Isotropic Vector Matrix Grid and Face-Centered Cubic Lattice Data Structures

J.F. Nystrom[1] and Carryn Bellomo[2]

[1] University of Akureyri, 602 Akureyri, Iceland
jamesn@unak.is
[2] University of Nevada, Las Vegas, Las Vegas NV 89154, USA
carryn.bellomo@ccmail.nevada.edu

Abstract. Techniques for generating data structures for isotropic vector matrix grids (or face-centered cubic lattices) are presented. Grid basics and some background mathematical foundations are also provided.

1 Introduction

An Isotropic Vector Matrix (IVM) grid, also known as the face-centered cubic (FCC) lattice, has the property that all neighbor nodes of every vertex are equally spaced therefrom[1]. There are 12 nearest neighbor nodes for each interior vertex in an IVM grid/FCC lattice arrangement. Compare now a three-dimensional Cartesian orientation of nodes where every vertex has exactly 26 neighbors; 6 of which are nearest neighbor nodes, 12 of which are next-nearest nodes, and the remaining 8 can be considered furthest neighbor nodes.

Herein we describe two techniques for constructing IVM grids. Section II gives an overview of the Vector Equilibrium (VE) cell and an algorithm[2] for generalized IVM grid construction. Section III encodes two overlapping IVM grids within a Cartesian-orientation of nodes. The Discussion includes ideas for simulations we plan to implement on IVM grids, and the Appendix describes the IVMCEM solver and the omni-directional curl operator.

2 IVM Basics and Generalized Grid Construction

The VE cell, also known as a cuboctahedron, is comprised of a central vertex surrounded by 12 nearest neighbor nodes. Figure 1 shows a VE cell and the six-element IVM vector basis. To distinguish the 12 exterior vertexes of the VE, we use `[+1]` to represent the point at the tip of the IVM basis vector $\mathbf{e}_1$ (see Fig. 1), `[0]` to designate the center of the VE cell, and `[-6]` to represent the vertex in the $-\mathbf{e}_6$ direction. The VE cell contains four hexagonal planes, each plane composed of 6 exterior vertexes and the center vertex. (The color rendering of Fig. 1 shows the a-plane in magenta, the b-plane in red, the c-plane in green, and the d-plane in aqua). The exterior vertexes of each hexagonal plane are:

[1] Herein we adopt R. Buckminster Fuller's[1] use of the term *Isotropic Vector Matrix* when referring to grids that utilize this specific geometrical arrangement of vertexes.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 1096–1099, 2005.

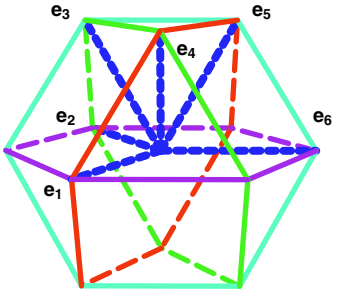

Fig. 1. VE cell and the IVM basis vectors $\mathbf{e}_1$ - $\mathbf{e}_6$

a-plane: `[+1] [-2] [+6] [-1] [+2] [-6]`; b-plane: `[+1] [+4] [+5] [-1] [-4] [-5]`;
c-plane: `[-2] [+4] [+3] [+2] [-4] [-3]`; d-plane: `[-5] [-3] [+6] [+5] [+3] [-6]`.

One way to build an IVM grid is to start with a VE and commence to make the exterior cells of the VE the center of its own VE cell. Each time we add new layer of VE to the outer shell of an IVM we create a higher *frequency* IVM grid. For each layer we add to an IVM grid, the number of nodes increases by $10F^2 + 2$; where F is the frequency of the new grid. For example, the first layer around the nuclear/center node has 12 nodes (and thus creates the VE cell). The second layer has 42 nodes, the third layer 92 nodes, and so on.

In [2] an algorithm which uses this process to build IVM grids is described in detail. The majority of the computation time for this generalized grid construction is occupied by a search into the node list for whether or not a node we are to add is in fact already in the node list. If we use a linear search technique, we obtain an $O(n^2)$ grid construction algorithm. We have implemented a binary search algorithm[3] to obtain an $O(n \log n)$ algorithm and a hashing algorithm[2] to produce an $O(n)$ grid construction algorithm.

3 The IVM Grids Within a Cartesian Grid

The algorithm described briefly in §2 creates a data structure for an IVM grid whereupon simulations can be built. Embedded in the data structure is information to access nearest neighbor vertexes; which is required, for example, to implement the *omni-directional curl operator* (see [2, 4, 5] and the Appendix) or generic finite difference stencils. We have developed another IVM grid generation technique[3] wherein nearest neighbor information is not stored, but instead is calculated using a simple index scheme. We overlay two IVM grids onto a

[2] This hashing technique was proposed and implemented by P.I. Wilson at Texas A&M University - Corpus Christi while working on a grant funded by a United States of America National Science Foundation grant # NSF MII 03-30822.

[3] R.W. Gray suggested this type of organization to one of the authors (JFN) quite a long time ago.

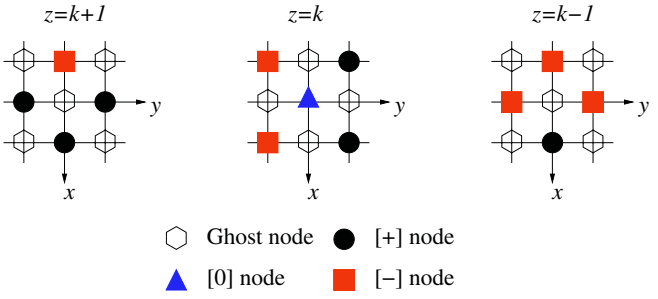

Fig. 2. Placement of VE cell vertexes and *ghosts* on Cartesian grid

three-dimensional Cartesian grid. Here we use the same orientation of lattice structure as is found in the common salt crystal, *NaCl* (see, for example, [6]). For salt, the *Na* atoms are arranged in a FCC lattice (i.e., an IVM grid) that interpenetrates a FCC lattice of *Cl* atoms. We will only be using one of the two IVM grids embedded within the Cartesian grid. The vertexes of the IVM grid not used are hereafter called *ghost* nodes.

The placement of IVM vertexes onto a Cartesian grid is shown in Fig. 2. We use a hexagon shape to indicate ghost nodes. Here we show the center and 12 exterior vertexes of a single VE cell. We maintain the same addressing notation as given in §2. The `[0]` node is shown on Fig. 2 as a solid triangle at the $z = k$ level, vertexes `[+1]` through `[+6]` are shown with a solid circle, and `[-1]` through `[-6]` are shown with a solid square.

Let the center of the VE cell, `[0]`, be located at a Cartesian index of `{i,j,k}`. Following the coordinate frame orientation in Fig. 2, the 12 exterior vertexes of the VE cell are then located with the following indexing scheme:

```
[+1] = {i+1,j,k-1}     [+2] = {i,j-1,k+1}     [+3] = {i+1,j,k+1}
[-1] = {i-1,j,k+1}     [-2] = {i,j+1,k-1}     [-3] = {i-1,j,k-1}

[+4] = {i+1,j+1,k}     [+5] = {i,j+1,k+1}     [+6] = {i-1,j+1,k}
[-4] = {i-1,j-1,k}     [-5] = {i,j-1,k-1}     [-6] = {i+1,j-1,k}
```

Using this indexing scheme it is trivial to write the IVM basis vectors $\mathbf{e}_1$ through $\mathbf{e}_6$ in terms of the Cartesian basis vectors.

4 Discussion

We have presented two disparate techniques for generating IVM grids. The first technique can produce IVM grids of generalized orientation. The second technique fixes the orientation of the IVM grid within a Cartesian framework.

There are many trade-offs associated with each choice of grid generation. The biggest difference we see relates to the parallel implementation of simulations that utilize IVM grids. The data structure created by generalized construction does not easily map to modern processor features such as caching, and can not be easily partitioned onto a multiprocessor cluster (like a *Beowulf* cluster), while

Cartesian placement method has access to all the latest parallelization research for Cartesian-based data structures.

We have used an IVM grid to construct a computational electromagnetic solver (see the Appendix). We also plan to investigate other application areas on IVM grids, including cell population dynamics, a lattice Boltzmann method (D3Q13), and cellular automata systems (including implementing ideas from the *computational cosmography*[5]).

References

1. Fuller, R.B. 1975. *Synergetics*. New York, NY: Macmillan.
2. Nystrom, J.F. 2003. Grid construction and boundary condition implementation for the isotropic vector field decomposition methodology. In Proceedings of *ACES 2003*, 745. Monterey, CA: The Applied Computational Electromagnetics Society.
3. Nystrom, J.F. and C. Bellomo. 2003. Efficient Grid Generation for the IVMCEM Solver. In Proceedings of *PIERS 2003*, 516. Cambridge, MA: The Electromagnetics Academy.
4. Nystrom, J.F. 2002. The isotropic vector field decomposition methodology. In Proceedings of *ACES 2002*, 257. Monterey, CA: The Applied Computational Electromagnetics Society.
5. Nystrom, J.F. 2004. On the Omni-directional Emergence of Form in Computation. *Lecture Notes in Computer Science* **3305**: 632.
6. Ashcroft, N.W. and N.D. Mermin. 1976. *Solid State Physics*. Philadelphia, PA: W.B. Saunders.

Appendix

The IVMCEM solver[2, 3, 4, 5] is a fully-discrete time-domain computational electromagnetic solver. This solver uses the omni-directional curl operator for the spatial part (of the Maxwell equations) and a Runge-Kutta fourth-order integrator (RK4) for the temporal advancement of a solution. In the IVMCEM solver we collocate field quantities at the grid locations using 12 doubled-valued components (6 for the electric field and 6 for the magnetic field).

The omni-directional curl operator calculates the vector curl for fields expressed in terms of the IVM basis vectors $\mathbf{e}_1$ through $\mathbf{e}_6$. For example, vector fields $\mathbf{S}$ and $\mathbf{T}$ can be written thusly:

$$\mathbf{S} = S_1\mathbf{e}_1 + S_2\mathbf{e}_2 + S_3\mathbf{e}_3 + S_4\mathbf{e}_4 + S_5\mathbf{e}_5 + S_6\mathbf{e}_6 \ ,$$
$$\mathbf{T} = T_1\mathbf{e}_1 + T_2\mathbf{e}_2 + T_3\mathbf{e}_3 + T_4\mathbf{e}_4 + T_5\mathbf{e}_5 + T_6\mathbf{e}_6 \ .$$

Let a', b', c', and d' be *plane variables*, each evaluated as a contour integrals on the hexagonal planes of the VE cell (see §2). The omni-directional curl operator calculates $\mathbf{S} = \nabla \times \mathbf{T}$ for $\mathbf{S}$ at the center of the VE cell based on values of $\mathbf{T}$ on the 12 exterior vertexes of the VE cell. Each of the six vector components of $\mathbf{S}$ depends on the contour integrals around two separate hexagonal planes.