A General Performance Model for Parallel Sweeps on Orthogonal Grids for Particle Transport Calculations*

Mark. M. Mathis   Nancy M. Amato
Department of Computer Science
Texas A&M University
{mmathis,amato}@cs.tamu.edu

Marvin L. Adams
Department of Nuclear Engineering
Texas A&M University
mladams@tamu.edu

Technical Report 00-004
Department of Computer Science
Texas A&M University
January 15, 2000

Abstract

There is a growing need to accurately simulate physical systems whose evolutions depend on the transport of subatomic particles. It has long been recognized that the huge computational demands of the transport problem mean that practical solution times will be obtained only by the efficient utilization of parallel processing. For example, since estimates place the time devoted to particle transport in multi-physics simulations at 50-80% of total execution time, parallelizing deterministic particle transport calculations is an important problem in many applications targeted by the DOE's Accelerated Strategic Computing Initiative (ASCI). One common approach to deterministic particle transport calculations is the *discrete-ordinates* method, whose most time consuming step is the transport sweep which involves multiple sweeps through the spatial grid, one for each direction of particle travel. The efficient parallel implementation of the transport sweeps is the key to parallelizing the discrete-ordinates method.

The key contribution of this paper is the first general model which can be used to predict the running time of transport sweeps on orthogonal grids for any regular mapping of the grid cells to processors. Our model, which accounts for machine dependent parameters such as computation cost and communication latency, can be used to analyze and compare the effects of various spatial decompositions on the running time of the transport sweep. Insight obtained from the model yields two significant contributions to the theory of optimal transport sweeps on orthogonal grids. First, our model provides a theoretical basis which explains why, and under what circumstances, the column decomposition of the current standard KBA algorithm is superior to the 'balanced' decomposition obtained by classic domain decomposition techniques. Second, our model enables us to identify a new decomposition, we call *Hybrid*, which proves to be as good as and often superior to the current standard KBA method. Our analysis covers sweeps in two- and three-dimensional spatial domains, and first considers sweeps in only one direction, and then sweeps involving multiple simultaneous directions. We obtain expressions for the completion time and discuss theoretical results.

*This research was supported in part by NSF by CAREER award CCR-9624315 and grants IRI-9619850, ACI-9872126, EIA-9805823, EIA-9810937, EIA-9975018 by DOE ASCI ASAP (Level 2 Program) grant B347886, and by the Texas Higher Education Coordinating Board grant ARP-036327-017. Mathis supported in part by a Dept. of Education Graduate Fellowship.

1 Introduction

There is a growing need to accurately simulate physical systems whose evolutions depend on the transport of subatomic particles (such as neutrons, gammas, thermal radiation, and charged particles) coupled with other complex physics (such as hydrodynamics). For example, photons of optical and near-optical wavelengths have long been used for remote sensing (e.g., in satellite imagery and lidar probing), and, more recently, have found increasing potential in important applications such as non-invasive medical diagnosis or detection of biological aerosols. Moreover, in many simulations, particle transport calculations consume the majority of the computational resources. For example, estimates place the time devoted to particle transport in multi-physics simulations at 50-80% of total execution time [3, 2]. It has long been recognized that the huge computational demands of the transport problem mean that practical solution times will be obtained only by the efficient utilization of parallel processing. Indeed, parallelizing deterministic particle transport calculations is recognized as an important problem in many applications targeted by the DOE's Accelerated Strategic Computing Initiative (ASCI).

One common approach to deterministic particle transport calculations is the *discrete-ordinates* method [5, 7]. This iterative method for solving the first-order form of the transport equation discretizes the energy variable E (using energy 'groups'), the angular directions Ω (using a quadrature set), and the spatial domain R (using a grid), and solves for the flux Ψ of particles at each grid point $r \in R$ for each group $g \in E_g$ and each direction $d \in \Omega$. Each iteration involves *source formation*, a *transport sweep*, and an *acceleration* step. In three dimensions, the number of unknowns needed to accurately characterize the transport solution can easily exceed 100,000 per particle species per spatial cell in each iteration. For example, a trilinear discontinuous finite-element spatial discretization requires 8 unknowns per hexahedral cell, a standard S16 discrete-ordinates angular discretization has 288 unknowns, and a typical calculation might require 50 energy groups. This rather ordinary discretization leads to $8 \times 288 \times 50 = 115,200$ unknowns per cell per particle species per time step.

The most time consuming step of a discrete-ordinates computation is the transport sweep which involves a sweep through the spatial grid in each direction of particle travel (i.e., a sweep for each $d \in \Omega$). In this paper, we concern ourselves with the particular, but important, case of orthogonal grids. In this case, there are only eight distinct sweep orderings in three dimensions (four in two dimensions), corresponding to the diagonal planes sweeping the grid from each of the corners. That is, if all directions in a given octant (quadrant) are processed together, then eight (four) distinct sweeps through the orthogonal spatial grid must be performed in each iteration. Although each such sweep is sequential in nature, all spatial cells on the diagonal sweep plane are independent and can be processed in parallel.

The efficient parallel implementation of the transport sweeps is the key to parallelizing the discrete-ordinates method. As the sweeps themselves must be performed in a particular order (dictated by the direction of particle travel), the main controllable parameter is the mapping of the spatial domain onto the processors, i.e., which spatial cells are assigned to which processors. Note that communication of data between dependent cells assigned to the same processor will be quasi-free. While it might appear that this is an instance of the standard domain decomposition problem (see, e.g., [4]), this is not the case. For example, an ideal domain decomposition would partition an orthogonal grid into sub-domains that are as 'balanced' as possible in all dimensions (since this minimizes boundary area, which directly corresponds to the amount of inter-processor communication, see Figure 1(c)). In contrast, the KBA algorithm [5], which is commonly considered to be the best parallel discrete-ordinates method for orthogonal grids, partitions the grid into columns (which in fact maximizes boundary area, see Figure 1(b)). Intuitively, the reason that classic domain decomposition methods do not work well on this problem is due to the directional dependencies of the sweeps, which are not considered by classic domain decomposition methods.

The key contribution of this paper is the first general model which can be used to predict the running time of transport sweeps in orthogonal grids for any regular mapping of the grid cells to processors. In particular, our model accounts for machine-dependent parameters such as computation and communication/latency costs (assumed constant for all grid cells) and is parameterized by p, the number of processors, (m, n, h), the dimensions of the underlying spatial transport grid, and (ϕ_m, ϕ_n, ϕ_h), the dimensions of the coarse grid processor overlay (which determines the dimensions of the sub-domain assigned to each processor). Thus, our model, which accounts for machine dependent parameters such as computation cost and communication latency, can be used to analyze and compare the effects of various spatial decompositions on the running time of the transport sweep. Insight obtained from the model yields the following contributions to the theory of optimal transport sweeps on orthogonal grids.

- Our model provides a theoretical basis which explains why, and under what circumstances, the column decomposition of the current standard KBA algorithm is superior to the 'balanced' decomposition obtained by classic domain decomposition techniques.
- Our model enables us to identify a new decomposition which proves to be as good as, and often superior to, the current standard KBA method. We call this new decomposition the *Hybrid* method because it incorporates positive aspects of both the KBA and the balanced decomposition.
- A more minor, but still potentially valuable, contribution of our work is to provide theoretical guidelines for selecting the 'block size' parameter for the sweep method (the number of cells a processor should process before communicating).

To our knowledge, this is the first general model for transport sweeps on orthogonal grids. Previous modeling efforts have all concentrated on specific methods of mapping the spatial domain to processors. For example, the KBA algorithm has been analyzed in several papers [5, 1], and a general wavefront method (i.e., one processor per grid cell) is modeled in [3, 2].

Outline of Paper. This paper is outlined as follows. In Section 2, we define parallel sweeps on regular grids for discrete-ordinates computations. We also introduce the three spatial decomposition methods analyzed throughout the paper: the KBA column decomposition, a *Volumetric* decomposition, and the *Hybrid* decomposition. In Sections 3–5 we develop the models for the two- and three-dimensional spatial domains. Section 6 summarizes our findings and discusses future work.

2 Transport Sweeps

As mentioned in Section 1, the transport sweep is the most time consuming step of the *discrete-ordinates* method [5, 7] for performing deterministic particle transport calculations. The discrete-ordinates method for solving the first-order form of the transport equation discretizes the energy variable E (using energy 'groups'), the angular directions Ω (using a quadrature set), and the spatial domain R (using a grid), and solves for the flux Ψ of particles at each grid point $r \in R$ for each group $g \in E_g$ and each direction $d \in \Omega$. The following discussion is based on the two-dimensional spatial domain. However, as we will see later, the discussion generalizes to three dimensions quite easily. Significant differences between the two- and three-dimensional spatial domains are noted throughout.

During each iteration of the discrete-ordinates method, a sweep through the spatial grid is performed for each direction of particle travel (i.e., a sweep for each $d \in \Omega$). A sweep of the grid begins in one of the four (eight) corners, depending on the direction desired. Of course, in general, there may be as many sweeps as there are directions in Ω. However, for the orthogonal, or regular, grids considered in this paper, there are only four unique sweep orderings in two dimensions (eight in three dimensions), one for each quadrant (octant) of directions starting from a corner. That is, if all directions in a given quadrant (octant) are processed together, then four (eight) distinct sweeps through the spatial grid must be performed in each iteration. The sweep for a given quadrant (octant) progresses from the selected corner across the grid to the opposite corner following a diagonal trajectory (see Figure 1(e)). The constraining factor is that every cell behind the sweep must be calculated before any cell ahead of the sweep can be calculated. In other words, assuming a sweep originating at the bottom left hand corner of the grid, for a cell to be processed it must be the case that the cells to the left and below the cell in question must have been previously processed. To simultaneously process all directions $d \in \Omega$, there is one sweep line (plane) for each quadrant (octant) of directions (see Figure 1(f)).

The key factor determining the parallel efficiency of the sweeps is the mapping of the spatial domain onto the processors, that is, the assignment of the spatial cells to the processors. In this paper, we develop a model that can be used to analyze transport sweeps in regular grids for any 'regular' mapping of the grid cells to processors. While the theory is general, our analysis will concentrate on three different methods for performing this decomposition: the column decomposition of the KBA algorithm [5] (which is the current standard), a *Volumetric* method which uses a 'balanced' decomposition (obtained by classic domain decomposition techniques), and then a *Hybrid* approach that combines the first two methods. We have selected these three decompositions since they represent the two extreme cases (the KBA's columns vs. the Volumetric's balanced sub-domains) and an intermediate case (Hybrid).

Each method differs in the manner in which the underlying grid is partitioned among the available processors. Consider the $m \times n$ grid shown in Figure 1(a) (where m and n are the number of rows and columns, respectively) as the base grid. The processor partitions can be represented as a *coarse grid overlay.* For KBA (Figure 1(b)) each processor is allocated an $m \times n/p$ block, where p is the number of processors. Alternatively, for the Volumetric method (Figure 1(c)) each processor operates on an $m/\sqrt{p} \times n/\sqrt{p}$ block. While for the Hybrid method (Figure 1(d)) each processor is given an $m/2 \times n/(p/2)$ block. In three dimensions the spatial decompositions for each method are quite similar.

Inter-processor communication will be required to communicate data from a 'downstream' cell on one processor to an adjacent 'upstream' cell on another processor, where which cells are upstream and downstream depends on the sweep direction. During a sweep, each processor will calculate one block of cells in its partition before it processes the next block in its partition and before the next processor can process its adjacent block. It is important to note that, as the number of cells in one block of a processor's partition is highly dependent on the method being used, so is the time to process one block of cells.

3 Sweeps in Two Dimensions

In this section, we develop our performance model for sweeps in two-dimensional regular grids. We first consider the case of a single sweep through the spatial domain (one quadrant of directions only), and then extend our analysis to multiple simultaneous sweeps (all quadrants of directions). The three-dimensional case, which is mostly a straight forward generalization, is considered in Section 4.

For the purpose of our analysis, it is useful to define a function of coarse grid overlay structure based on the number of processors and the partitioning method used.

$$\phi = (\phi_m, \phi_n) = \begin{cases} (1, p) & \text{KBA}, \\ (2, p/2) & \text{Hybrid}, \\ (\sqrt{p}, \sqrt{p}) & \text{Volumetric}. \end{cases} \tag{1}$$

Here, ϕ_m and ϕ_n are the number of rows and columns in the coarse grid overlay, respectively, such that $\phi_m \times \phi_n = p$. Note that ϕ is not the number of cells each processor is assigned, but rather the overall dimensions of the coarse grid overlay. While we have only shown three cases, we note that ϕ actually represents a family of $\sqrt{p}$ partitions corresponding to $(\phi_m, \phi_n) = (i, p/i)$, $i = 1, 2, \ldots, \sqrt{p}$. Using ϕ we can simplify the previous description of how the underlying grid is partitioned among the available processors. Now each processor is allocated an $m/\phi_m \times n/\phi_n$ block of cells of the base grid.

3.1 A single sweep

For sweep algorithms which perform a single sweep on a regular rectangular grid, the completion time, T, is the sum of the times for each individual step or *work quanta*, t_i.

$$T = \sum_{i=1}^{m/k+\phi_n-1} t_i \tag{2}$$

The upper limit of the summation is equal to one half the perimeter of the grid, as partitioned by each method, minus one. This is the number of steps required to complete a sweep of the underlying grid. Although the sweep does not proceed strictly as a series of 'bulk synchronous' [8] steps, the summation is still a valid approximation of completion time. The reason is that the limit of the summation represents a critical path of *work quanta* that must be computed in order to complete the sweep. Since the limit of the summation depends on ϕ_n, the critical path length will depend on the chosen spatial decomposition. Later, we will make the simplifying assumption that $m/k + \phi_n - 1 \approx m/k + \phi_n$.

We allow for *row blocking* by introducing a block size parameter, k. When $k = 1$, each processor's block partition will be equal to one row of the base grid. Increasing k will allow a processor to calculate multiple rows of its subdomain before communicating. This can be quite advantageous since large messages can typically be sent as easily as small messages.

The time to complete a step or *work quanta* is the time to do all local *computation* plus the time to do all non-local *communication* for one block of cells in a processor's partition. The amount of local computation required is equal to the number of cells in a block times the amount of work required to compute one cell. The amount of non-local communication is the amount of time required to communicate between processors at the boundaries. Combining these two assumptions gives us an expression for t_i.

$$t_i = \omega k \frac{n}{\phi_n} + L \tag{3}$$

where

$$\begin{aligned} \omega &= \text{local computation (Work per base grid cell)} \\ L &= \text{non-local communication (Latency)} \end{aligned}$$

The work per base grid cell, ω, is often referred to as the *grind time* in the transport literature. The grind time is the effective time to update a spatial and angle phase-space cell. As mentioned before, the amount of information communicated at each step depends on k. However, we allow a uniform cost for communication, regardless of the amount of information. This assumption is based on the observation that the startup time associated with sending a message often outweighs the transmission time. Also note that it must be the case that $n \geq \phi_n$ so that the inequality, $n/\phi_n \geq 1$, holds. In other words, the width of a processor's partition must be at least one. We further require that $m \geq n$. That is, if the grid is rectangular, it should be arranged such that the number of rows is larger than the number of columns.

Now, substituting this expression for t_i into the summation and simplifying gives us an expression for the completion time, T.

$$\begin{aligned} T &= \sum_{i=1}^{m/k+\phi_n} \omega k \frac{n}{\phi_n} + L \\ &= (\frac{m}{k} + \phi_n)(\omega k \frac{n}{\phi_n} + L) \\ &= \omega k \frac{n}{\phi_n}(\frac{m}{k} + \phi_n) + L(\frac{m}{k} + \phi_n) \\ &= \omega n(\frac{m}{\phi_n} + k) + L(\frac{m}{k} + \phi_n) \end{aligned} \tag{4}$$

We find it useful to normalize the equation by dividing through by ω.

$$\frac{T}{\omega} = n(\frac{m}{\phi_n} + k) + \frac{L}{\omega}(\frac{m}{k} + \phi_n) \tag{5}$$

3.2 Multiple simultaneous sweeps

Now we expand our analysis to include four simultaneous sweeps through the grid, one for each quadrant of directions. The analysis is much the same as for a single sweep, however, an additional parameter must be included to account for the additional computation involved. Specifically, each processor will now be required to sequentially calculate every block in it's partition through which a wavefront passes. In the single sweep case, there was only one wavefront, so only one block in a processor's partition was processed at any given step. However, for d quadrants there can be as many as d blocks.

So, we introduce a new variable, δ, that corresponds to the number of blocks in a processor's partition that must be computed at each step. The time for each step is now

$$t_i = \delta \omega k \frac{n}{\phi_n} + L \tag{6}$$

where

$$\delta = \begin{cases} 4 & \text{KBA,} \\ 2 & \text{Volumetric \& Hybrid.} \end{cases}$$

The meaning of δ and the justifications for the values chosen for each method are explained in detail in section 5. At this point it is worth noting that δ does not necessarily equal d. We also assume that we have non–blocking sends. After a processor computes one block, it may send results and immediately start the next block. If non–blocking sends are not available, the processor must wait for the communication and L must also be multiplied by δ in equation 6

Now, substituting this new expression for t_i into the summation and simplifying gives us a new expression for the completion time, T.

$$\begin{aligned} T &= \sum_{i=1}^{m/k+\phi_n} \delta\omega k \frac{n}{\phi_n} + L \\ &= \delta\omega n(\frac{m}{\phi_n} + k) + L(\frac{m}{k} + \phi_m) \end{aligned} \tag{7}$$

Again, we normalize the equation by dividing through by ω.

$$\frac{T}{\omega} = \delta n(\frac{m}{\phi_n} + k) + \frac{L}{\omega}(\frac{m}{k} + \phi_n) \tag{8}$$

3.3 Determining k_{opt}

The performance of any method can be tuned using the block size parameter, k. Specifically, we want to choose a k such that the completion time is minimized. The optimal value of k can easily be found by minimizing equation 8 with respect to k.

$$\begin{aligned} \frac{\partial(T/\omega)}{\partial k} &= \frac{\delta n k^2 - (L/\omega)m}{2k^2} \\ 0 &= \frac{\delta n k_{opt}^2 - (L/\omega)m}{2k_{opt}^2} \\ 0 &= \delta n k_{opt}^2 - (L/\omega)m \\ (L/\omega)m &= \delta n k_{opt}^2 \\ k_{opt}^2 &= \frac{(L/\omega)m}{\delta n} \\ k_{opt} &= \sqrt{\frac{(L/\omega)m}{\delta n}} \end{aligned} \tag{9}$$

Of course, since we are only interested in whole rows, we are only interested in positive integer values of k_{opt}. Therefore, k_{opt} should be rounded and we must be content with an *almost* optimal value of k. Note that k_{opt} does not depend on ϕ_n. Therefore, since ϕ_n changes with p, k_{opt} can be chosen *a priori* without knowing how many processors you will be using. However, k_{opt} does depend on δ, so k_{opt} is not independent of the method being used. If two methods do happen to have the same δ, k_{opt} will also be the same.

Examples of the relationship between T/ω and k for each method for different values of L/ω (a *machine parameter* representing the relative cost of communication vs. computation) are displayed in Figure 2. The minimal values of T/ω derived from k_{opt} are shown as lines on the three-dimensional surfaces. In general, T/ω increases with k. However, as L/ω increases (as communication becomes relatively more expensive), it becomes advantageous to choose a slightly larger value of k, i.e., to have fewer (larger) communications (recall our assumption that all communication cost does not depend on message size). Also, for similar reasons, the methods with smaller δ values can afford a slightly larger value of k.

The information in these figures follows logically from equation 8. For example, let us assume that L/ω is large (communcation is relatively more expensive). In this situation, the magnitude of L/ω will increase the contribution of the second term in equation 8. In order to decrease the effect of L/ω we deduce that it may be advantageous to increase k. However, although increasing k will decrease the effect of the second term, it will also increase the contribution of the first term. So, although we may want to increase k, we will probably only increase k slightly. We see from Figure 2(g-i) that this is indeed the case.

On the other hand, let us assume that L/ω is small (communication is relatively inexpensive). Now, the magnitude of the second term in equation 8 will certainly be much smaller than the magnitude of the first term. Therefore, we will probably want to pick a k close to one, thus minimizing the first term. We see from Figure 2(a-c) that this is also true.

3.4 Theoretical Results

Normalized completion times and KBA relative ratios for a variety of scenarios are displayed in Figures 3 and 4. Each method is plotted out until it cannot use any more processors. After that point it is plotted using the maximum number of processors it can use. KBA can use up to n processors, Volumetric can use up to n^2 processors, and Hybrid can use up to $2 \times n$ processors. The k values displayed next to the method names in the legend are the optimal values of k chosen *a priori* using equation 9. We look first at a single sweep (for one quadrant of directions) and then at multiple sweeps (one for each quadrant of directions).

For a single sweep, and for all values of L/ω (which represents the relative cost of communication and computation), KBA is better than Volumetric and Hybrid for small values of p. This is what we expect from equation 5. If p is small, the effect of L/ω will be minimized. The first term will also be minimized, since ϕ_n is largest for KBA. However, as p increases, having the largest ϕ_n becomes a disadvantage. This is due to the increasing contribution of L/ω. Consequently, as L/ω increases (communication is relatively more expensive), KBA is the most affected. Surprisingly, Volumetric, which performs very poorly for small values of p, actually surpasses the other two methods for very large values of p. Note that k is usually pretty small and increases with m and L/ω (up to 10 in this case). Also note that k is the same for all methods, which follows directly from equation 9.

When performing four simultaneous sweeps, one for each quadrant of directions, the Hybrid approach is consistently better than KBA. This is largely due to the fact that Hybrid has a lower δ value than KBA. There is a clear trade off between δ and ϕ_n. We can minimize the first term in equation 5 by choosing a large ϕ_n, but this comes at the price of increasing the second term. Likewise, we can attempt to minimize the first term in equation 8 by choosing a large ϕ_n. However, δ is directly related to ϕ_n, and it happens that the largest value of ϕ_n comes with the largest value of δ. So, any gain that we hoped to obtain by choosing a large ϕ_n is lost. Volumetric also benefits from having a small δ. As before, it surpasses KBA and Hybrid for very large values of p, but it is as fast as Hybrid (and faster than KBA) for the maximum number of processors. Now, k_{opt} will be smaller for all methods (up to 7 in this case), since $\delta > 1$. Also, k_{opt} is still the same for all methods that have the same δ values (Hybrid and Volumetric) and a little smaller for methods with larger δ values (KBA).

The general relationships between the methods are the same for rectangular grids as for square grids (if $m > n$). In general, increasing m has a dampening effect on the behavior of the methods. This follows from equation 8, a larger value of m will decrease the effect of ϕ_n and k on T/ω. The values of k_{opt} for rectangular grids will also be larger. From equation 9 we can see that k_{opt} will be exactly $\sqrt{m/n}$ times larger, since $m/n = 1$ for square grids and $m/n > 1$ for rectangular grids.

4 Sweeps in Three Dimensions

In this section, we generalize our analysis for two-dimensional spatial domains in Section 3 to three-dimensional domains. We first expand the definition of ϕ to three dimensions.

$$\phi = (\phi_m, \phi_n, \phi_h) = \begin{cases} (1, \sqrt{p}, \sqrt{p}) & \text{KBA,} \\ (2, \sqrt{p/2}, \sqrt{p/2}) & \text{Hybrid,} \\ (\sqrt[3]{p}, \sqrt[3]{p}, \sqrt[3]{p}) & \text{Volumetric.} \end{cases} \tag{10}$$

Now, ϕ_m, ϕ_n, and ϕ_h are the height, width, and depth of the coarse grid overlay, respectively, such that $\phi_m \times \phi_n \times \phi_h = p$. Again, ϕ is not the number of cells each processor is assigned, but rather the overall dimensions of the coarse grid overlay. Again, while we only show three cases, we note that ϕ actually represents a family of $\sqrt[3]{p}$ partitions corresponding to $(\phi_m, \phi_n, \phi_h) = (i, \sqrt{p/i}, \sqrt{p/i})$, $i = 1, 2, \ldots, \sqrt[3]{p}$. Now, in terms of ϕ, each processor is allocated an $m/\phi_m \times n/\phi_n \times h/\phi_h$ block of cells of the base grid.

4.1 A single sweep

In three dimensions, the completion time, T, for a single sweep is the sum of the times for each individual step or *work quanta*, t_i.

$$T = \sum_{i=1}^{m/k+\phi_n+\phi_h-2} t_i \tag{11}$$

Again, the upper limit of the summation is equal to the critical path length of the grid, as partitioned by each method. This is the number of steps required to complete a sweep of the underlying grid. Note that the critical path length is still highly dependent on the method used. We will also assume that $m/k + \phi_n + \phi_h - 2 \approx m/k + \phi_n + \phi_h$. In three dimensions, k serves much the same purpose as in two. Careful selection of k allows the processors to calculate more grid cells before communicating, thus minimizing the time spent communicating and consequently minimizing the completion time.

The time to complete a step or *work quanta* is the time to do all local *computation* plus the time to do all non-local *communication* for one block of cells in a processor's partition. The amount of local computation required is equal to the number of cells in a block times the amount of work required to compute one cell. The amount of non-local communication is the amount of time required to communicate between processors at the boundaries. Combining these two assumptions gives us an expression for t_i.

$$t_i = \omega k \frac{n}{\phi_n} \frac{h}{\phi_h} + L \tag{12}$$

where

$$\begin{aligned} \omega &= \text{local computation (Work per base grid cell)} \\ L &= \text{non-local communication (Latency)} \end{aligned} \tag{13}$$

Again, we allow a uniform cost for communication, regardless of the amount of information. Also note that it must be the case that $n \geq \phi_n$ and $h \geq \phi_h$ so that the inequalities, $n/\phi_n \geq 1$ and $h/\phi_h \geq 1$, hold. In other words, the width and depth of a processor's partition must be at least one. We further require that $m \geq n \geq h$. That is, if the grid is rectangular, it should be arranged such that the height is the largest dimension, followed by the width and finally the depth.

Now, substituting this expression for t_i into the summation and simplifying gives us an expression for the completion time, T.

$$\begin{aligned} T &= \sum_{i=1}^{m/k+\phi_n+\phi_h} \omega k \frac{n}{\phi_n}\frac{h}{\phi_h} + L \\ &= (\frac{m}{k} + \phi_n + \phi_h)(\omega k \frac{n}{\phi_n}\frac{h}{\phi_h} + L) \\ &= \omega k \frac{n}{\phi_n}\frac{h}{\phi_h}(\frac{m}{k} + \phi_n + \phi_h) + L(\frac{m}{k} + \phi_n + \phi_h) \\ &= \omega n h(\frac{m}{\phi_n \phi_h} + \frac{k}{\phi_n} + \frac{k}{\phi_h}) + L(\frac{m}{k} + \phi_n + \phi_h) \end{aligned} \tag{14}$$

Once again, we obtain a normalized expression for the time by dividing through by ω.

$$\frac{T}{\omega} = nh(\frac{m}{\phi_n \phi_h} + \frac{k}{\phi_n} + \frac{k}{\phi_h}) + \frac{L}{\omega}(\frac{m}{k} + \phi_n + \phi_h) \tag{15}$$

4.2 Multiple simultaneous sweeps

As with the two-dimensional case, multiple simultaneous sweeps in three dimensions can be satisfactorily addressed by including the δ parameter. The time for each step is now

$$t_i = \delta \omega k \frac{n}{\phi_n}\frac{h}{\phi_h} + L \tag{16}$$

where

$$\delta = \begin{cases} 8 & \text{KBA}, \\ 4 & \text{Volumetric \& Hybrid.} \end{cases}$$

The meaning of δ and the justifications for the values chosen for each method are explained in detail in section 5. As in two dimensions, we again assume we have non-blocking sends.

Now, substituting this new expression for t_i into the summation and simplifying gives us a new expression for the completion time, T.

$$\begin{aligned} T &= \sum_{i=1}^{m/k+\phi_n+\phi_h} \delta \omega k \frac{n}{\phi_n}\frac{h}{\phi_h} + L \\ &= \delta \omega n h(\frac{m}{\phi_n \phi_h} + \frac{k}{\phi_n} + \frac{k}{\phi_h}) + L(\frac{m}{k} + \phi_n + \phi_h) \end{aligned} \tag{17}$$

Finally, we normalize the equation by dividing through by ω.

$$\frac{T}{\omega} = \delta nh(\frac{m}{\phi_n \phi_h} + \frac{k}{\phi_n} + \frac{k}{\phi_h}) + \frac{L}{\omega}(\frac{m}{k} + \phi_n + \phi_h) \tag{18}$$

4.3 Determining k_{opt}

As before, the performance of any method in can be tuned using the block size parameter,k. Specifically, we want to choose a k such that the completion time is minimized. The optimal value of k can easily be found by minimizing equation 18 with respect to k.

$$
\begin{aligned}
\frac{\partial(T/\omega)}{\partial k} &= \delta nh(\frac{1}{\phi_n} + \frac{1}{\phi_h}) - (\frac{L}{\omega})(\frac{m}{k^2}) \\
0 &= \delta nh(\frac{\phi_n + \phi_h}{\phi_n \phi_h}) - (\frac{L}{\omega})(\frac{m}{k_{opt}^2}) \\
(\frac{L}{\omega})(\frac{m}{k_{opt}^2}) &= \delta nh(\frac{\phi_n + \phi_h}{\phi_n \phi_h}) \\
k_{opt}^2 &= (\frac{(L/\omega)m}{\delta nh})(\frac{\phi_n \phi_h}{\phi_n + \phi_h}) \\
k_{opt} &= \sqrt{(\frac{(L/\omega)m}{\delta nh})(\frac{\phi_n \phi_h}{\phi_n + \phi_h})}
\end{aligned}
\tag{19}
$$

Of course, we are only interested in positive integer values of k_{opt}. Therefore, k_{opt} should be rounded and we must be content with an *almost* optimal value of k. Note that in three dimensions k_{opt} depends on the number of processors used. This means that k_{opt} can only be chosen once the number of processors to be used is known. Also, since k_{opt} depends on ϕ_n and ϕ_h, as well as δ, k_{opt} will be more dependent on the method used in three dimensions than in two dimensions.

Examples of the relationship between T/ω and k for each method for different values of L/ω are displayed in Figure 5. The minimal values of T/ω derived from k_{opt} are shown as lines on the three-dimensional surfaces. In general, T/ω does not change much with k and it is generally best to choose a small value for k. However, as L/ω increases (as communication becomes more expensive), it becomes advantageous to choose a slightly larger value of k, specifically when using a very large number of processors.

As in two dimensions, the information in these figures follows logically from the equation for normalized completion time. For example, let us assume that L/ω is large (communication is relatively more expensive than computation). In this situation, the magnitude of L/ω will increase the contribution of the second term in equation 18. In order to decrease the effect of L/ω we deduce that it may be advantageous to increase k. However, although increasing k will decrease the effect of the second term, it will also increase the contribution of the first term. Furthermore, the situation looks less promising than before because k appears twice in the first term. It also appears that there is a much greater dependence on ϕ_n and ϕ_h. So, although we may want to increase k, we will probably only increase k slightly, even less than before. We see from Figure 5(g-i) that this is indeed the case. In fact, the only time it seems profitable to choose a very large k is when ϕ_n and ϕ_h are also very large, which also follows from equation 18.

For another example, let us assume that L/ω is small (communication is relatively inexpensive). Now, the magnitude of the second term will be much smaller than the magnitude of the first term. In this case, it will probably not be useful to choose a very large k at all. Figure 5(a-c) confirms our suspicions.

4.4 Theoretical Results

Normalized completion times and KBA relative ratios for a variety of scenarios are displayed in Figures 6 and 7. Each method is plotted out until it can not use any more processors. After that point it is plotted using the maximum number of processors it can use. KBA can use up to $n \times h$ processors, Volumetric can use up to $n^2 \times h$ processors, and Hybrid can use up to $2 \times n \times h$ processors. In each scenario, k_{opt} is chosen individually for each number of processors using equation 19. The k_{opt} values displayed next to the method names in the legend are the largest values of k_{opt} used by that method for any number of processors. This is different from the two-dimensional case, where k_{opt} is the same for any number of processors and we were able to choose k_{opt} *a priori*. We look first at sweeps for one octant of directions then at sweeps for all octants of directions.

The relationships between the three methods are largely the same in two- and three dimensions. However, there are some notable exceptions. The most important difference is that Hybrid does not do as well in three dimensions as it did in two dimensions. Now, in the multiple sweep case, where before Hybrid was better than KBA for the entire range of p, Hybrid does not surpass KBA until p is fairly large. This follows from the equations. In the first term of equation 8, m is divided by ϕ_n and k is not scaled at all. In the first term of equation 18, however, m is divided by $\phi_n \phi_h$ and k is also scaled by $(\phi_n + \phi_h)/(\phi_n \phi_h)$. Furthermore, the second term of equation 18 is dependent on only $\phi_n + \phi_h$, not $\phi_n \phi_h$. What this means is that we can be much more aggressive using ϕ to minimize T/ω. We can minimize the first term of equation 18 by choosing large ϕ_n and ϕ_h, without the second term blowing up. The only time we need to worry about the effect of the second term is when p becomes sufficiently large, this is the point where Hybrid surpasses KBA in the figures. Also not that the value of p where this occurs will be lower for higher values of L/ω (relatively expensive communication).

Volumetric also behaves differently. In two dimensions, Volumetric starts out slower than KBA and Hybrid for small values of p, surpasses both methods for large values of p, and then winds up equal to the fastest method for the maximum number of processors. In three dimensions, Volumetric starts out slow for small values of p, then just barely surpasses KBA and Hybrid for large values of p, and then winds up much worse than either method for the maximum number of processors. This fact can be inferred from equation 18. As p increases, the effect of L/ω (relatively higher communication costs) on the completion time increases. In fact, if $\phi_n \approx \phi_h$, it increases twice as fast as before. So, the negative effects of L/ω on Volumetric's completion time will appear sooner in three dimensions than in two.

One other notable difference is that the figures for three dimensions are not as smooth as those for two dimensions. This is because k_{opt} has to be calculated separately for each number of processors. In two dimensions we were able to calculate k_{opt} *a priori*. This explains some of the unusual jumps in the three-dimensional figures.

5 Determining δ

The δ function represents the depth of local computation that must be done on each processor during the sweep. Alternatively, the depth of local computation is the number of sweeps that are computed on each processor during each step. We could arbitrarily set δ equal to the number of distinct sweeps being computed, but that would not accurately reflect the situation. It is certainly not the case that every processor participates in each sweep at at every step. Therefore, we should derive an expression for δ that accounts for the asymmetry of the sweeps.

The exact value of δ for any individual step can easily be determined by inspecting the grid. For a given step, δ is precisely the maximum number of wavefronts crossing any processor's partition. We can use this method of inspection to calculate values of δ for every step for each method. The results for two- and three dimensions are displayed in Figure 8. The one direction cases are included to demonstrate that equation 5 is really a special case of equation 8 with $\delta = 1$.

It is clear from these figures that δ follows a regular and predictable pattern. Most of the time is spent on the plateau representing the steady-state operation of the sweep. The values of δ that we use above in equations 6 and 16 are the steady-state values. The remainder of the time is spent in either pipeline fill or empty stages. It is worth noting that the amount of time spent in these stages is less for Volumetric and Hybrid than for KBA. Also note that, although Volumetric has a short startup time, it also has some difficulty keeping its pipeline full.

During steady-state operation, the δ value for Volumetric and Hybrid is (generally) half that of KBA. The reason for this is that both of these methods subdivide the grid horizontally as well as vertically, while KBA does not. Therefore, wavefronts crossing the top of the grid will not interfere with wavefronts crossing the bottom of the grid, and δ can be *at most* half the total number of wavefronts. Alternatively, since KBA does not subdivide the grid horizontally, δ must be *at least* half the total number of wavefronts and is generally *equal* to the number of wavefronts.

6 Conclusion

The methods we have analyzed represent a family of algorithms for two- and three-dimensional sweeps. Given an input grid and a parallel computer system (i.e., number of processors and estimates of computation and communication costs for each grid cell), one can select the best method for the given configuration. Furthermore, one can minimize the completion time by selecting an optimal block size.

In general, the Hybrid method performs as well as and often better than KBA on an average system, such as an SMP. KBA is better for systems with a small number of processors where the communication latency is low, such as a node of an SMP where there are two processors sharing a board. Volumetric would work well on a system with very high communication costs but a very large number of processors, such as a network of workstations or SMP cluster.

We are currently implementing our methods in SWEEP3D [6], a simple benchmark discrete-ordinates particle transport code for regular grids. The final version of the paper will compare experimental results with the theoretical results presented here.

Acknowledgements

We would like to thank Lawrence Rauchwerger, Paul Nelson, and the rest of the ASCI group at Texas A&M for useful discussions regarding this work.

References

[1] Randal S. Baker and Raymond E. Alcouffe. Parallel 3D S_N Performance for DANTSYS/MPI on the Cray T3D. In *Proc. Int. Conf. on Mathematical Methods and Supercomputing for Nuclear Applications*, volume 1, pages 377–393, October 1997.

[2] A. Hoisie, O. Lubeck, and H. Wasserman. Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications. Technical Report LAUR-98-3316, Los Alamos National Laboratories, August 1998.

[3] A. Hoisie, O. Lubeck, and H. Wasserman. Scalability analysis of multidimensional wavefront algorithms on large-scale SMP clusters. In *Frontiers*, 1999.

[4] George Karypis and Vipin Kumar. METIS, unstructured graph partitioning and sparse matrix ordering system. version 2.0. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, August 1995.

[5] K. R. Koch, R. S. Baker, and R. E. Alcouffe. Solution of the first-order form of the 3D discrete ordinates equation on a massively parallel processor. *Transactions of the American Nuclear Society*, 65:198–199, 1992. 1992 Annual Meeting, Boston, MA.

[6] Lawrence Livermore National Laboratories. The ASCI SWEEP3D README File. On the web at www.llnl.gov/asci_benchmarks/asci/limited/sweep3d/sweep3d_readme.html, December 1999.

[7] E. E. Lewis and W. F. Miller. Computational methods of neutron transport. In *American Nuclear Society Inc.*, LaGrange Park, IL, 1993.

[8] L. Valiant. Bridging model for parallel computation. *Comm. ACM*, 33(8):103–111, 1990.

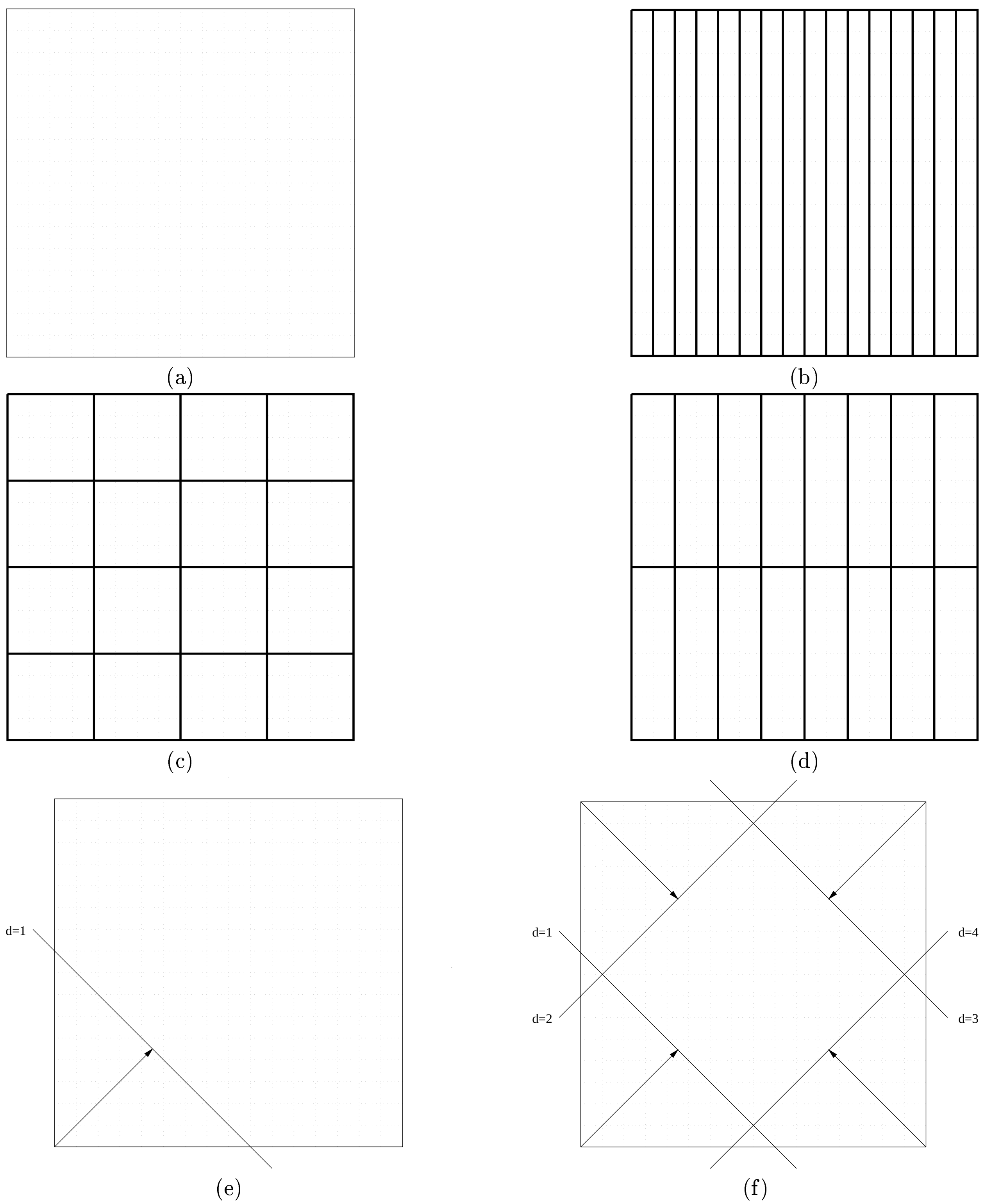


Figure 1: Figure (a) is a diagram of an example two-dimensional base grid ($m = n = 16$). Figures (b), (c), and (f) demonstrate how KBA ($\phi_m = 1, \phi_n = 16$), Volumetric ($\phi_m = \phi_n = 4$), and Hybrid ($\phi_m = 2, \phi_n = 8$) partition the base grid shown in (a), given $p = 16$ processors. In (e) and (f) we show snapshots of two-dimensional sweeps in progress for one sweep only and all distinct sweeps, respectively ($m = n = 16$, $i = 10$).

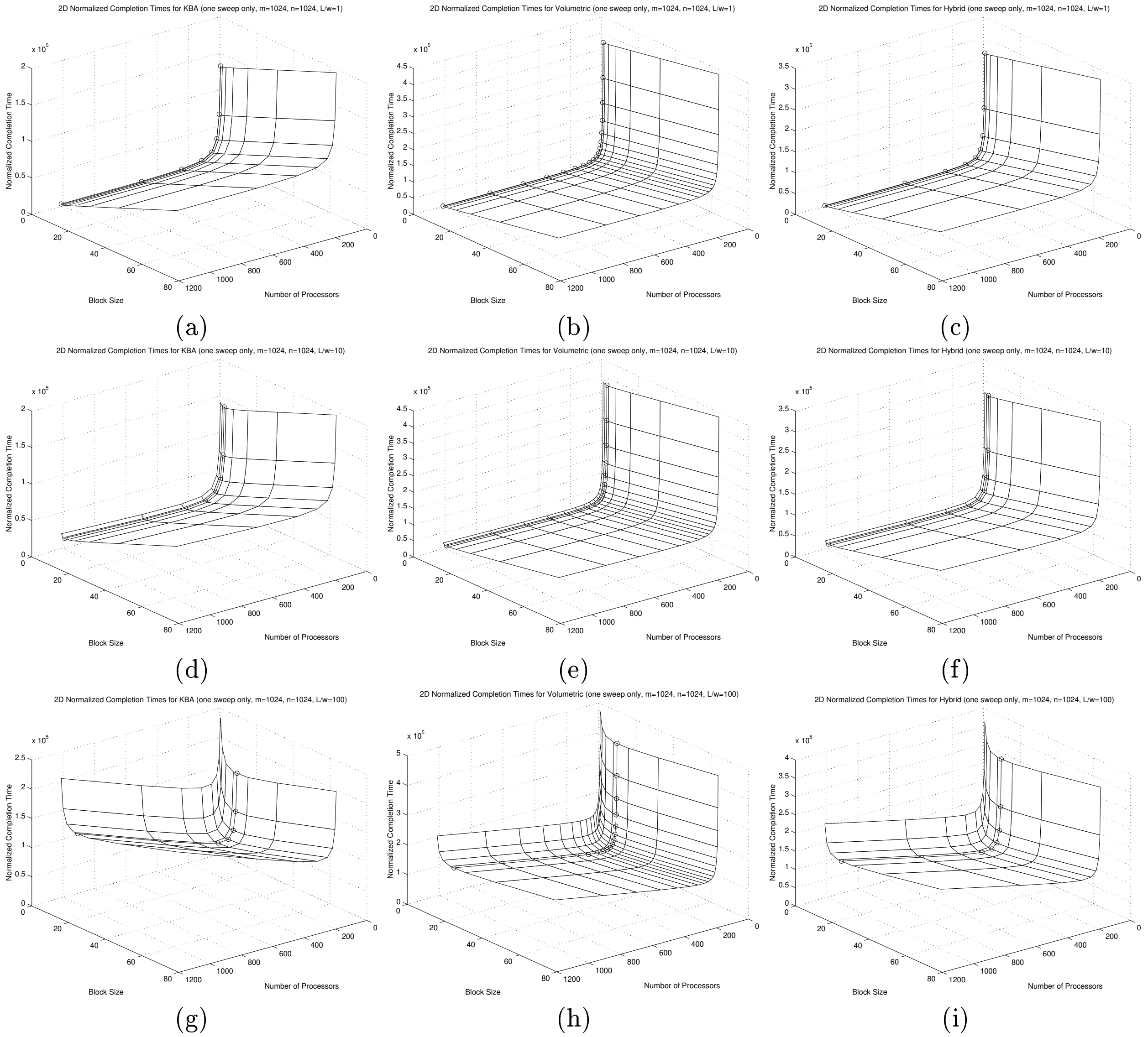


Figure 2: These 3D surfaces demonstrate the effect of the block size parameter k on the normalized completion times of KBA ((a), (d), and (g)), Volumetric ((b), (e), and (h)), and Hybrid ((c), (f), and (i)). These plots are for a single sweep in a square two-dimensional spatial domain ($m = n = 1024$) over $1 \leq k \leq 64$. We look at three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-c)), communication is moderately expensive ($L/\omega = 10$, (d-f)), and communication is relatively expensive ($L/\omega = 100$, (g-i)). The minimal normalized completion times obtained using k_{opt} are traced out as lines on the 3D surfaces.

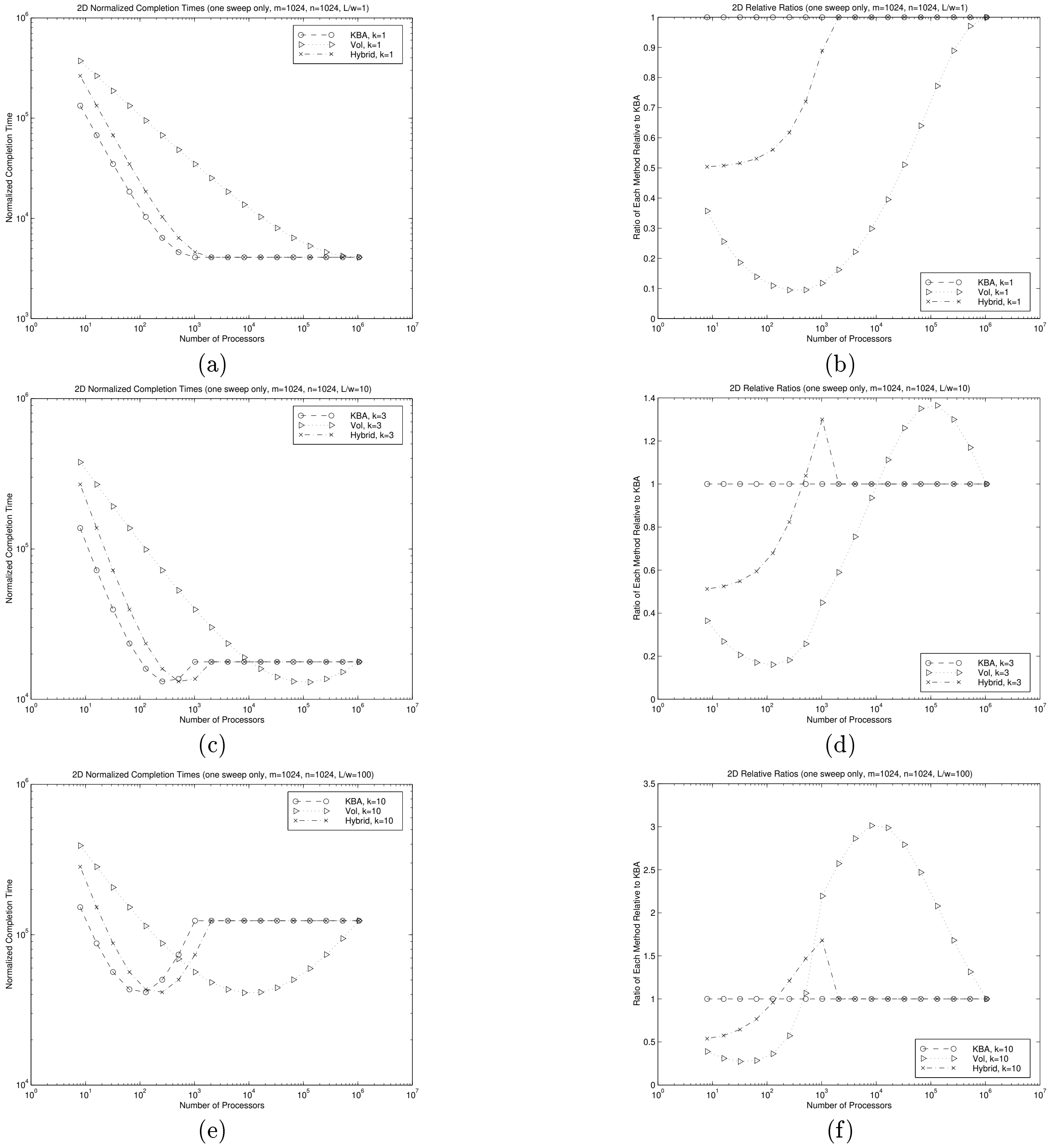
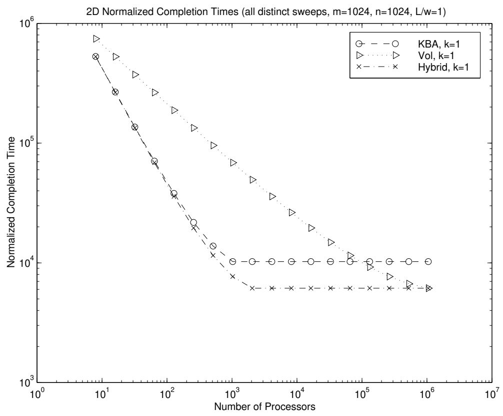
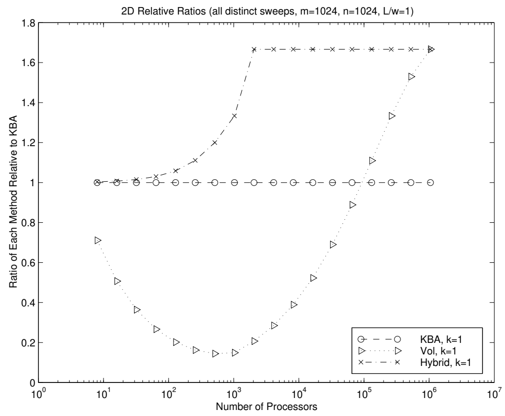
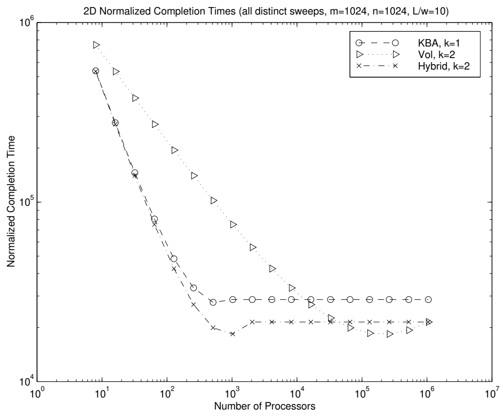


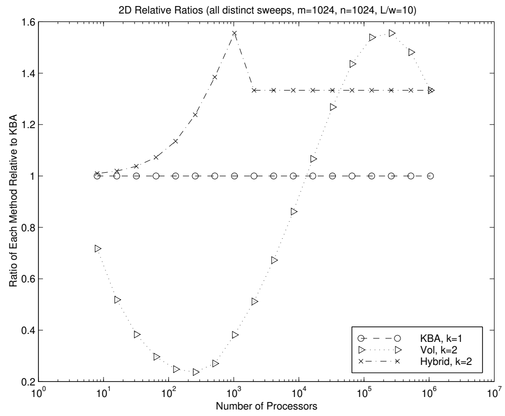
Figure 3: These plots consider a single sweep in a square two-dimensional spatial domain ($m = n = 1024$) and study the relative performance of the three decomposition methods analyzed (KBA, Volumnetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-b)), communication is moderately expensive ($L/\omega = 10$, (c-d)), and communication is relatively expensive ($L/\omega = 100$, (e-f)). For each case of L/ω, we show the normalized completion times of each method ((a), (c), and (e)) and relative ratios of the completion times with respect to the standard KBA algorithm ((b),(d), and (f)). The relative ratios were computed as $T_{\mathrm{KBA}}/T_{\mathrm{Method}}$ so values less than (more than) one indicate T_{Method} was slower (faster) than KBA.



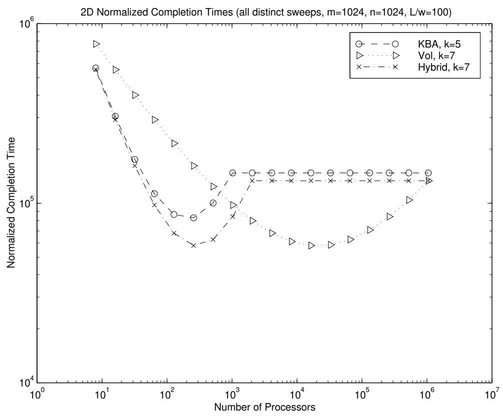


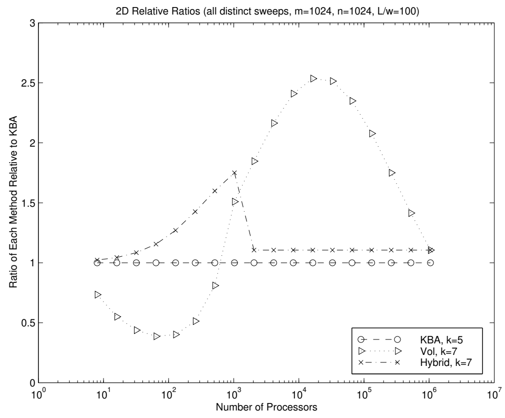
(a) (b)





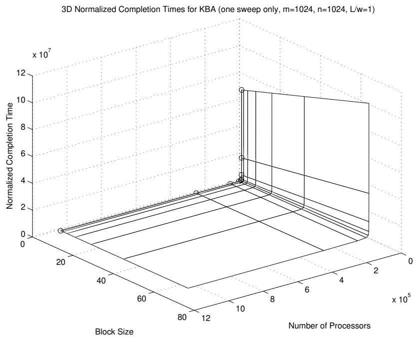
(c) (d)



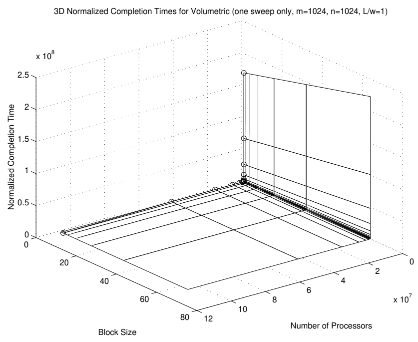


(e) (f)

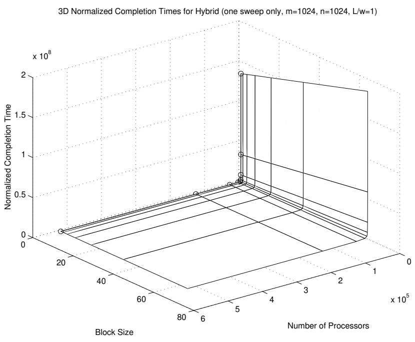
Figure 4: These plots consider all distinct sweeps, one for each quadrant, in a square two-dimensional spatial domain ($m = n = 1024$) and study the relative performance of the three decomposition methods analyzed (KBA, Volumnetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-b)), communication is moderately expensive ($L/\omega = 10$, (c-d)), and communication is relatively expensive ($L/\omega = 100$, (e-f)). For each case of L/ω, we show the normalized completion times of each method ((a), (c), and (e)) and relative ratios of the completion times with respect to the standard KBA algorithm ((b),(d), and (f)). The relative ratios were computed as $T_{\mathrm{KBA}}/T_{\mathrm{Method}}$ so values less than (more than) one indicate T_{Method} was slower (faster) than KBA.



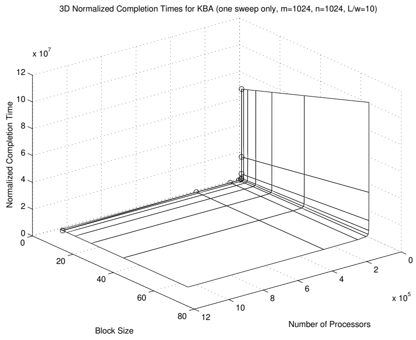
(a)



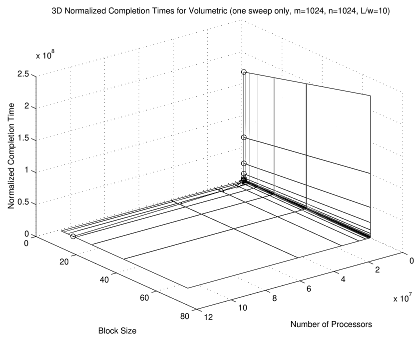
(b)



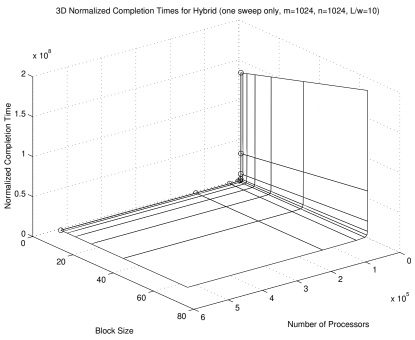
(c)



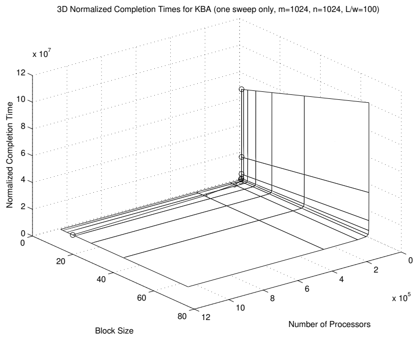
(d)



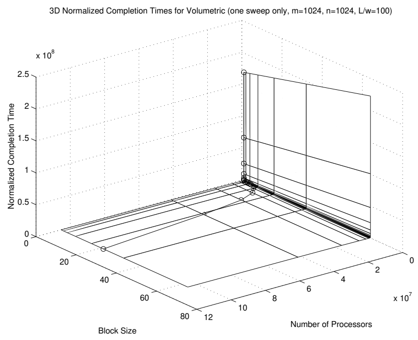
(e)



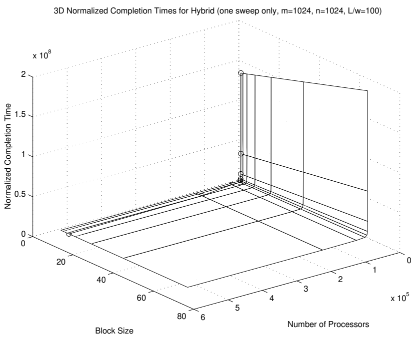
(f)



(g)



(h)



(i)

Figure 5: These 3D surfaces demonstrate the effect of the block size parameter k on the normalized completion times of KBA ((a), (d), and (g)), Volumetric ((b), (e), and (h)), and Hybrid ((c), (f), and (i)). These plots are for a single sweep in a cubic three-dimensional spatial domain ($m = n = h = 1024$) over $1 \leq k \leq 64$. We look at three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-c)), communication is moderately expensive ($L/\omega = 10$, (d-f)), and communication is relatively expensive ($L/\omega = 100$, (g-i)). The normalized completion times obtained using k_{opt} are traced out as lines on the 3D surfaces.

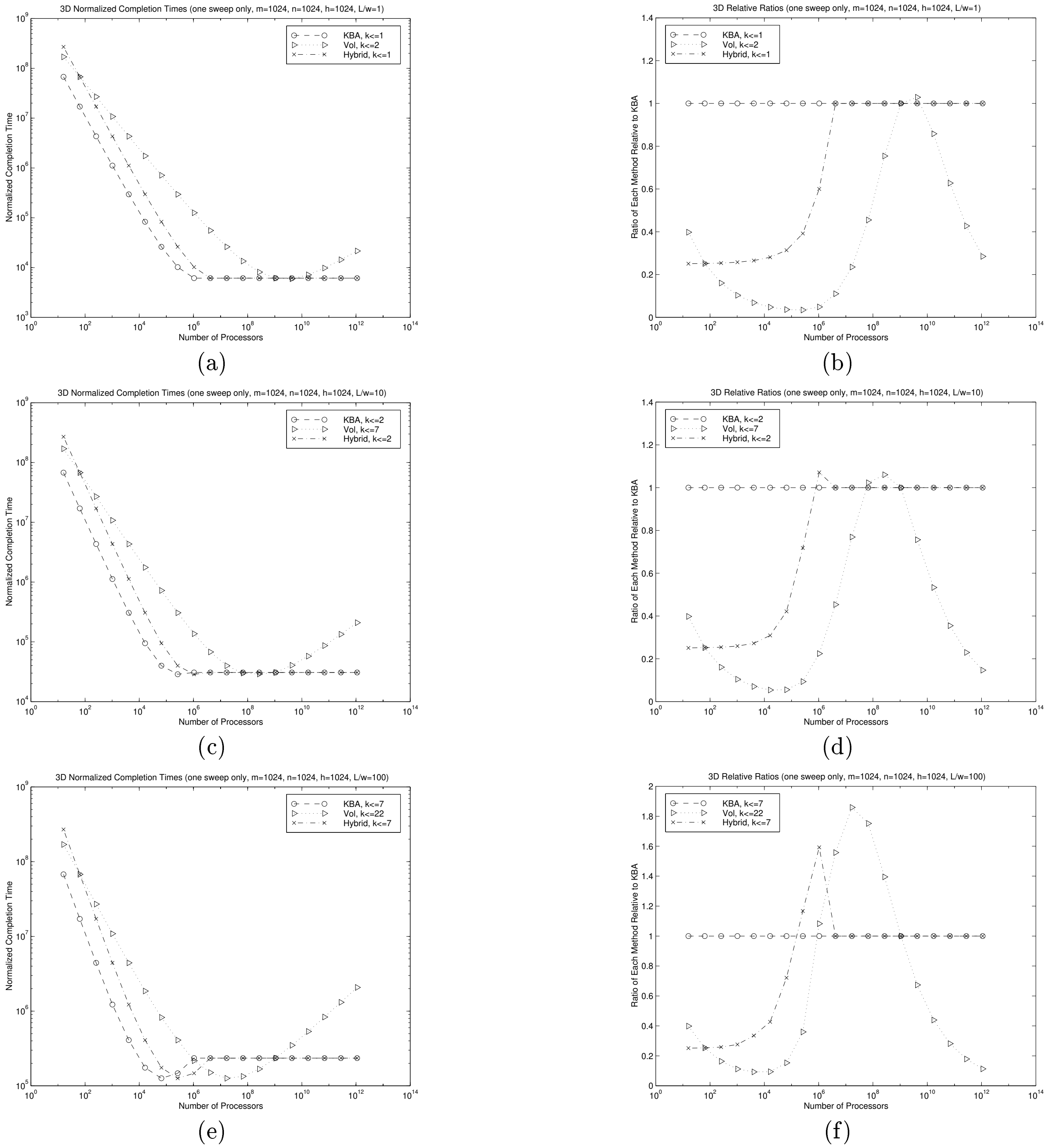
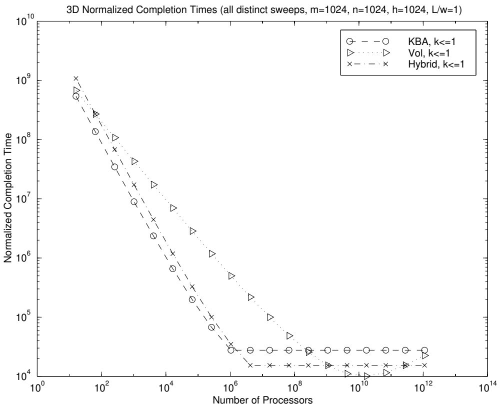
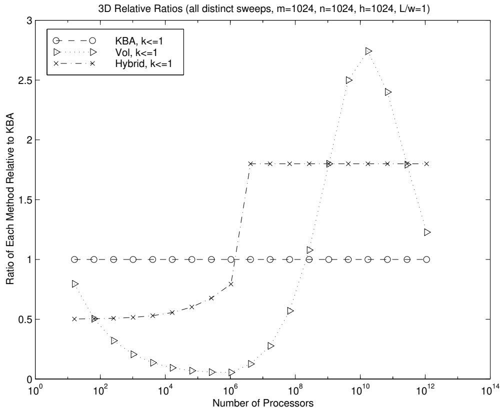


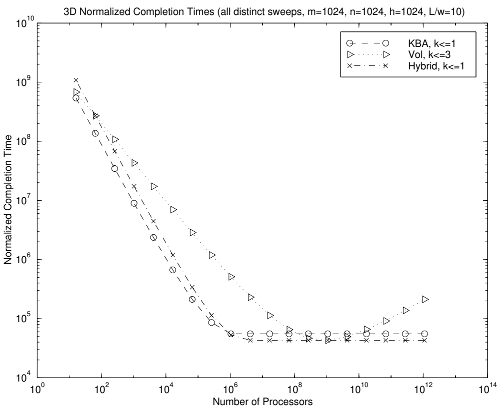
Figure 6: These plots consider a single sweep in a cubic three-dimensional spatial domain ($m = n = h = 1024$) and study the relative performance of the three decomposition methods analyzed (KBA, Volumnetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-b)), communication is moderately expensive ($L/\omega = 10$, (c-d)), and communication is relatively expensive ($L/\omega = 100$, (e-f)). For each case of L/ω, we show the normalized completion times of each method ((a), (c), and (e)) and relative ratios of the completion times with respect to the standard KBA algorithm ((b),(d), and (f)). The relative ratios were computed as $T_{\text{KBA}}/T_{\text{Method}}$ so values less than (more than) one indicate T_{Method} was slower (faster) than KBA.



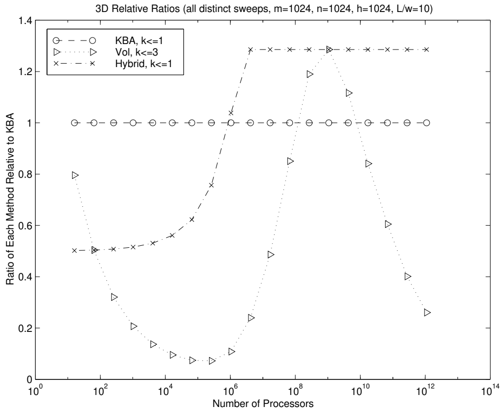
(a)



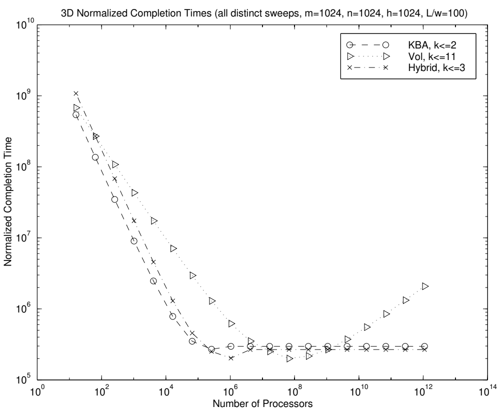
(b)



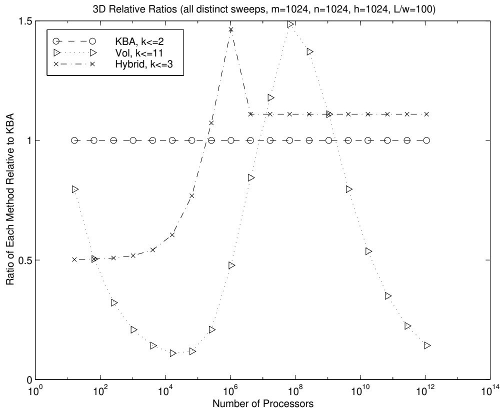
(c)



(d)



(e)



(f)

Figure 7: These plots consider all distinct sweeps, one for each octant of directions, in a cubic three-dimensional spatial domain ($m = n = h = 1024$) and study the relative performance of the three decomposition methods analyzed (KBA, Volumnetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-b)), communication is moderately expensive ($L/\omega = 10$, (c-d)), and communication is relatively expensive ($L/\omega = 100$, (e-f)). For each case of L/ω, we show the normalized completion times of each method ((a), (c), and (e)) and relative ratios of the completion times with respect to the standard KBA algorithm ((b),(d), and (f)). The relative ratios were computed as $T_{\mathrm{KBA}}/T_{\mathrm{Method}}$ so values less than (more than) one indicate T_{Method} was slower (faster) than KBA.

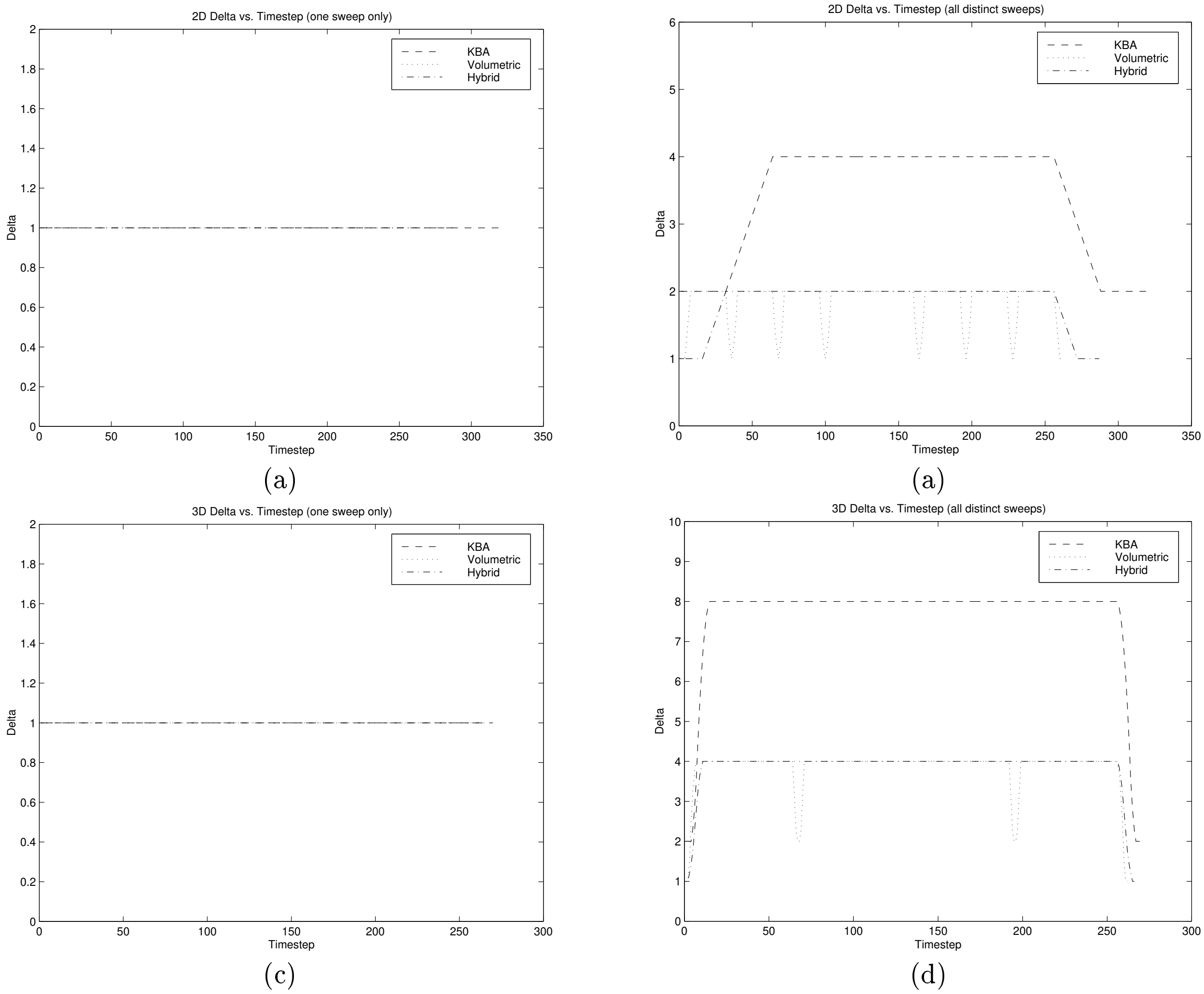


Figure 8: These plots show values of δ obtained by inspecting the grid at each step. These cases for all distinct sweeps, one for each quadrant and octant of directions, respectively, are shown in (b) and (d). Here we see that KBA typically has a δ equal to the number of sweeps, while Volumetric and Hybrid typically have δ values equal to one half of the number of sweeps. We include (a) and (c) to show that the single sweep case can be obtained using the multiple sweep equations with $\delta = 1$. These examples assume representative grids of size $m = n = 256$ with $p = 64$ processors in two dimensions and size $m = n = h = 256$ with $p = 64$ processors in three dimensions.