mathbf{x} : \|\mathbf{x}\|_2 = 0.4\}$ and Dirichlet conditions applied to the grid-aligned portion of the boundary $\partial\Omega_d = \{\mathbf{x} : \|\mathbf{x}\|_\infty = 1\}$. Fig. 11 depicts the geometry at resolution $N = 32$, a convergence plot of the errors, and several z-slices of u^h at $N = 32$. A least-squares linear regression on the error data yields a convergence order of *1.893* for u and 1.002 for ∇u.

5.2. Embedded Neumann Example 2

Our second example is also an embedded Neumann problem, with $\beta(x,y,z) = 3 + x\cos z + y\sin z$ and f and q set according to the exact solution $u(x,y,z) = z\cos(x^2 - y^2)$. The domain Ω is bounded by the 24-point star level set given in Algorithm 5 with parameters $r_{\min} = 0.6$ and $r_{\max} = 0.9$. Additionally, we rotate the star surface described in Algorithm 5 by -0.3 radians about the $+x$-axis (to introduce some asymmetry). See Fig. 12 for a graphic of the star level set at resolution $N = 64$.

Algorithm 5. Level set function for the 24-point star surface in Example Section 5.2.

{input: $\mathbf{x}\in\mathbb{R}^3$}
{parameters: $0 < r_{\min} < r_{\max}$}
let $i := \text{argmax}_i|x_i|$
if $x_i = 0$ **then**
  return $-r_{\min}$
end if
let $j_1, j_2 \in \{1,2,3\}$ be the other 2 indices other than i
let $s_k := x_{j_k}/|x_i|$, for $k = 1, 2$
{s_1,s_2 are local coordinates on the face of the $[-1,+1]^3$ cube intersected by the ray from **0** through **x**}
assert $(-1 \leqslant s_k \leqslant +1)$, for $k = 1, 2$
$s_k \leftarrow \frac{1}{2}(s_k + \sin\frac{\pi}{2}s_k)$ {apply a slight distortion to give better spacing to the star's points}
let $h := (1 - \cos 2\pi s_1)(1 - \cos 2\pi s_2)$
return $|\mathbf{x}| - (r_{\min} + (r_{\max} - r_{\min})h)$

We apply Neumann boundary conditions over the entire star surface ($\partial\Omega_n = \partial\Omega$), hence the solution u is only determined up to a constant shift. We accounted for this both during during the linear solves (the stiffness matrix is indefinite) and in the evaluation of the error. Fig. 12 shows the convergence plot of the errors and some typical z-slices of u^h at $N = 64$. We obtain convergence orders of 1.775 and 0.875 for u and ∇u, respectively.

5.3. Embedded Dirichlet example

We next demonstrate our method on the embedded Dirichlet problem:

$$-\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x}\in\Omega;$$
$$u = p(\mathbf{x}), \quad \mathbf{x}\in\partial\Omega_d.$$

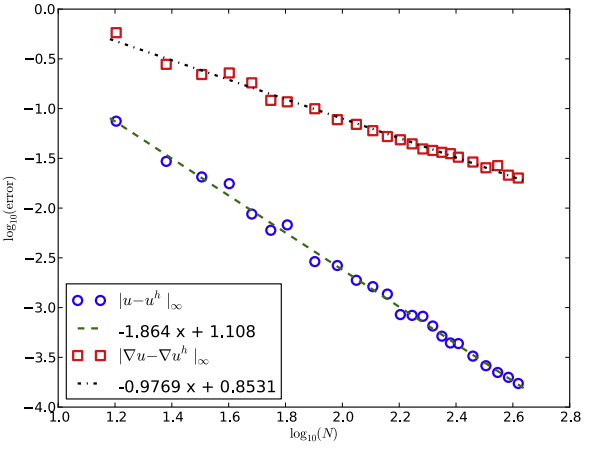
This example uses $\beta(x,y,z) = 7 + x + 2y + 3z$ and sets f and p according to the exact solution $u(x,y,z) = xe^y + \sqrt{1+y^2}e^z$. The domain Ω is bounded by a torus centered at **0** with major radius 0.6, minor radius 0.3, and axis along $(0, -\sin 0.75, \cos 0.75)$ (the $\hat{\mathbf{k}}$ vector rotated -0.75 radians with respect to the $+x$-axis; again, to introduce some asymmetry). We apply Dirichlet boundary conditions over all of $\partial\Omega$, i.e., $\partial\Omega_d = \partial\Omega$. Fig. 13 depicts a graphic of the torus surface at resolution $N = 64$, a convergence plot of the errors, and a few x-slices of u^h at $N = 64$ (that is, we plot $u^h(x_0,y,z)$ against (y,z) for fixed $x = x_0$). We calculated convergence orders of 1.864 and 0.977 for u and ∇u, respectively.

5.4. Embedded interface examples

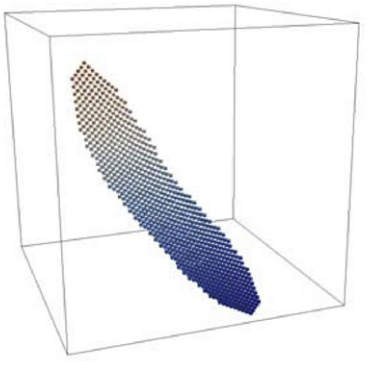
We now apply our method to the embedded interface problem:

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

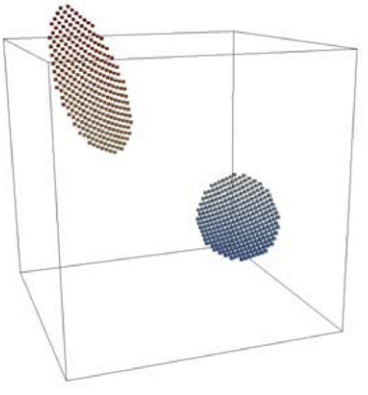
(a) Embedded geometry of $\partial\Omega_d = \partial\Omega$



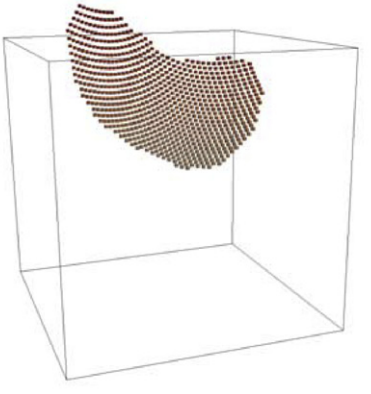
(b) Estimated orders of 1.864 for u, 0.977 for ∇u



(c) $x = -\frac{1}{2}$ slice of u^h



(d) $x = 0$ slice of u^h



(e) $x = +\frac{1}{2}$ slice of u^h

Fig. 13. for Example embedded Section 5.3: geometry of $\partial\Omega_d$ at $N = 64$, convergence plot of the errors, and x-slices of u^h at $N = 64$. The black wireframe box in (c)–(e) is $\{(y,z) \in [-1,+1]^2\} \times [1,3]$.

$$
\begin{aligned}
&-\nabla\cdot(\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x}\in\Omega;\\
&[u] = a(\mathbf{x}), \quad \mathbf{x}\in\Gamma;\\
&[\beta\nabla u\cdot\hat{\mathbf{n}}] = b(\mathbf{x}), \quad \mathbf{x}\in\Gamma.
\end{aligned}
$$

We take $\beta^-(x,y,z) = \alpha^-(10+\sin(xy+z))$ and $\beta^+(x,y,z) = \alpha^+(10+\cos(x+yz))$, where α^- and α^+ are constants. We will vary the ratio α^-/α^+ between 1/100 and 100 to demonstrate the behavior of our method with respect to the contrast in β. We set a and b according to the exact solution $u^-(x,y,z) = x^2+y^2+z^2$, $u^+(x,y,z) = (x+z)^2\sqrt{2+y}$. The interface Γ is the surface of a thickened trefoil knot, with major radius $r_{\text{major}} = 0.8$ and minor radius $r_{\text{minor}} = 0.23$. To be precise, let γ_{trefoil} denote the trefoil knot curve parameterized as

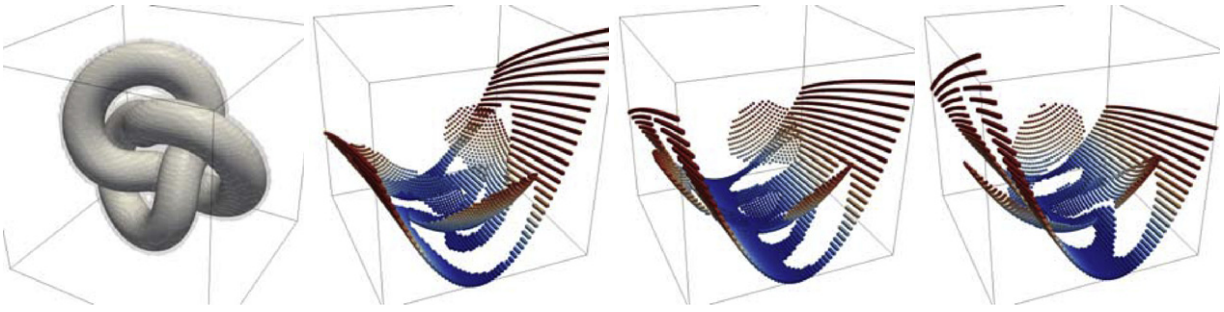
$$\gamma_{\text{trefoil}} := \left\{\frac{r_{\text{major}}}{3}((2+\cos 3t)\cos 2t, (2+\cos 3t)\sin 2t, \sin 3t) : 0 \leqslant t < 2\pi\right\}.$$

We then take

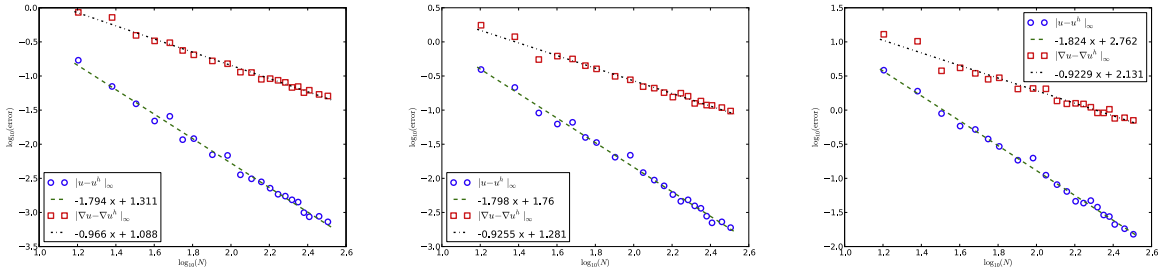
$$\Omega^- := \left\{\mathbf{x}\in\mathbb{R}^3 : \min_{\mathbf{y}\in\gamma_{\text{trefoil}}} \|\mathbf{x}-\mathbf{y}\|_2 < r_{\text{minor}}\right\},$$

with $\Gamma = \partial\Omega^-$ and $\Omega^+ = (-1,+1)^3\backslash(\Omega^- \sqcup \Gamma)$. See Fig. 14 for a graphic of the trefoil knot surface at resolution $N = 64$, a few z-slices of u^h with $(\alpha^-,\alpha^+) = (2,1)$ at $N = 64$, and convergence plots of the errors for various combinations of α^- and α^+. For all tested combinations of α^- and α^+ we obtained an estimated convergence order of $\geqslant 1.794$ and $\geqslant 0.923$ for u and ∇u, respectively.

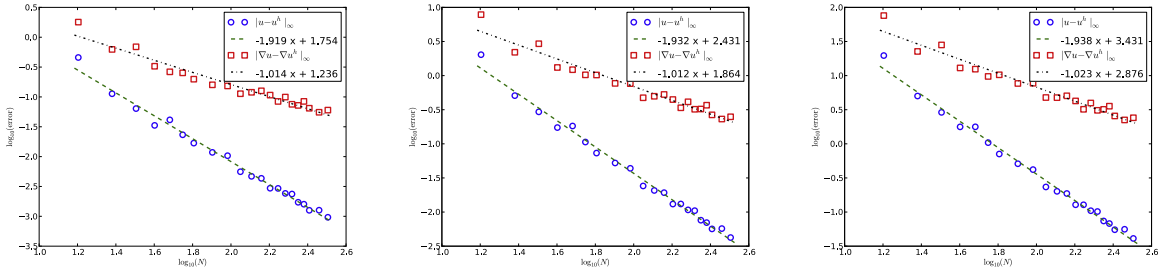
Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Embedded geometry of Γ (b) $z = -\frac{1}{8}$ slice of u^h (c) $z = 0$ slice of u^h (d) $z = +\frac{1}{8}$ slice of u^h



(e) $(\alpha^-, \alpha^+) = (2, 1)$; estimated orders of 1.794 for u, 0.966 for ∇u (f) $(\alpha^-, \alpha^+) = (10, 1)$; estimated orders of 1.798 for u, 0.926 for ∇u (g) $(\alpha^-, \alpha^+) = (100, 1)$; estimated orders of 1.824 for u, 0.923 for ∇u



(h) $(\alpha^-, \alpha^+) = (1, 2)$; estimated orders of 1.919 for u, 1.014 for ∇u (i) $(\alpha^-, \alpha^+) = (1, 10)$; estimated orders of 1.932 for u, 1.012 for ∇u (j) $(\alpha^-, \alpha^+) = (1, 100)$; estimated orders of 1.938 for u, 1.023 for ∇u

Fig. 14. for Example Section 5.4: geometry of Γ, z-slices of u^h with $(\alpha^-, \alpha^+) = (2, 1)$ at $N = 64$, and convergence plots of the errors at various combinations of α^- and α^+. The black wireframe box in (b)–(d) is $\{(x, y) \in [-1, +1]^2\} \times [0, 2]$.

Table 1
Condition numbers (as estimated by PETSc) and number of (preconditioned) conjugate gradient ((P) CG) iterations for the linear systems resulting from discretizing Example Section 5.4 at resolution $N = 256$ for various combinations of (α^-, α^+). For the preconditioning, we used PETSc's incomplete Cholesky (ICC) preconditioner. We also include statistics for the standard 7-pt Laplacian matrix for reference.

Test case	Cond. # (no ICC)	Cond. # (w/ICC)	# of CG iter. (no ICC)	# of PCG iter. (w/ICC)
(2,1)	4.0×10^5	5.6×10^3	5148	616
(10,1)	1.4×10^6	6.5×10^5	8421	5856
(100,1)	1.3×10^7	6.1×10^6	12855	8817
(1,2)	3.3×10^5	5.5×10^3	5168	630
(1,10)	4.7×10^5	2.3×10^5	6450	4529
(1,100)	6.6×10^5	3.1×10^5	7709	5350
7-pt Laplacian, β_-	2.7×10^4	2.7×10^3	1190	395
7-pt Laplacian, β_+	2.7×10^4	2.7×10^3	1194	427

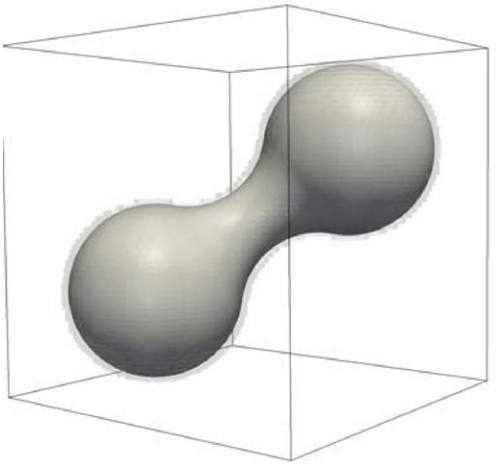
Table 1 shows the effect of the β contrast on the conditioning of the linear systems and the number of (preconditioned) conjugate gradient iterations. We compare the various combinations of α^- and α^+ together with, for reference, the standard

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

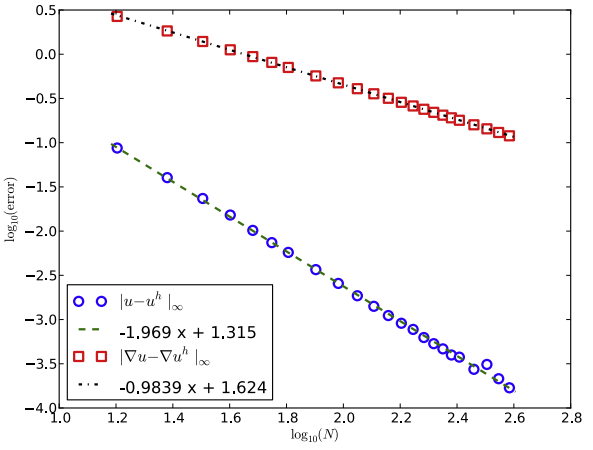
7-pt variable coefficient Laplacian with no interface. For the 7-pt Laplacian system, we show the results from using each of $\beta^- := 10 + \sin(xy + z)$ and $\beta^+ := 10 + \cos(x + yz)$ as the Laplacian coefficient throughout the whole domain. All tests are at a resolution of $N = 256$. We normalized the linear systems to have constant diagonal (Jacobi preconditioning) and solved them via PETSc's [90–92] conjugate gradient function to a relative residual norm of 2.3×10^{-13} of the Jacobi preconditioned system. We configured PETSc to estimate the extreme singular values of the system upon completion of a solve and computed the condition number as the ratio of these extreme singular values. In each test case, we also demonstrate the effects of preconditioning (using PETSc's incomplete Cholesky (ICC) preconditioner, applicable since the Z^tAZ system is symmetric positive definite) on the conditioning of the system and the number of conjugate gradient iterations. We observe that high β constrasts could moderately increase solve times over low β constrasts and the standard 7-pt Laplacian matrix.

5.5. Discontinuity removal

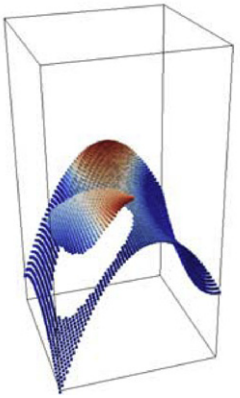
Recall from Section 3.4.1 that if β is smooth across the interface Γ, our method reduces to solving a standard 7-point Poisson system. We demonstrate the applicability of this procedure in this example. We take $\beta(x, y, z) = e^{1+x^2+z^2} + x \sin 4y$ and set a and b according to the exact solution $u^-(x,y,z) = (\cos 4x) \log(1 + y^2 + z^2)$, $u^+(x,y,z) = x\,y^2 + 3yz^2 + 7zx^2$. The interface Γ is the surface of a dumbbell, described by the level set function in Algorithm 6. In this example, the balls of the dumbbell are centered at $\mathbf{x}_0 = (-0.4, -0.4, -0.4)$ and $\mathbf{x}_1 = (0.4, 0.4, 0.4)$ with radii $r_{\text{ball}} = 0.5$; the neck of the dumbbell has radius $r_{\text{neck}} = 0.2$. See



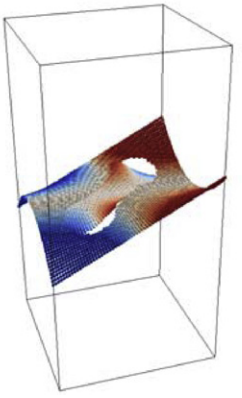
(a) Embedded geometry of Γ



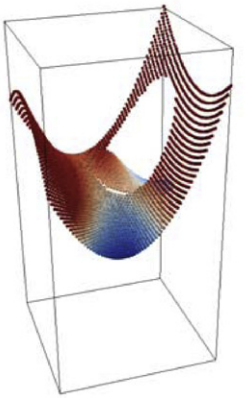
(b) Estimated orders of 1.969 for u, 0.984 for ∇u



(c) $z = -\frac{3}{8}$ slice of u^h



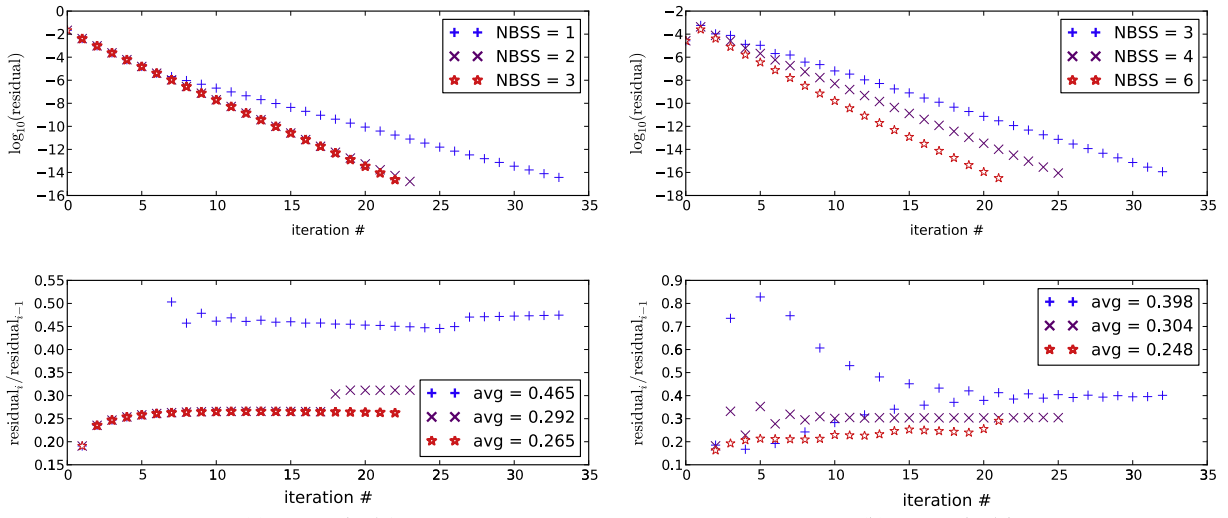
(d) $z = 0$ slice of u^h



(e) $z = +\frac{3}{8}$ slice of u^h

Fig. 15. for Example Section 5.5: geometry of Γ, convergence plot of the errors, and z-slices of u^h at $N = 64$. The black wireframe box in (c)–(e) is $\{(x,y) \in [-1, +1]^2\} \times [-4, 4]$.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) Example 5.1 (b) Example 5.2

Fig. 16. Multigrid v-cycle convergence plots for embedded Neumann Examples 5.1 and 5.2 with $\beta \equiv 1$. The grid resolution is $N = 384$ and the boundary smoothing region width is 1. The top plot in each subfigure shows the residual norm $\|\vec{f} - A\vec{u}\|_\infty$ after each v-cycle iteration for various numbers of boundary smoothing sweeps (NBSS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.

Fig. 15 for a graphic of the dumbbell level set at $N = 64$, a convergence plot of the errors, and a few z-slices of u^h at $N = 64$. We calculated convergence orders of 1.969 and 0.984 for u and ∇u, respectively.

Algorithm 6. Signed distance function for the dumbbell surface in Example Section 5.5

{input: $\mathbf{x} \in \mathbb{R}^3$}
{parameters: $0 < r_{\text{neck}} \leqslant r_{\text{ball}}$; $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^3$}
let $\mathbf{y} := \mathbf{x} - \frac{1}{2}(\mathbf{x}_0 + \mathbf{x}_1)$
{(a,b) are the local coordinates of $\mathbf{x}$ projected onto the plane defined by $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}$ where $(0,0)$ corresponds to $\frac{1}{2}(\mathbf{x}_0 + \mathbf{x}_1)$ and $(\pm\tilde{a}, 0)$ corresponds to $\mathbf{x}_i$}
let $a := \mathbf{y} \cdot \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2}$; $\left(b := \left\|\mathbf{y} - a\frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2}\right\|\right.$
let $\tilde{a} := \frac{1}{2}\|\mathbf{x}_1 - \mathbf{x}_0\|_2$; $\tilde{b} := \frac{\tilde{a}^2 - (r_{\text{ball}} - r_{\text{neck}})^2}{2(r_{\text{ball}} - r_{\text{neck}})}$
if $\tilde{b} \leqslant 0$ or $\frac{|a|}{\tilde{a}} + \frac{b}{\tilde{b}} \geqslant 1$ **then**
  let $d_0 := \sqrt{(\tilde{a} + a)^2 + b^2}$ {distance from $\mathbf{x}$ to $\mathbf{x}_0$}
  let $d_1 := \sqrt{(\tilde{a} - a)^2 + b^2}$ {distance from $\mathbf{x}$ to $\mathbf{x}_1$}
  assert $(d_i = \|\mathbf{x} - \mathbf{x}_i\|)$ for $i = 1,2$
  return $\min\{d_0, d_1\} - r_{\text{ball}}$
else
  return $\left(\tilde{b} - r_{\text{neck}}\right) - \sqrt{a^2 + \left(\tilde{b} - b\right)^2}$
end if

5.6. Multigrid

We described a collection of multigrid algorithms in Section 4 to solve embedded Neumann and embedded Dirichlet problems with β constant (i.e., $\beta \equiv 1$) and embedded interface problems with β^+ and β^- constant (i.e., β is constant over Ω^- and Ω^+ but still discontinuous along Γ). We demonstrate the efficacy of these algorithms in this section. For each of the following examples, we study the convergence behavior of iteratively applying the multigrid v-cycle described in Section 4. We vary the number of pre-restriction and post-prolongation additional boundary/interface smoothing sweeps together with the width of the boundary/interface smoothing region, and show what kind of parameters might be typically necessary

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

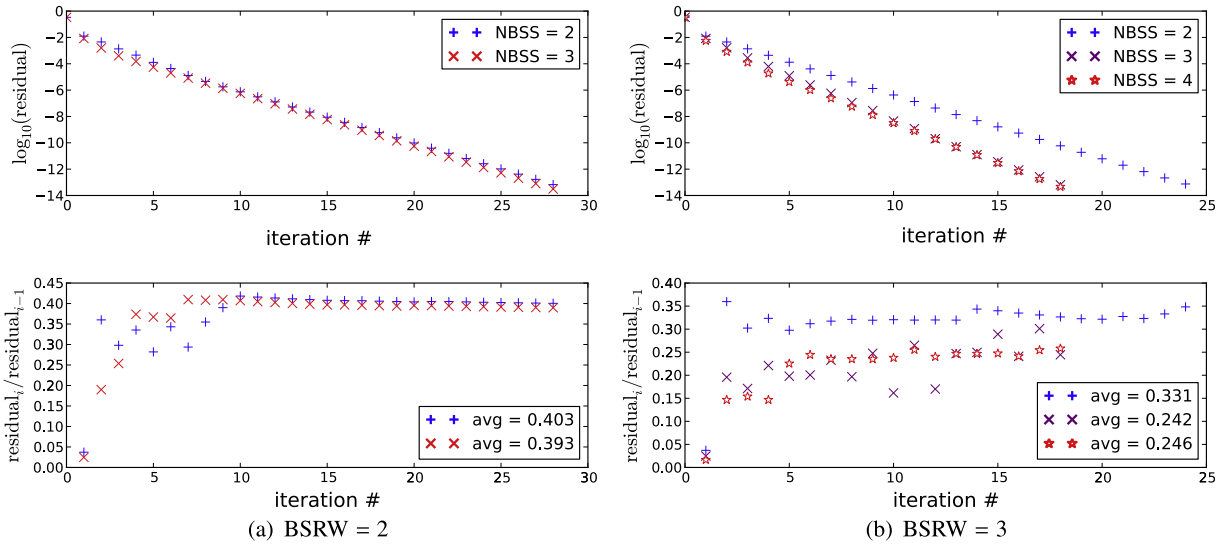


Fig. 17. Multigrid v-cycle convergence plots for embedded Dirichlet Example Section 5.3 with $\beta \equiv 1$ for a boundary smoothing region width (BSRW) of 2 and 3. The grid resolution is $N = 384$. The top plot in each subfigure shows the residual norm $\|\vec{f} - A\vec{u}\|_\infty$ after each v-cycle iteration for various numbers of boundary smoothing sweeps (NBSS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.

to achieve good v-cycle convergence. We note that, generally speaking, for the class of smoothers we are using (straightforward Gauss–Seidel or variants thereof), embedded Neumann and embedded Dirichlet problems require relatively little additional smoothing effort along the boundary. Embedded interface problems, on the other hand, may require significantly more work along the interface, depending highly on the contrast in β.

We first present the results of applying our multigrid algorithm to the embedded Neumann examples in Section 5.1 and Section 5.2, but with $\beta \equiv 1$. Fig. 16 shows plots of the residual norm $\|\vec{f} - A\vec{u}\|_\infty$ and the ratio of successive residual norms versus the v-cycle iteration number at resolution $N = 384 = 3 \cdot 2^7$. For both examples, we were able to obtain a v-cycle convergence rate of about 0.25 with a boundary smoothing region width of only 1 (i.e., only expending extra smoothing effort on boundary degrees of freedom) and relatively few additional boundary smoothing sweeps.

Fig. 17 shows the results of applying our multigrid algorithm to the embedded Dirichlet example in Section 5.3 (except, again, with $\beta \equiv 1$). For this example, we found it necessary to extend the boundary smoothing region out to a width of 2 or 3 to obtain good v-cycle convergence, encompassing all degrees of freedom incident to a grid cell with an L^∞-distance from a boundary grid cell of at most 1 or 2, respectively. In each case, we needed only 3 or 4 additional boundary smoothing sweeps to achieve a stable v-cycle convergence rate. Additional boundary smoothing sweeps above 3 or 4 did not significantly improve the convergence rate. Unsurprisingly, a boundary smoothing region width of 3 gives a better convergence rate (again, about 0.25) than a boundary smoothing region width of 2 (where the convergence rate is, at best, about 0.39). The former, however, requires non-negligibly more effort for smaller resolutions.

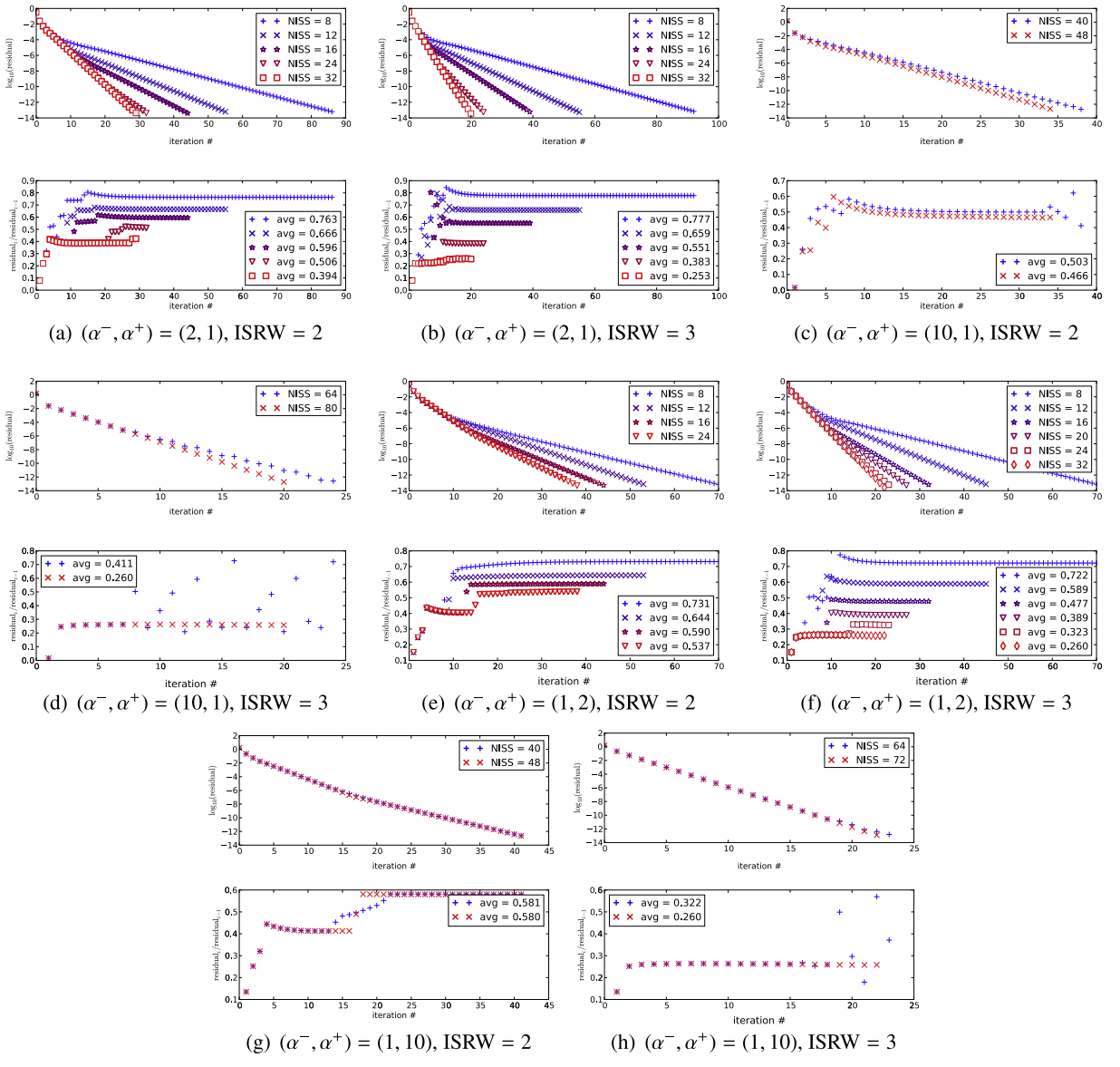
Lastly, we demonstrate our multigrid algorithm on the embedded interface example in Section 5.4 with $\beta^- \equiv \alpha^-$ and $\beta^+ \equiv \alpha^+$. See Fig. 18 for the results. Here, we vary α^-/α^+ only between 1/10 and 10. As for the embedded Dirichlet case, an interface smoothing region width of 2 or 3 is sufficient to obtain a v-cycle convergence rate of about 0.40 or 0.25, respectively. We found that we also needed significantly more additional interface smoothing sweeps than for the embedded Neumann and embedded Dirichlet cases, especially at more extreme β contrasts (e.g., 1/100 or 100).

6. Discussion and conclusion

We presented a virtual node method to solve embedded Neumann, Dirichlet, and interface problems (1)–(5) (cf. [1]) which uses Lagrange multipliers to enforce the Dirichlet condition (4) and the jump condition (2) weakly. We described a general algorithm to define the Lagrange multiplier space that ultimately yields a symmetric positive definite system with better conditioning than that yielded when using the double-wide constraints described in [1]. The geometric intuitiveness of our method makes it relatively easy to implement, and the numerical examples in Section 5 demonstrate its second order accuracy in L^∞. Although simpler embedded domain discretizations exist (see, for example, [30,45]), we believe one distinct advantage of our embedded domain discretizations is that they naturally extend to our embedded interface discretization. Thus, it takes relatively little machinery to understand and implement all three methods.

We described a collection of multigrid algorithms in Section 4 to solve our embedded Neumann, Dirichlet, and interface problems. The results given in Section 5.6 demonstrate that simple v-cycle iteration built around our multigrid algorithms

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023



(a) $(\alpha^-, \alpha^+) = (2, 1)$, ISRW = 2 (b) $(\alpha^-, \alpha^+) = (2, 1)$, ISRW = 3 (c) $(\alpha^-, \alpha^+) = (10, 1)$, ISRW = 2

(d) $(\alpha^-, \alpha^+) = (10, 1)$, ISRW = 3 (e) $(\alpha^-, \alpha^+) = (1, 2)$, ISRW = 2 (f) $(\alpha^-, \alpha^+) = (1, 2)$, ISRW = 3

(g) $(\alpha^-, \alpha^+) = (1, 10)$, ISRW = 2 (h) $(\alpha^-, \alpha^+) = (1, 10)$, ISRW = 3

Fig. 18. Multigrid v-cycle convergence plots for embedded interface Example Section 5.4 with $\beta^- \equiv \alpha^-$, $\beta^+ \equiv \alpha^+$ for a interface smoothing region width (ISRW) of 2 and 3 and various combinations of α^-,α^+. The grid resolution is $N = 256$. The top plot in each subfigure shows the residual norm $\|\vec{f} - A\vec{u}\|_\infty$ after each v-cycle iteration for various numbers of interface smoothing sweeps (NISS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.

yields an efficient solver for embedded Neumann and embedded Dirichlet problems at almost any resolution. Using simple v-cycle iteration to solve embedded interface problems requires a significant amount of interface-local smoothing, so it would likely be most effective at higher resolutions. One avenue of research would be to investigate alternative grid-transfer operators or smoothers around the embedded interface with the hopes of reducing the amount of interface-local smoothing. We would also expect that far fewer interface-local (and boundary-local) smoothing sweeps would be necessary when using a single multigrid v-cycle as a preconditioner to a Krylov method, such as is done in [86].

Acknowledgments

We wish to thank Jacob Bedrossian and James H. von Brecht for their helpful discussions. We also wish to thank Russell Howes and Alexey Stomakhin for submitting typographical errors and providing comments on the later drafts of this paper.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

This work was supported in part by NSF DMS-0502315, NSF DMS-0652427, NSF CCF-0830554, DOE 09-LR-04-116741-BERA, ONR N000140310071, and ONR N000141010730.

Appendix A. Quadrature

For convenience, we reproduce the triangle Gaussian quadrature rules of various orders we used from [93] in Table A.2. For the quadrature points with multiplicity 3, the coordinates should be permuted to give 3 total symmetrically distributed quadrature points, all with the same given quadrature weight. For example, to integrate a cubic polynomial $p(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$ over a triangle T with vertices $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\} \subset \mathbb{R}^3$, one would use the order 3 quadrature rule from Table A.2, which manifests itself as

$$\int_T p(\mathbf{x})d\mathbf{x} = \text{area}(T)\left(-\frac{27}{48}\tilde{p}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) + \frac{25}{48}\left(\tilde{p}\left(\frac{1}{5},\frac{1}{5},\frac{3}{5}\right) + \tilde{p}\left(\frac{1}{5},\frac{3}{5},\frac{1}{5}\right) + \tilde{p}\left(\frac{3}{5},\frac{1}{5},\frac{1}{5}\right)\right)\right), \tag{35}$$

where $\tilde{p}(\alpha_1, \alpha_2, \alpha_3) = p(\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \alpha_3\mathbf{x}_3)$. Note how the multiplicity 3 quadrature point with barycentric coordinates (1/5,1/5,3/5) in Table A.2 represents all of the latter 3 quadrature points in (35) via permutation of the coordinates.

Appendix B. Cell averages

The discretizations described in Section 3 require computing cell averages $\bar{\beta}$, $\bar{f}$, and $\bar{q}$ of β, f, and q, respectively. One can use any of a variety of techniques to compute these averages, and one's choice would likely depend upon whether one has β, f, and q immediately defined pointwise at grid vertices; pointwise at grid edge, face, or cell centers; or analytically throughout the domain, domain boundary, or interface.

We used evaluations of β and f at grid vertices to compute their cell averages. For non-boundary/non-interfacial grid cells, the cell average amounts to a straightforward, equal-weighted average of the values at the eight grid vertices of the cell. For boundary/interfacial grid cells, we used trilinear interpolation to compute the cell average. For example, in the embedded Neumann and embedded Dirichlet discretizations, we compute the cell average of β over grid cell $c_k \in \mathcal{C}^h_{\partial\Omega}$ as

$$\bar{\beta} := \frac{\int_{c_k\cap\Omega}\beta d\mathbf{x}}{\int_{c_k\cap\Omega} d\mathbf{x}} \approx \frac{\sum_{i\in\mathcal{N}^h_{c_k}}\beta_i\int_{c_k\cap\Omega}N_i d\mathbf{x}}{\int_{c_k\cap\Omega} d\mathbf{x}},$$

where, as introduced in Section 3.2, $\{N_i : i \in \mathcal{N}^h_{c_k}\}$ denotes the set of trilinear basis functions associated to the 8 grid vertices of c_k. Note that all integrands remaining in the right-most expression are polynomials, hence the integrals may be evaluated as described in Section 3.1. We compute $\bar{f}$, as well as cell averages in embedded interface discretizations, in a completely analogous fashion.

Table A.2
Triangle Gaussian quadrature rules of order 1–5, as given in [93].

Order	Mult.	Weight	Barycentric coordinates
1	1	1	(1/3,1/3,1/3)
2	3	1/3	(1/6,1/6,2/3)
3	1	−27/48	(1/3,1/3,1/3)
	3	25/48	(1/5,1/5,3/5)
4	3	0.109951743655322	(0.091576213509771,0.091576213509771,0.816847572980459)
	3	0.223381589678011	(0.108103018168070,0.445948490915965,0.445948490915965)
5	1	9/40	(1/3,1/3,1/3)
	3	0.125939180544827	(0.101286507323456,0.101286507323456,0.797426985353087)
	3	0.132394152788506	(0.059715871789770,0.470142064105115, 0.470142064105115)

Table C.3
Condition numbers and (preconditioned) conjugate gradient ((P) CG) solve iterations, both with and without incomplete Cholesky (ICC) preconditioning, for the Z^TAZ system arising from the discretization of a Dirichlet and from the discretization of an interface problem at grid resolution $32 \times 32 \times 32$. The Dirichlet problem has $\Omega = \{\mathbf{x} : |\mathbf{x}| \leqslant 0.8\}$ and $\beta \equiv 1$; the interface problem has $\Gamma = \{\mathbf{x} : |\mathbf{x}| = 0.8\}$ and $(\beta^-, \beta^+) \equiv (1,2)$.

Test case	Cond. # (no ICC)	Cond. # (w/ICC)	# of CG iter. (no ICC)	# of PCG iter. (w/ICC)
Dirichlet, $\Lambda^h = \Lambda^h_2$	3.7×10^{12}	1.1×10^{12}	59,846	61,568
Interface, $\Lambda^h = \Lambda^h_2$	4.4×10^{12}	1.4×10^{13}	97,061	80,225
Dirichlet, $\Lambda^h = \Lambda^h_a$	9.3×10^{2}	2.3×10^{1}	200	44
Interface, $\Lambda^h = \Lambda^h_a$	3.9×10^{3}	4.1×10^{1}	494	61

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

For embedded Neumann discretizations, to simplify implementation, we assume that $q \equiv \beta \nabla u \cdot \hat{\mathbf{n}}$ is available everywhere along the polyhedral representation of $\partial\Omega$. We use the second order quadrature rule from Table A.2 in Appendix A over each polygon of $\mathcal{P}_{\partial\Omega}^{c_k}$ (where $c_k \in \mathcal{C}_{\partial\Omega}^h$) to approximate $\bar{q}$:

$$\bar{q} := \frac{\int_{c_k \cap \partial\Omega} q d\mathbf{S}(\mathbf{x})}{\int_{c_k \cap \partial\Omega} d\mathbf{S}(\mathbf{x})} \approx \frac{\sum_{g \in \mathcal{P}_{\partial\Omega}^{c_k}} \int_g q d\mathbf{S}(\mathbf{x})}{\sum_{g \in \mathcal{P}_{\partial\Omega}^{c_k}} \text{area}(g)}.$$

Appendix C. Double-wide constraint conditioning

As mentioned in Section 3.3.2, we found that the double-wide constraints introduced in [1] present significant conditioning issues in 3 dimensions that do not exist in 2 dimensions. Table C.3 shows the condition numbers and the number of conjugate gradient solve iterations for the Z^TAZ matrices resulting from the discretization of a simple Dirichlet problem and from the discretization of a similarly simple interface problem using each of two alternate discretizations $\mathbf{\Lambda}^h$ of the Lagrange multiplier space $\mathbf{\Lambda}$: $\mathbf{\Lambda}_2^h$, which corresponds to the double-wide constraints; and $\mathbf{\Lambda}_a^h$, which corresponds to the aggregate constraints constructed via the algorithm described in Section 3.3.2. We calculated these statistics using PETSc in exactly the same way as for Table 1. This includes applying Jacobi preconditioning and then solving with (incomplete Cholesky preconditioned) conjugate gradient to a relative residual norm of 2.3×10^{-13} of the Jacobi preconditioned system. Clearly, the conditioning of the Z^TAZ system arising from the double-wide constraints is several orders of magnitude worse than that arising from the constraint aggregation algorithm described in Section 3.3.2.

References

[1] J. Bedrossian, J.H. von Brecht, S. Zhu, E. Sifakis, J.M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains, J. Comput. Phys. 229 (18) (2010) 6405–6426.
[2] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing 5 (1970) 207–213.
[3] J. Bramble, J. King, A finite element method for interface problems in domains with smooth boundaries and interfaces, Adv. Comput. Math. 6 (1996) 109–138.
[4] Z. Chen, J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math 79 (1996) 175–202.
[5] M. Dryja, A Neumann–Neumann algorithm for a mortar discretization of elliptic problems with discontinuous coefficients, Numer. Math. 99 (2005) 645–656.
[6] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous Galerkin, mixed, and conforming Galerkin methods for second order elliptic problems, SIAM J. Numer. Anal. 47 (2009) 1319–1365.
[7] B.I. Wohlmuth, R.H. Krause, Multigrid methods based on the unconstrained product space arising from mortar finite element discretizations, SIAM J. Numer. Anal. 39 (1999) 2001.
[8] J. Huang, J. Zou, A mortar element method for elliptic problems with discontinuous coefficients, IMA J. Numer. Anal. 22 (4) (2001) 549–576.
[9] B.P. Lamichhane, B.I. Wohlmuth, Mortar finite elements for interface problems, Computing 72 (2004) 333–348.
[10] R.J. Leveque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[11] R.J. LeVeque, Z. Li, Immersed interface methods for stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709–735.
[12] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[13] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.
[14] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. 220 (2006) 109–138.
[15] S. Xu, Z.J. Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, J. Comput. Phys. 216 (2006) 454–493.
[16] Z. Tan, D.V. Le, Z. Li, K.M. Lim, B.C. Khoo, An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane, J. Comput. Phys. 227 (2008) 9955–9983.
[17] S. Xu, Z.J. Wang, A 3d immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Eng. 197 (25–28) (2008) 2068–2086. Immersed Boundary Method and Its Extensions.
[18] Z. Li, K. Ito, The immersed interface method: numerical solutions of pdes involving interfaces and irregular domains, Frontiers in Applied Mathematics, vol. 33, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.
[19] J.T. Beale, A.T. Layton, On the accuracy of finite difference methods for elliptic problems with interfaces, Commun. Appl. Math. Comput. Sci. 1 (2006) 91–119.
[20] Z. Li, K. Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. Sci. Comput. 23 (2001) 339–361.
[21] S. Deng, K. Ito, Z. Li, Three-dimensional elliptic solvers for interface problems and applications, J. Comput. Phys. 184 (2003) 215–243.
[22] T. Chen, J. Strain, Piecewise-polynomial discretization and krylov-accelerated multigrid for elliptic interface problems, J. Comput. Phys. 227 (2008) 7503–7542.
[23] Z. Li, A fast iterative algorithm for elliptic interface problems, SIAM J. Numer. Anal. 35 (1998) 230–254.
[24] A. Wiegmann, K.P. Bube, The explicit-jump immersed interface method: finite difference methods for pdes with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (2000) 827–862.
[25] P.A. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. Comput. Phys. 197 (2004) 364–386.
[26] L. Adams, Z. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24 (2002) 463–479.
[27] L. Adams, T.P. Chartier, New geometric immersed interface multigrid solvers, SIAM J. Sci. Comput. 25 (2004) 1516–1533.
[28] L. Adams, T.P. Chartier, A comparison of algebraic multigrid and geometric immersed interface multigrid methods for interface problems, SIAM J. Sci. Comput. 26 (2005) 762–784.
[29] X.-D. Liu, R.P. Fedkiw, M. Kang, A boundary condition capturing method for poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151–178.
[30] F. Gibou, R.P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the poisson equation on irregular domains, J. Comput. Phys. 176 (2002) 205–227.
[31] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem, J. Comput. Phys. 202 (2005) 577–601.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

[32] Z. Jomaa, C. Macaskill, The embedded finite difference method for the poisson equation in a domain with an irregular boundary and dirichlet boundary conditions, J. Comput. Phys. 202 (2005) 488–506.
[33] Y.C. Zhou, S. Zhao, M. Feig, G.W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (2006) 1–30.
[34] I.-L. Chern, Y.-C. Shu, A coupling interface method for elliptic interface problems, J. Comput. Phys. 225 (2007) 2138–2174.
[35] X.-D. Liu, T.C. Sideris, Convergence of the ghost fluid method for elliptic equations with interfaces, Math. Comput. 72 (2003) 1731–1746.
[36] N. Molino, Z. Bao, R. Fedkiw, A virtual node algorithm for changing mesh topology during simulation, in: ACM SIGGRAPH 2005 Courses, SIGGRAPH '05, ACM, New York, NY, USA, 2005.
[37] Z. Bao, J.-M. Hong, J. Teran, R. Fedkiw, Fracturing rigid materials, IEEE Trans. Visual. Comput. Graph. 13 (2007) 370–378.
[38] E. Sifakis, K.G. Der, R. Fedkiw, Arbitrary cutting of deformable tetrahedralized objects, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 73–80.
[39] Y.C. Zhou, G.W. Wei, On the fictitious-domain and interpolation formulations of the matched interface and boundary (mib) method, J. Comput. Phys. 219 (2006) 228–246.
[40] S. Yu, G.W. Wei, Three-dimensional matched interface and boundary (mib) method for treating geometric singularities, J. Comput. Phys. 227 (2007) 602–632.
[41] S. Zhao, G.W. Wei, Matched interface and boundary (mib) for the implementation of boundary conditions in high-order central finite differences, Int. J. Numer. Meth. Eng. 77 (2009) 1690–1730.
[42] K. Pan, Y. Tan, H. Hu, An interpolation matched interface and boundary method for elliptic interface problems, J. Comput. Appl. Math. 234 (2010) 73–94.
[43] S. Hou, X.-D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, J. Comput. Phys. 202 (2005) 411–445.
[44] S. Hou, W. Wang, L. Wang, Numerical method for solving matrix coefficient elliptic equation with sharp-edged interfaces, J. Comput. Phys. 229 (2010) 7162–7179.
[45] Y.T. Ng, C. Min, F. Gibou, An efficient fluid–solid coupling algorithm for single-phase flows, J. Comput. Phys. 228 (2009) 8807–8829.
[46] J. Papac, F. Gibou, C. Ratsch, Efficient symmetric discretization for the poisson, heat and stefan-type problems with robin boundary conditions, J. Comput. Phys. 229 (2010) 875–889.
[47] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for dirichlet problem and applications, Comput. Methods Appl. Mech. Eng. 111 (1994) 283–303.
[48] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A cartesian grid projection method for the incompressible euler equations in complex geometries, SIAM J. Sci. Comput. 18 (1997) 1289–1309.
[49] L. Parussini, V. Pediroda, Fictitious domain approach with hp-finite element approximation for incompressible fluid flow, J. Comput. Phys. 228 (2009) 3891–3910.
[50] Z. Jomaa, C. Macaskill, The shortley-weller embedded finite-difference method for the 3d poisson equation with mixed boundary conditions, J. Comput. Phys. 229 (2010) 3675–3690.
[51] A.J. Lew, G.C. Buscaglia, A discontinuous-galerkin-based immersed boundary method, Int. J. Numer. Methods Eng. 76 (2008) 427–454.
[52] G. Guyomarc'h, C.-O. Lee, K. Jeon, A discontinuous galerkin method for elliptic interface problems with application to electroporation, Commun. Numer. Methods Eng. 25 (2009) 991–1008.
[53] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Meth. Eng. 46 (1999) 131–150.
[54] C. Daux, N. Moës, J. Dolbow, N. Sukumar, T. Belytschko, Arbitrary branched and intersecting cracks with the extended finite element method, Int. J. Numer. Meth. Eng. 48 (2000) 1741–1760.
[55] T. Belytschko, N. Moës, S. Usui, C. Parimi, Arbitrary discontinuities in finite elements, Int. J. Numer. Methods Eng. 50 (2001) 993–1013.
[56] N. Moës, M. Cloirec, P. Cartraud, J.F. Remacle, A computational approach to handle complex microstructure geometries, Comput. Methods Appl. Mech. Eng. 192 (28–30) (2003) 3163–3177.
[57] H. Ji, J.E. Dolbow, On strategies for enforcing interfacial constraints and evaluating jump conditions with extended finite element method, Int. J. Numer. Meth. Eng. 61 (2004) 2508–2535.
[58] T.-P. Fries, T. Belytschko, The intrinsic xfem: a method for arbitrary discontinuities without additional unknowns, Int. J. Numer. Methods Eng. 68 (2006) 1358–1385.
[59] N. Moës, E. Béchet, M. Tourbier, Imposing dirichlet boundary conditions in the extended finite element method, Int. J. Numer. Meth. Eng. 67 (2006) 1641–1669.
[60] S. Groí, A. Reusken, An extended pressure finite element space for two-phase incompressible flows with surface tension, J. Comput. Phys. 224 (2007) 40–58.
[61] F. van der Bos, V. Gravemeier, Numerical simulation of premixed combustion using an enriched finite element method, J. Comput. Phys. 228 (2009) 3605–3624.
[62] B.L. Vaughan, Jr., B.G. Smith, D.L. Chopp, A comparison of the extended finite element method with the immersed interface method for elliptic equations with discontinuous coefficients and singular sources, Commun. Appl. Math. Comput. Sci. 1 (1) (2006) 207–228.
[63] J.T. Beale, D.L. Chopp, R.J. LeVeque, Z. Li, Correction to the article a comparison of the extended finite element method with the immersed interface method for elliptic equations with discontinuous coefficients and singular sources by Vaughan et al., Commun. Appl. Math. Comput. Sci. 3 (1) (2008).
[64] 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 (1990) 1–66.
[65] A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Eng. 191 (2002) 5537–5552.
[66] Z. Li, T. Lin, X. Wu, New cartesian grid methods for interface problems using the finite element formulation, Numer. Math. 96 (1) (2003) 61–98.
[67] A. Hansbo, P. Hansbo, A finite element method for the simulation of strong and weak discontinuities in solid mechanics, Comput. Methods Appl. Mech. Eng. 193 (2004) 3523–3540.
[68] J.-H. Song, P.M.A. Areias, T. Belytschko, A method for dynamic crack and shear band propagation with phantom nodes, Int. J. Numer. Methods Eng. 67 (2006) 868–893.
[69] H.M. Mourad, J. Dolbow, I. Harari, A bubble-stabilized finite element method for dirichlet constraints on embedded interfaces, Int. J. Numer. Methods Eng. 69 (2007) 772–793.
[70] J. Dolbow, L. Franca, Residual-free bubbles for embedded dirichlet problems, Comput. Methods Appl. Mech. Eng. 197 (2008) 3751–3759.
[71] A.V. Kumar, S. Padmanabhan, R. Burla, Implicit boundary method for finite element analysis using non-conforming mesh or grid, Int. J. Numer. Meth. Eng. 74 (2008) 1421–1447.
[72] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, Int. J. Numer. Methods Eng. 78 (2009) 229–252.
[73] I. Harari, J. Dolbow, Analysis of an efficient finite element method for embedded interface problems, Comput. Mech. 46 (2010) 205–211.
[74] D.Y. Kwak, K.T. Wee, K.S. Chang, An analysis of a broken p_1-nonconforming finite element method for interface problems, SIAM J. Numer. Anal. 48 (6) (2010) 2117–2134.
[75] H. Wu, Y. Xiao, An unfitted *hp*-interface penalty finite element method for elliptic interface problemsarXiv:arXiv/1007.2893.
[76] C.L. Richardson, J. Hegemann, E. Sifakis, J. Hellrung, J.M. Teran, An XFEM method for modeling geometrically elaborate crack propagation in brittle materials, Int. J. Numer. Meth. Eng. 88 (10) (2011) 1042–1065.
[77] I. Babuška, The finite element method with lagrangian multipliers, Numer. Math. 20 (1973) 179–192.
[78] J. Pitkäranta, Boundary subspaces for the finite element method with lagrange multipliers, Numer. Math. 33 (1979) 273–289.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023

[79] D. Chapelle, K. Bathe, The inf-sup test, Comput. Struct. 47 (4–5) (1993) 537–545.
[80] Z. Li, The immersed interface method using a finite element formulation, Appl. Numer. Math. 27 (1998) 253–267.
[81] H. Johansen, P. Colella, A cartesian grid embedded boundary method for poisson's equation on irregular domains, J. Comput. Phys. 147 (1998) 60–85.
[82] P. Schwartz, M. Barad, P. Colella, T. Ligocki, A cartesian grid embedded boundary method for the heat equation and poisson's equation in three dimensions, J. Comput. Phys. 211 (2006) 531–550.
[83] R.K. Crockett, P. Colella, D.T. Graves, A cartesian grid embedded boundary method for solving the poisson and heat equations with discontinuous coefficients in three dimensionsLBNL Paper LBNL-2929E.
[84] M. Oevermann, R. Klein, A cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces, J. Comput. Phys. 219 (2006) 749–769.
[85] M. Oevermann, C. Scharfenberg, R. Klein, A sharp interface finite volume method for elliptic equations on cartesian grids, J. Comput. Phys. 228 (2009) 5184–5206.
[86] A. McAdams, E. Sifakis, J. Teran, A parallel multigrid poisson solver for fluids simulation on large grids, ACM SIGGRAPH/Eurograph Symp. Comput. Anim. (2010) 1–10.
[87] J.W.L. Wan, X.-D. Liu, A boundary condition-capturing multigrid approach to irregular boundary problems, SIAM J. Sci. Comput. 25 (2004) 1982–2003.
[88] C. Min, F. Gibou, Geometric integration over irregular domains with application to level-set methods, J. Comput. Phys. 226 (2007) 1432–1443.
[89] M. Benzi, G.H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137.
[90] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, Petsc, 2009. web page, <http://www.mcs.anl.gov/petsc>.
[91] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, Petsc users manual, Tech. Rep. ANL-95/11-Revision 3.0.0, Argonne National Laboratory, 2008.
[92] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkh, pp. 163–202.
[93] G.R. Cowper, Gaussian quadrature formulas for triangles, Int. J. Numer. Methods Eng. 7 (1973) 405–408.

Please cite this article in press as: J.L. Hellrung Jr. et al., A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. (2011), doi:10.1016/j.jcp.2011.11.023