https://doi.org/10.2352/ISSN.2470-1173.2017.1.VDA-397
© 2017, Society for Imaging Science and Technology

Accelerating Advection Via Approximate Block Exterior Flow Maps

Ryan Bleile[1], *Linda Sugiyama*[2], *Christoph Garth*[3], *Hank Childs*[1]
[1] *University of Oregon, Eugene,*[2] *Massachusetts Institute of Technology,* [3] *University of Kaiserslautern*

Abstract

Flow visualization techniques involving extreme advection workloads are becoming increasingly popular. While these techniques often produce insightful images, the execution times to carry out the corresponding computations are lengthy. With this work, we introduce an alternative to traditional advection, which improves on performance at the cost of decreased accuracy. Our approach centers around block exterior flow maps (BEFMs), which can be used to accelerate flow computations by reducing redundant calculations. Our algorithm uses Lagrangian interpolation, but falls back to Eulerian advection whenever regions of high error are encountered. In our study, we demonstrate that the BEFM-based approach can lead to significant savings in time, with limited loss in accuracy.

Introduction

A myriad of scientific simulations, including those modeling fluid flow, astrophysics, fusion, thermal hydraulics, and others, model phenomena where constituents move through their volume. This movement is captured by a velocity field stored at every point on the mesh. Further, other vector fields, such as force fields for electricity, magnetism, and gravity, also govern movement and interaction. A wide range of flow visualization techniques are used to understand such vector fields. The large majority of these techniques rely on placing particles in the volume and analyzing the trajectories they follow. Traditionally, the particles are displaced through the volume using an advection step, i.e., solving an ordinary differential equation using a Runge-Kutta integrator.

As computational power on modern desktops has increased, flow visualization algorithms have been empowered to consider designs that include more and more particles advecting for longer and longer periods. Techniques such as Line Integral Convolution and Finite-Time Lyapunov Exponents (FTLE) seed particles densely in a volume and examine where these particles end up. For these operations, and many others, only the ending position of the particle is needed, and not the details of the path the particle took to get there.

Despite seemingly abundant computational power, some techniques have excessively long running times. For example, ocean modelers often study the FTLE within an ocean with both high seeding density and very long durations for the particles (years of simulation time) [2, 3]. As another example, fusion scientists are interested in FTLE computations inside a tokamak where particles travel for hundreds of rotations [1]. In both cases, FTLE calculations, even on supercomputers, can take tens of minutes.

With this work, we consider an alternative to traditional Eulerian advection. The key observation that motivates the work is that, in conditions with dense seeding and long durations, particles will tread the same (or very similar) paths over and over. Where the current paradigm carries out the same computation over and over, we consider a new paradigm where a computation can be carried out a single time, and then reused. That said, we find that, while particle trajectories do often travel quite close to each other, they typically follow their own (slightly) unique paths. Therefore, to effectively reuse computations, we consider a method where we interpolate new trajectories from existing ones, effectively trading accuracy for speed.

Our method depends on Block Exterior Flow Maps, or BEFMs. The idea behind BEFMs is to pre-compute known trajectories that lie on block boundaries. It assumes data is block-decomposed, but this assumption is common when dealing with parallel, distributed-memory computations. When a compute-intensive flow visualization algorithm is then calculated, it consults with the BEFMs and does Lagrangian-style interpolation from its known trajectories. While this approach introduces error, it can be considerably faster, since it avoids Eulerian advection steps inside each block.

The contributions of the paper are as follows:

- Introduction of BEFMs as an operator for accelerating dense particle advection calculations;
- A novel method for generating an approximate BEFM that can be used in practice;
- A study that evaluates the approximate BEFM approach, including comparisons with traditional advection.

Related Work

McLouglin et al. recently surveyed the state of the art in flow visualization [4], and the large majority of techniques they described incorporate particle advection. Any of these techniques could possibly benefit from the BEFM approach, although the tradeoff in accuracy is only worthwhile for those that have extreme computational costs, e.g., Line Integral Convolution [5], finite-time Lyapunov exponents [6], and Poincare analysis [7].

One solution for dealing with extreme advection workloads is parallelization. A summary of strategies for parallelizing particle advection problems on CPU clusters can be found in [8]. The basic approaches are to parallelize-over-data, parallelize-over-particles, or a hybrid of the two [9]. Recent results using parallelization-over-data demonstrated streamline computation on up to 32,768 processors and eight billion cells [11]. These parallelization approaches are complementary with our own. That is, traditional parallel approaches can be used in the current way, but the phase where they advect particles through a region could be replaced by our BEFM approach.

In terms of precomputation, the most notable related work comes from Nouanesengsy et al. [10]. They precomputed flow patterns within a region and used the resulting statistics to decide which regions to load. While their precomputation and ours have similar elements, we are using the results of the precomputation in different ways: Nouanesengsy et al. use precomputation for load balancing, while we use it to replace multiple integrations with one interpolation.

In terms of accelerating particle advection through approximation, two works stand out. Brunton et al. [18] also looked at accelerating FTLE calculation, but they considered the unsteady state problem, and used previous calculations to accelerate new ones. While this is a compelling approach, it does not help with the steady state problem we consider. Hlwatsch et al. [15] employ an approach where flow is calculated by following hierarchical lines. This approach is well-suited for their use case, where all data fits within the memory of a GPU, but it is not clear how to build and connect hierarchical lines within a distributed memory parallel setting. In contrast, our method, by focusing on flow between exteriors of blocks, is well-suited for this type of parallelism.

Bhatia et al. [19] studied edge maps, and the properties of flow across edge maps. While this work clearly has some similar elements to our, their focus was more on topology and accuracy, and less on accelerating particle advection workloads.

Scientific visualization algorithms are increasingly using Lagrangian calculations of flow. Jobard et al. [12] presented a Lagrangian-Eulerian advection scheme which incorporated forward advection with a backward tracing Lagrangian step to more accurately shift textures during animation. Salzbrunn et al. delivered a technique for analyzing circulation and detecting vortex cores given predicates from pre-computed sets of streamlines [14] and pathlines [13]. Agranovsky et al. [16] focused on extracting a basis of Lagrangian flows as an *in situ* compression operator, while Chandler at al. [17] focused on how to interpolate new pathlines from arbitrary existing sets. Of these works, none share our focus on accelerating advection.

Method

Our method makes use of *block exterior flow maps* (**BEFM**). We begin by defining this mapping. We then describe our method, and how it incorporates these maps.

Block Exterior Flow Map

Definition

In scientific computing, parallel simulation codes often partition their spatial volume over their compute nodes. Restated, each compute node will operate on one spatial region, and that compute node will be considered the “owner” of that region. Such a region is frequently referred to as a *block*. For example, a simulation over the spatial region X: [0-1], Y: [0-1], and Z: [0-1] and having N compute nodes could have N blocks, with each block covering a volume of $\frac{1}{N}$.

Consider a point P that lies on the exterior of a block B. If the velocity field points toward the interior of B at point P, then Eulerian advection of a particle originating at P will take the particle through the interior of B until it exits. In this case, the particle will exit B at some location P', where P' is also located on the exterior of B. The BEFM captures this mapping. The BEFM's domain is all spatial locations on the exterior of blocks, and its range is also spatial locations on the exteriors of blocks. Further, for any given P in the BEFM's domain, $BEFM(P,B)$ will produce a location that is on B's exterior. Saying it concisely, the BEFM is the mapping from particles at exteriors of blocks to the locations where those particles will exit the block under Eulerian advection. Figure 1 illustrates an example of a BEFM.

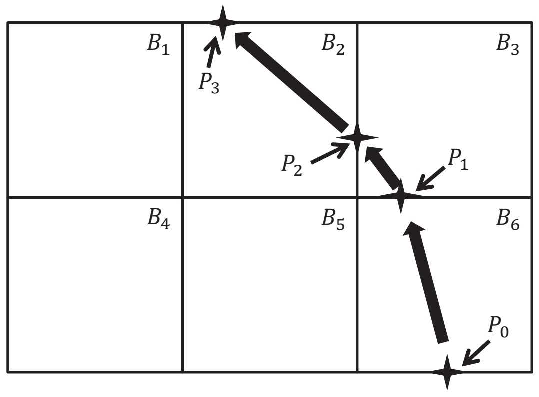

Figure 1. *Notional example of a BEFM on a two-dimensional vector field. This example shows the path of a particle moving through a region, with an emphasis on the blocks it travels through. Particle P_0 travels through block B_6 and exits B_6 at location P_1. Thus, $BEFM(P_0,B_6) = P_1$. Similarly, $BEFM(P_1,B_3) = P_2$, $BEFM(P_2,B_2) = P_3$, etc. In the case of particles placed in an outgoing region of flow, the BEFM returns the particle itself, e.g., $BEFM(P_1,B_6) = P_1$.*

Using BEFMs for Calculating Particle Trajectories

Now consider a particle P that lies on the interior of block B_0. Further, consider the trajectory of P when traveling for T time units. Assume P travels through blocks B_1, B_2, ..., B_{N-1}, before terminating in the interior of block B_N at time T. Consider how BEFMs can be used to calculate P's trajectory:

- Since P lies in the interior of B_0, traditional advection is needed to calculate the path of P until it reaches B_0's exterior.
- The BEFM can then be used to calculate the path of P through B_1, B_2, ..., B_{N-1}.
- P's trajectory into the interior of B_N is then again calculated with traditional advection.

Putting it all together, if BEFMs can calculate mappings more quickly than the calculations for advecting a particle through a block, then this method should be faster than traditional advection. Further, if a BEFM can calculate mappings instantaneously, then the speedup for the BEFM-style calculation would be limited only by the cost for the steps through the initial and final blocks (B_0 and B_N).

Approximate BEFMs

There are many ways to implement a BEFM. For example, a BEFM could respond to each mapping request (i.e., a $BEFM(P,B)$) by going back to the original vector field and employing traditional advection. In this case, the BEFM would have the same performance characteristics as traditional advection, and the abstraction of BEFMs on top of traditional advection would be unnecessarily complicated.

For our research, we are interested in BEFMs where each mapping request can be satisfied much more quickly than the work it takes to advect a particle using traditional advection. For this reason, we consider precomputation, i.e., evaluating the BEFM before the main work begins of calculating particle trajectories. However, it is not obvious how to precompute a perfect BEFM. Our approach to this problem is to precompute an Approximate BEFM or **ABEFM**. This ABEFM will know the exact mappings for certain locations on the boundary. We refer to this list of locations as the KnownParticleList.

When an ABEFM is asked to calculate mappings for particles that are not in the KnownParticleList, it will interpolate the exit location from the nearest particles that are in the KnownParticleList.

There are many ways to establish an ABEFM's KnownParticleList. We chose to generate locations uniformly along the exterior of a block at some chosen sample density. With this approach, the accuracy and pre-computation time are in tension. High sample densities will increase accuracy at the cost of pre-computation time. Low sample densities will reduce pre-computation time at the cost of accuracy.

Conditions Where an ABEFM Cannot Be Used

It is not always possible to interpolate new trajectories from the ABEFM's known trajectories. Through our experiments, we have identified three ways in which interpolation is not possible. They are listed below and illustrated in Figure 2:

1. If a particle trajectory from the exterior of block B never again reaches the exterior of block B, i.e., if a particle lands in a sink or is caught in a vortex inside the block.
2. If a particle trajectory differs too significantly from its neighbors, i.e., neighboring trajectories separated and exit through different faces of B.
3. If all neighboring trajectories are not uniformly entering the block or uniformly exiting the block, e.g., some neighboring particles get displaced to the interior of the block while others are displaced into neighboring blocks.

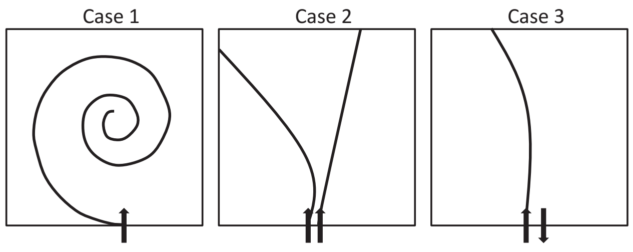

Figure 2. Cases where an ABEFM cannot interpolate a new trajectory. Note that on the right figure one of the particles enters the block while the other particle exits the block.

Fortunately, we can detect each of these three cases, and fall back to traditional advection to determine a particle trajectory. However, the rate at which the three cases occur is critical to understanding possible performance improvements. In our experiments, we determined these rate of the three cases occurring to be around 10% for our data sets.

An Approach for Creating and Using an ABEFM

In this section we describe our algorithms for creating an ABEFM and utilizing an ABEFM for advection.

Examples in this outline will follow the assumption that ABEFM's KnownParticleList points are uniformly generated at the mesh resolution, i.e., one particle trajectory for every node in the mesh that lies on the exterior of a block. For example, in a 10×10 two-dimensional mesh with 4 blocks laid in a 2×2 pattern, each block's external edge will consist of 5 cells and therefore 6 points. Additionally, these mapped points are not duplicated across shared faces. Figure 3 illustrates this example.

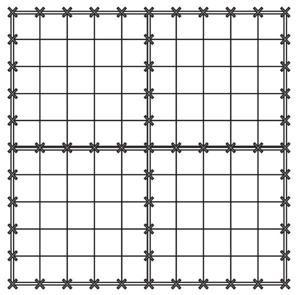

Figure 3. Initial locations for particle trajectories to be mapped during the pre-computation phase of an ABEFM. Depicted is a 10x10 mesh with 2x2 blocks overlaid and the locations of the mappings defined on the block's exteriors.

Building an ABEFM

ABEFM construction consists of generating flows for each location in the KnownParticleList. This is done by initializing particles at each location in the KnownParticleList and then advecting those particles across a block. Advection is done using traditional Eulerian methods such as Runge-Kutta. Pseudocode for this method is outline in Algorithm 1.

Algorithm 1 Build Flow Map

```
function GET BLOCK ID(Particle P)
    Determine the block that P advects through
    return BlockID
end function
function ADVECT ON BLOCK(Particle P, Block B)
    Advect P until it exits B (using Eulerian advection)
    Stop P on boundary of B
    Compute which Face of B that P landed on
    return P, FaceID
end function
for all P in KnownParticleList do
    Bid = GET BLOCK ID(P)
    NewP, Fid = ADVECT ON BLOCK(P, Bid)
    Def: Flow F as the set $< P, NewP, Bid, Fid >$
end for
```

Advecting With an ABEFM

An earlier section, Using BEFMs for Calculating Particle Trajectories, describes how to use a BEFM for particles at arbitrary locations in a volume. For this discussion, we focus on the case of a particle P that lies on the boundary of block B, and calculating where P exits B.

The trajectory for a particle P is calculated as follows. First, the neighboring particles, P_1, P_2, ..., P_n, from the KnownParticleList are identified. For our study, the KnownParticleList had particles seeded at regular intervals, so n would be four, and we would find the four particles that formed a square around P. Next, we check the P_i for our three conditions where an ABEFM cannot be used (see section "Conditions Where an ABEFM Cannot Be Used"). If we cannot use the P_i, then we fall back to traditional Eulerian advection using Runge-Kutta solves. If we can use the P_i, then we take the output location to be the weighted average of the exit locations of the P_i. For our construction of four P_i's in a square configuration, this can be accomplished with bilinear interpolation. We also interpolated the time to advance through the volume from the times of the P_i's. If this time was greater than the amount of time remaining for the particle to travel, then we reject the interpolated result (since it traveled too far), and fall back to Eulerian advection. However, if the interpolated projection was within the time bounds, then we use it and avoid Eulerian advection. Pseudocode for this method is outlined in Algorithm 2.

Algorithm 2 Advect with Flow Map

```
function ADVECT BLOCK(Particle P, Block B)
    Integrate to find P's exit location
    Stop P on boundary of B
    if P.Time $\geq$ End Time then
        return 0
    else
        return 1
    end if
end function
function ADVECT VIA FLOW MAP(Particle P, Block B)
    Interpolate Output location and time from (P,B)
    if Output.Time $>$ End Time then
        return ADVECT BLOCK(P,B)
    end if
    Set P = Output
    return 1
end function
AdvectionList: List of particles to be advected
for all Particles P in AdvectionList do
    keepGoing = 1
    while P.time $<$ End Time && keepGoing do
        Bid = GET BLOCK ID(P)
        if Particle on Computable Face then
            keepGoing = ADVECT VIA FLOW MAP(P,Bid)
        else
            keepGoing = ADVECT BLOCK(P, Bid)
        end if
    end while
end for
```

Study Overview

Data Sets

We considered three data sets. Each had steady state flow (i.e., one time slice) and was defined on a regular mesh. They are:

- **Tokamak**: the magnetic field inside a tokamak. Inside the tokamak, the velocity vector values lead to circulation around the tokamak. Outside the tokamak, the velocity field is all zero vectors. This data set had dimensions 300^3.
- **Astro**: a supernova simulation. The vector field has high variability in its central spherical region, and steadily points out or in when approaching the edges. This data set had dimensions 256^3.
- **TH**: a thermal hydraulics simulation of air mixing in a "fish tank" box with two inlets — one with hot air and one with cold air — and an outlet. This data set had dimensions 500^3.

Testing Factors

We considered six dimensions of configurations:

- Domain block layout: what are the impacts of having fewer or more blocks?
- Density of known particles: what are the impacts in calculating more or less particles during preprocessing? — time for preprocessing, time for regular execution, and accuracy?
- Integration time: how does performance and accuracy change as particles go for shorter or longer periods?
- Step size: how does step size affect performance and accuracy?
- Data set: how does the underlying vector field affect performance and accuracy?

Testing Methodology

Our methodology consisted of seven phases. The first phase studied our "default" case in detail. Each of the remaining six phases sweep through one dimension of our testing factors, and explores the impact of that factor by looking at performance and accuracy measurements. Performance is measured in terms of both run times for each method as well as the speedup of the ABEFM compared to Eulerian advection. Accuracy is measured in the following way. The locations of each particle advected by the ABEFM are compared to the locations of each particle advected by the traditional Eulerian method. The average distance between points is divided by the largest possible distance in the problem space to get a measure of the error, giving 1-error as a measure of accuracy.

Phase 1: Baseline Test

Our Baseline case combines a "default" advection workload with an ABEFM configuration. The default advection workload was a mesh resolution number of particles – a particle located at each node in the mesh – integrating for 10 time units with a step size of 0.001 on the vector field from the Tokamak data set. The default ABEFM configuration on the Tokamak was (10x10x10) blocks and a KnownParticleList with 300 particles in each dimension.

Phase 2: Block Layout

With this phase, we wanted to understand the effects of changing block size. Large blocks cause particles to travel larger distances, but the interpolated path may be less accurate. Small blocks cause particles to travel shorter distances – and so the number of operations needed to go the same distance is greater – but the interpolated path may be more accurate. With this phase, we wanted to understand the magnitude of these effects.

We considered 9 block layouts: (5x5x5), (10x10x10), (15x15x15), (20x20x20), (25x25x25), (30x30x30), (40x40x40), (50x50x50), and (100x100x100). Additionally we take a detailed look at a few more layouts designed around the motion in the underlying data set – since the tokamak data set has circular motion in the X and Y planes we used block layout targeting only those dimensions: (4x4x2), (5x5x2),..., (11x11x2).

Phase 3: Integration Time

With this phase, we considered integration time. Short integration times imply that we spend the majority of our time using traditional advection to get to block boundaries, mitigating the opportunity for speedup. Longer integration times, however, create the potential for applying the ABEFM repeatedly, and possibly significant speedups.

We considered 5 integration times: 1, 5, 10, 25, and 50 time units.

Phase 4: Step Size

With this phase, we considered step size. Small steps sizes move more slowly through a volume, while large step sizes move more quickly. However, for the ABEFM method, the step sizes only impact performance for stepping to the boundary, so the principal change is in the comparison with traditional advection.

We considered 7 step sizes: 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, and 0.0001.

Phase 5: Density of Known Particles

With this phase, we wanted to understand the effects of changing the number of known particles in the precomputation phase. Increasing this density will increase accuracy and the ability to use an ABEFM, but also increases precomputation costs. Decreasing this density could impact accuracy and decrease the ability to use an ABEFM, but reduces precomputation costs. With this phase, we again wanted to understand the magnitude of these effects.

We considered 5 densities along each dimension of the mesh: 100, 200, 300, 400, and 500.

Phase 6: Data Set

The performance of the ABEFM can clearly be affected by the underlying vector field. With this phase, we considered all three data sets. We performed the study from Phase 2 on each of the data sets keeping the total number of Eulerian steps constant. The Tokamak data set values are already listed in Phase 2. The Astro data set used an integration time of 5000 and a step size of 1. The TH data set used the same configuration as the Tokamak data set. Each data set used their own native resolution for precomputed particles for the KnownParticleList: 256 per dimension for Astro and 500 per dimension for TH. Each considered workloads of 20^3 particles.

Phase 7: Accuracy Performance Comparison

It is important to compare the accuracy and performance results of the ABEFM method with the traditional method. Since the traditional method already uses accuracy and performance tradeoffs, we need to understand how this tradeoff compares with that of the ABEFM. For this study, we hold the comparison step size fixed at 0.0001. We then run a series of Euler and ABEFM runs at different step sizes to compare the resulting differences in accuracy and performance. In doing this we will be able to understand the region of performance and accuracy where the ABEFM method will out perform the traditional approach.

Hardware

All studies were performed on a machine with dual 8-core 3.2GHz E5-2667 v3 Intel Xeon processors and 132 GB of RAM. This initial study was done on a single node, using 16 cores, with OpenMP parallelism over particles. Our goal was to enable the most direct comparisons between the ABEFM approach and traditional Eulerian integration methods, so the parallelism for each scheme was simplified to only OpenMP and looping over particles in each advection routine. Additionally, the Eulerian integration method used was a Runge-Kutta 4 integrator.

Measurements

The measurements we took for each experiment were:

- Time: the total run time of the ABEFM approach (meaning both build time and advection time using the ABEFM). We also would run a separate experiment with the traditional Eulerian approach and measure its time.
- Speedup: the total speed up from using an ABEFM compared to just Eulerian integration.
- Usability Metric: the percentage of time spent interpolating with the ABEFM versus using Eulerian integration and the percentage of faces for which an ABEFM can be used.
- Error: the average accuracy of advected particles comparing particle end locations of the ABEFM and Eulerian methods.

Results

Phase 1: Baseline Test Analysis

Phase 1 explores a single configuration, to set baseline expectations for the ABEFM method, and how the ABEFM method compares with traditional Eulerian advection. Table 1 shows the key results gathered in our study.

This baseline test demonstrates the viability of the ABEFM approach. Although the precomputation is non-trivial, it is still much smaller than the time to perform Eulerian advection. Additionally, the errors incurred were minimal — with the average less than 1% different from the Eulerian value. Figure 4 shows the FTLE computed using both the ABEFM method and traditional Eulerian advection showing that ABEFM version maintains all visible features distinguishable in the original Eulerian version.

Phase 2: Varying Domain Block Layouts

This phase studies the effect on ABEFM calculations when dividing the mesh into different numbers of blocks. Figure 5 shows tradeoffs in accuracy and speedup as the number of blocks increases. It shows that with the lowest number of blocks, both

Time (seconds)	
ABEFM Build Time	2.19
ABEFM Run Time	164.05
Eulerian Run Time	1426.35
Speedup	
ABEFM Run Time over Eulerian Time	8.69
ABEFM Total Time over Eulerian Time	8.58
Usability	
Percent Usable Faces	88.84%
Percent ABEFM Jumps	90.84%
Error	
Average Accuracy	99.30%
Average Error	0.70%
Max Error	30.43%
StdDev Error	2.56%

Table 1: Phase 1 Results

speedup and accuracy is quite good. By increasing the number of blocks, our accuracy drops off slightly and our run time slowly increases as the number of ABEFM jumps required to travel the same distance increases.

This configuration confirms that fastest run times with the Tokamak data set come from the block layouts with smaller numbers. In Figure 6 we look closer at block layouts based on the data set symmetry. Due to the cyclic nature of the data in the x and y dimensions, we reduced the number of blocks in the z dimension and varied only the x and y dimensions. Before these studies, our initial intuition was that the closer the block jumps are the less error there will be, but this was not true here. Having less block jumps can also decrease the error, as there are less interpolations that are approximating the flows locations and/or there are a greater percentage of Eulerian updates. This displays a trade-off between the number of times an error is introduced versus the size of the errors introduced.

Phase 3: Integration Time

This phase looked at performance and accuracy as particles were allowed to travel for longer and longer distances. Figures 7 and 8 show the results of this study. The take away from these figures is that, as integration time increases, the speedups from the ABEFM method become increasingly higher. While this is expected, the study shows the extent of speedup that is possible. Speedup is ultimately limited by the number of faces along a block that can be used for interpolation (and thus do not have to fall back to Eulerian advection). The choice to fall back to Eulerian advection produces an Amdahl's Law effect in our speedups as we get to higher and higher values of integration time.

Phase 4: Step Size

This phase varied the step size used for computing the Eulerian advection steps. This is used in building the BEFM and for all advections using the traditional Eulerian method. Figures 9 and 10 show the results of this phase.

Step size affects both the Eulerian method and the Eulerian portions of the ABEFM preprocessing phase. As step size decreases, the speedup increases.

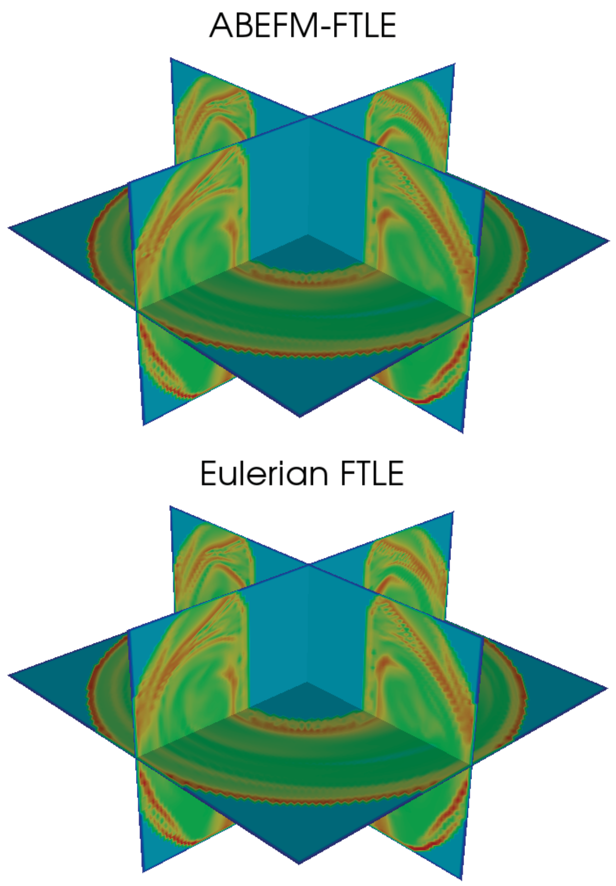

Figure 4. The FTLE field computed using the ABEFM (Top) and the Eulerian advection technique (Bottom).

Phase 5: Varying the KnownParticleList

Phase 5 varied the density of the KnownParticleList. Figure 11 shows the effect on accuracy and performance when this factor is varied. For this test, the average accuracy decreases more significantly as we vary the density below the mesh resolution and increases only gradually as we vary above the density of the mesh resolution. Additionally, the time to build the ABEFM increases with the increase in density of this list, though not significantly as this range spanned a time of 0.33 seconds at 100 points per dimension and 4.33 seconds at 400 points per dimension, growing steadily in between. Our increase in performance and therefore our increasing speedups are accounted for by the larger number of acceptable BEFM jumps that can be performed at higher densities with less increase in performance as we reach the point where we stop improving our mapping significantly.

Phase 6: Varying the Data Set

This study incorporated the remaining two data sets (TH and Astro) to see how well they performed compared to the Tokamak data set. The data sets were studied with a variety of blocks (i.e., the same study that was performed in Phase 2, but for these new data sets). For reference, the Tokamak data set's results for this analysis were listed in Figure 5.

In the Astro data set, the velocity field shows significant mix-

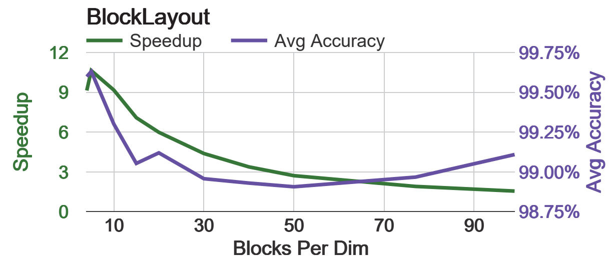

Figure 5. *Results from Phase 2: The accuracy and speedup for the Tokamak data set with respect to varying block dimensions.*

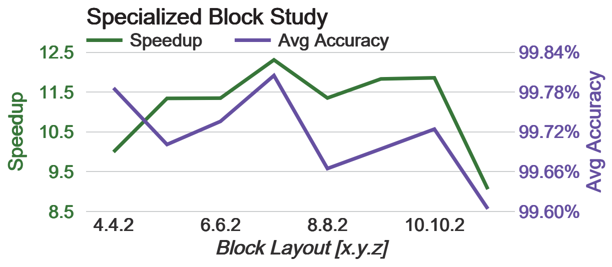

Figure 6. *Results from Phase 2: The accuracy and speedup for the Tokamak data set with respect to varying block dimensions. This study held the z component of the block layouts at its minimum value, 2, due to the symmetry present in the problem. Taking advantage of this property, we see a much higher accuracy and speedup, showing that we can fine tune the best block layout for a given problem.*

ing in the center and is headed straight out or straight in towards the edges of the domain. The result of varying block dimension can be seen in Figure 12. It shows an optimal layout for run time at around 20^3 resolution of blocks. The accuracy at this level is not significantly different then at other block sizes. This data set shows significantly less performance benefits as there are fewer blocks that have acceptable mappings for the ABEFM approach.

The second data set, TH, captures the mixing of hot and cold air currents. The vector field for this data set has significant mixing throughout its volume. The results of varying block dimension can be seen in Figure 13. The optimal layout for runtime is at around a block resolution of 20^3. For this problem we do not see a significant effect on accuracy with a change in the block layouts.

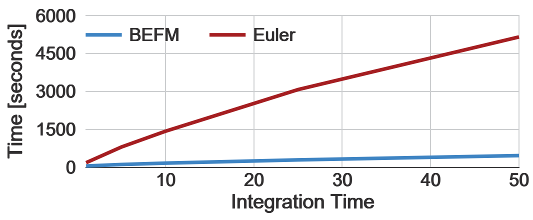

Figure 8. *Results from Phase 4: Runtime for both the ABEFM and Eulerian methods as a function of integration time. Values of ABEFM runtime range from 50.5 seconds to 462 seconds while Eulerian runtimes range from 180 seconds to 5,150 seconds.*

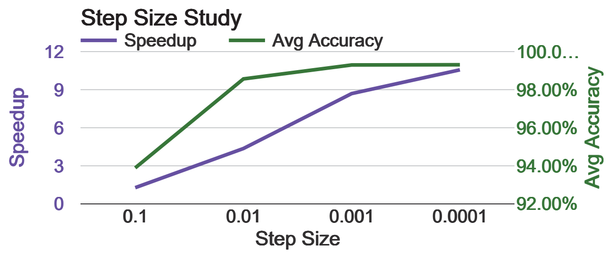

Figure 9. *Results from Phase 5: Speedup and Accuracy for the ABEFM method and Eulerian Method as a function of step size. Accuracy is calculated with respect to the Eulerian method running with the same step size.*

Phase 7: BEFM vs. Euler

The ABEFM method makes a tradeoff between accuracy and performance. Such a tradeoff is possible already within the traditional Eulerian approach, by increasing step size. In our final phase, we compare our approach with increased step sizes using the traditional approach.

Figure 14 shows a significant drop off in accuracy of the traditional method as step size increases, compared to the relatively small decrease in accuracy seen in the ABEFM method for the same changes. Figure 15 additionally shows that the ABEFM method reaches and maintains a higher speedup for all reasonable choices of step size. The performance gains of the ABEFM method begin immediately as step size increases, while maintaining a relatively accurate calculation. The Eulerian method however suffers from a immediate drop in accuracy while not seeing any performance benefits until much further from the baseline step size. In the long run the Eulerian method is more affected by the change in step size and so gains a significant speed im-

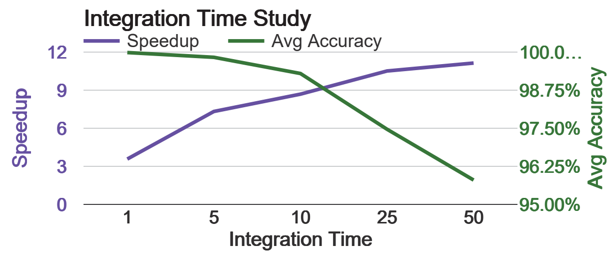

Figure 7. *Results from Phase 4: Speedup and accuracy for both the ABEFM and Eulerian methods as a function of integration time.*

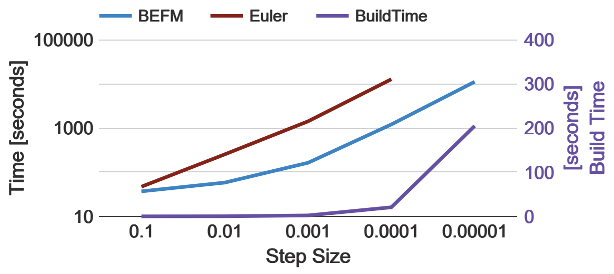

Figure 10. *Results from Phase 5: Runtimes and build time for the ABEFM method and Eulerian Method as a function of step size.*

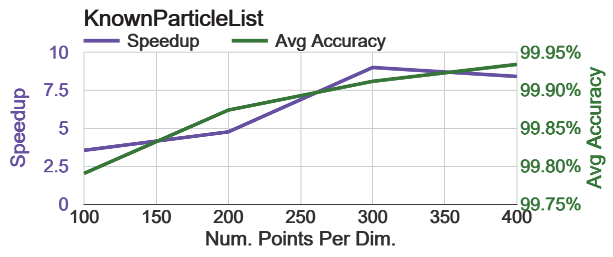

Figure 11. Results from Phase 4: Accuracy and speedup for varying the density of the KnownParticleList.

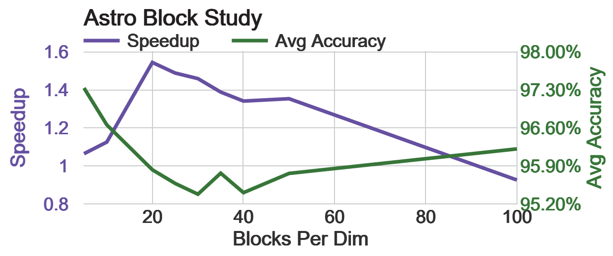

Figure 12. Results from Phase 5: Accuracy and runtime for the Astro data set as a function of varying block dimensions.

provement but by this point has given up a significant amount of accuracy to do so.

Conclusion and Future Work

We introduced Block Exterior Flow Maps (BEFMs) and designed an algorithm for accelerating flow calculations using Approximate BEFMs (ABEFMs). The approach has two significant controlling parameters — block layout and density of known particles calculated in the preprocessing phase — and we studied the impacts of these parameters for multiple particle advection workloads. We found that ABEFMs provided significant winnings for extreme particle advection workloads, with one workload completing in 159 seconds where the traditional approach took 3,320 seconds, a speedup of more than 20X and with an average error of less than 2%. Further, as particles are advected for longer and longer distances, our technique has the possibility to show even greater gains.

This technique was developed in response to needs within the fusion community to advect for long periods around a tokamak. They are interested in the steady-state problem, so that is all we considered in this initial work. While our technique is currently useful for stand-alone *post hoc* analysis, our future work will be to insert the method into their simulation codes for *in situ* processing. While our preprocessing times are currently large, we believe they can be accelerated on the many-core architectures now prevalent on top supercomputers. Further, our block-centric approach lends itself well to distributed memory parallelism. In another branch of future work, we would like to consider constructing the ABEFM adaptively, in an effort to minimize unneeded calculations, and to increase resolution in complex flow regions. We can leverage the work of Barakat and Tricoche for construction of a flow map through sparse adaptive sampling to accomplish this [20]. An additional branch of future work will be to consider ways of extending the ABEFM method to unsteady state problems as certain assumptions will need to be revisited for that case.

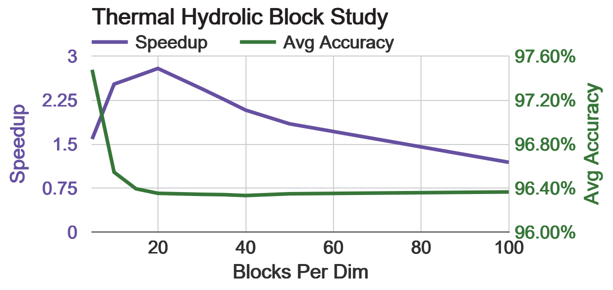

Figure 13. Results from Phase 5: Accuracy and runtime for the TH data set as a function of varying block dimensions.

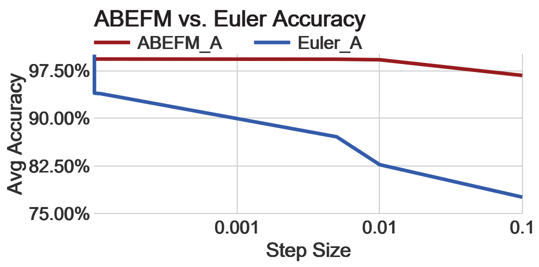

Figure 14. ABEFM and Euler accuracy at varying step sizes.

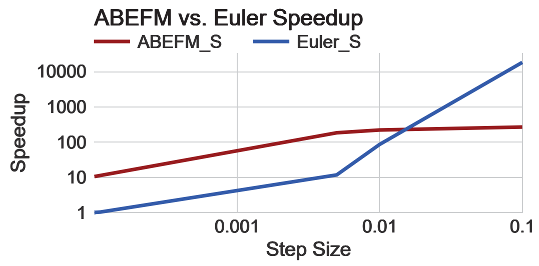

Figure 15. ABEFM and Euler Speedups at varying step sizes. Speedups are compared to the running time of the Eulerian problem with a step size of 0.0001.

Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the grant "Towards Exascale: High Performance Visualization and Analytics", Program Manager Lucy Nowell, and by the SciDAC Institute for Scalable Data Management, Analysis, and Visualization.

References

[1] Sugiyama, Linda and Krishnan, Harinarayan, Finite Time Lyapunov Exponents for magnetically confined plasmas, Bulletin of the American Physical Society, 57 (2012).

[2] Tamay M. Özgökmen and Andrew C. Poje and Paul F. Fischer and Hank Childs and Harinarayan Krishnan and Christoph Garth and Angelique C. Haza and Edward Ryan, On Multi-Scale Dispersion Under the Influence of Surface Mixed Layer Instabilities, Ocean Modeling, 56, 16-30 (oct 2012).

[3] Larry Pratt and Irina Rypina and Tamay Özgökmen and Peng Wang and Hank Childs and Yana Bebieva, Chaotic Advection in a Steady, Three-Dimensional, Ekman-Driven Eddy, Journal of Fluid Mechanics, 738, 143-183 (jan 2014).
[4] Tony McLoughlin and Robert S. Laramee and Ronald Peikert and Frits H. Post and Min Chen, Over Two Decades of Integration-Based, Geometric Flow Visualization, EuroGraphics 2009 - State of the Art Reports, April 2009, pg. 73
[5] Cabral, Brian and Leedom, Leith Casey, Imaging Vector Fields Using Line Integral Convolution, Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, 2014.
[6] G. Haller, Distinguished material surfaces and coherent structures in three-dimensional fluid flows, Physica D: Nonlinear Phenomena, 149, 248 - 277, (2001).
[7] Sanderson, Allen R and Chen, Guoning and Tricoche, Xavier and Pugmire, David and Kruger, Scott and Breslau, Joshua, Analysis of recurrent patterns in toroidal magnetic fields, Visualization and Computer Graphics, IEEE Transactions on, 16, 1431-1440 (2010).
[8] David Pugmire and Tom Peterka and Christoph Garth, Parallel Integral Curves, High Performance Visualization—Enabling Extreme-Scale Scientific Insight, 91-113 (oct 2012).
[9] David Pugmire and Hank Childs and Christoph Garth and Sean Ahern and Gunther H. Weber, Scalable Computation of Streamlines on Very Large Datasets, Proceedings of the ACM/IEEE Conference on High Performance Computing (SC09) (nov 2009).
[10] Boonthanome Nouanesengsy and Teng-Yok Lee and Han-Wei Shen, Load-Balanced Parallel Streamline Generation on Large Scale Vector Fields, IEEE Transactions on Visualization and Computer Graphics, 17, 1785-1794 (2011).
[11] Tom Peterka and Robert Ross and Boonthanome Nouanesengsey and Teng-Yok Lee and Han-Wei Shen and Wesley Kendall and Jian Huang, A Study of Parallel Particle Tracing for Steady-State and Time-Varying Flow Fields, Proceedings of IPDPS 11, Anchorage AK (2011).
[12] Jobard, B. and Erlebacher, G. and Hussaini, M.Y., Lagrangian-Eulerian advection of noise and dye textures for unsteady flow visualization, IEEE Transactions on Visualization and Computer Graphics, 8, 211-222 (2002).
[13] Salzbrunn, Tobias and Garth, Christoph and Scheuermann, Gerik and Meyer, Joerg, Pathline predicates and unsteady flow structures, The Visual Computer, 24, 1039-1051 (2008).
[14] Salzbrunn, T. and Scheuermann, G., Streamline Predicates, IEEE Transactions on Visualization and Computer Graphics, 12, 1601-1612 (2006).
[15] Hlawatsch, M. and Sadlo, F. and Weiskopf, D., Hierarchical Line Integration, Visualization and Computer Graphics, IEEE Transactions on, 17, 1148-1163 (2011).
[16] Alexy Agranovsky and David Camp and Christoph Garth and E. Wes Bethel and Kenneth I. Joy and Hank Childs, Improved Post Hoc Flow Analysis Via Lagrangian Representations, Proceedings of the IEEE Symposium on Large Data Visualization and Analysis (LDAV), 67-75 (nov 2014).
[17] Chandler, Jennifer and Obermaier, Henriette and Joy, Kenneth and others, Interpolation-based pathline tracing in particle-based flow visualization, IEEE Transactions on Visualization and Computer Graphics, 21, 68-80 (2015).
[18] Brunton, Steven and Rowley, Clarence, A method for fast computation of FTLE fields, APS Division of Fluid Dynamics Meeting Abstracts, 1, (2008).
[19] Bhatia, Harsh and Jadhav, Shreeraj and Bremer, P and Chen, Guoning and Levine, Joshua A and Nonato, Luis Gustavo and Pascucci, Valerio, Flow visualization with quantified spatial and temporal errors using edge maps, IEEE Transactions on Visualization and Computer Graphics, 18, 1383–1396 (2012).
[20] S. Barakat and X. Tricoche. Sparse Adaptive Sampling for Scalable Flow Map Computation. IEEE Transactions on Visualization and Computer Graphics, 19(12), 27532762, 2013.

Author Biography

Ryan Bleile received his BS in Computer Science and Physics from the University of the Pacific (2013) and is a PhD student in Computer Science at the University of Oregon, Eugene. Ryan is currently a Lawrence Graduate Scholar at Lawrence Livermore National Laboratory working on research in the field of Monte Carlo Particle Transport for next generation architectures.

Linda Sugiyama received a BS in the Applied Mathematics Engineering Physics program at the University of Wisconsin-Madison in 1975 and a PhD in Applied Mathematics from M.I.T. in 1980. Since then, she has been a research scientist at M.I.T., where she has worked in a variety of areas in plasma physics related to magnetically confined fusion. A major focus of her work has been the development and application of large scale nonlinear MHD simulations of plasma instabilities and "extended MHD" nonlinear models that include important particle motion effects.

Christoph Garth is an Assistant Professor in the Computer Science Department at the University of Kaiserslautern, Germany, from which he also received his Ph.D. in Computer Science in 2007. Christophs research is focused on topology-based methods and high-performance computing techniques with applications to visualization and data analysis.

Hank Childs is an Associate Professor in the Computer and Information Science Department at the University of Oregon. He received his Ph.D. in computer science from the University of California at Davis in 2006. Hank's research focuses on scientific visualization, high performance computing, and the intersection of the two.