VisGap'24 – The Gap between Visualization Research and Visualization Software (2024)
C. Gillmann, M. Krone, G. Reina, T. Wischgoll (Editors)

Performance Improvements of Poincaré Analysis for Exascale Fusion Simulations

D. Pugmire[1], J. Y. Choi[1], S. Klasky[1], K. Moreland[1], E. Suchyta[1], T. M. Athawale[1], Z. Wang[1], C.-S. Chang[2], S.-H. Ku[2], and R. Hager[2]

[1]Oak Ridge National Laboratory, USA
[2]Princeton Plasma Physics Laboratory, USA

Abstract

Understanding the time-varying magnetic field in a fusion device is critical for the successful design and construction of clean-burning fusion power plants. Poincaré analysis provides a powerful method for the visualization of magnetic fields in fusion devices. However, Poincaré plots can be very computationally expensive making it impractical, for example, to generate these plots in situ during a simulation. In this short paper, we describe a collaboration among computer science and physics researchers to develop a new Poincaré tool that provides a significant reduction in the time to generate analysis results.

CCS Concepts

• ***Human-centered computing*** → ***Scientific visualization;*** *Visualization techniques;* • ***Applied computing*** → *Physics;*

1. Introduction

Fusion energy research is focused on understanding the science needed to develop energy sources based on the controlled fusion of light atomic nuclei. Fusion energy is an active area of research that promises to provide large amounts of clean energy. One promising avenue for achieving fusion energy is using a machine called a tokamak. A tokamak confines a plasma using magnetic fields within a torus. Scientists around the world are actively researching high-fidelity models to predict the performance and behavior of fusion in tokamak devices. Significant efforts are currently underway to plan experiments on the International Thermonuclear Experimental Reactor (ITER) tokamak under construction in France. ITER, and follow-on devices, will operate in physics regimes not achieved by any current or past experiments, making advanced and predictive numerical simulation critical for success.

The tokamak contains many magnetic coils producing a strong magnetic field that confines the charged particles within the plasma. One set of magnetic coils generates an intense *torroidal* field, in the direction around the torus. A second set of coils generates a magnetic *poloidal* field in the direction around a cross-section of the torus. These two fields result in a twisted magnetic field that twists the plasma, resulting in confinement. Additional coils are used to drive the shape of the plasma during confinement.

The time-varying behavior of the magnetic fields is complex and vital to the performance of the tokamak. This makes analysis tools for understanding the dynamic nature of the magnetic field critical. The complexity of the three-dimensional, periodic magnetic field lines makes direct visualization difficult. Because the field lines are periodic, the complexity can be reduced by making use of a recurrence or Poincaré map. This technique reduces the dimension of each field line from three down to two by intersecting the periodic orbit with a lower-dimensional subspace (called the Poincaré section). Here we intersect each field line with a plane to create a sequence of points. The pattern produced by each field line provides valuable insight into the topology of the complex magnetic field lines inside the plasma.

Proper characterization of the magnetic field requires a large number of orbits to a large number of field lines. It is common to create Poincaré maps containing tens of thousands of field lines, each consisting of thousands of orbits. In practice, the field lines are computed using particle advection to trace the path of massless particles through the magnetic field. Calculating the orbit of each field line consists of taking a large number of advection steps using a numerical solver. To ensure the accuracy of such long field lines, an appropriately small step size must be used for the numerical solver. This in turn results in a large number of solves for each orbit. As an example, for an orbit that takes 500 advection steps, computing 1000 orbits results in 500,000 advection steps for each field line. For 50,000 field lines, this results in a total of $2.5x10^{10}$ advection steps.

In this short paper, we describe a collaboration among computer science and physics researchers to provide a significant reduction in the time required to perform Poincaré analysis on the magnetic fields in plasma simulations using the X-Point Gyrokinetic Code (XGC) [HYK*16] code. Before the collaboration began, the tool used by the physicists to perform this analysis took

DOI: 10.2312/visgap.20241120

up to several hours. Because of this, the amount of analysis that can be performed was very limited. Further, because the magnetic field evolves over the course of the simulation, a detailed analysis of the time-varying evolution was not possible. This collaboration resulted in a new Poincaré tool being developed using VTK-m [MSU*16], a visualization toolkit that provides portability across CPUs and GPUs. This new tool was run on four different workloads using both CPUs and GPUs on a modern supercomputer. It achieved time speedups of between 5.8× and 8.9× on the CPU and between 11.4× and 15.4× on the GPU. In addition, because the new Poincaré tool required fewer resources, the cost of running was significantly lower. The cost savings for the four workloads were between 14.4× and 22.4× for the CPUs and between 58.4× and 77.1× for the GPU.

One of the major challenges in this collaboration was ensuring that the complex representation in the simulation code of the magnetic field was accurately conveyed in the new visualization tool. The VTK-m accelerated tool has made it possible for Poincaré analysis to be performed in situ for large-scale simulations running on supercomputers located at the Oak Ridge Leadership Computing Facility (OLCF) [SKP*22].

2. Related Work

2.1. Poincaré Maps

Poincaré maps are an important tool developed for the study of dynamical systems. One of the key advantages is that it provides a dimension reduction to study periodic systems [Mor00]. In the case of our fusion example, this reduction is from 3D to 2D. Löeffelmann et al. [LKG97] provide a summary of the usage of Poincaré maps for visualization across a variety of application areas and demonstrate their usefulness in flow analysis. Poincaré maps have been applied to fusion in a number of works, including the following [SCT*10, SCTC12, TGS11].

2.2. Streamlines and Particle Advection

A widely used visualization algorithm for flow fields is a technique called streamlines [SML04]. The technique starts with a velocity vector field representing the flow of a fluid at each point in the domain, which is a common output from computational fluid dynamics simulations. This fluid is visually represented by one or more curves that trace the trajectory of that part of the flow. This is modeled and computed as a massless particle instantaneously moving with the velocity determined by the field at the particle's position. The particle is placed at a seed point and then is advected by pushing it by the vector field. The computation (described in more detail in Section 3) solves a differential equation to find the curve that is tangent to the vector field everywhere.

Although these traces of advected particles can be visualized directly as streamlines, they also form the basis of numerous visualization algorithms, some of which require the advection of a great many particles [Hul92, GHP*16]. Because particle advection is the greatest computational cost of these algorithms, much research has been invested in optimizing this process [ZY18, YSB*23]. The work in this paper leverages the particle advection provided by the VTK-m library [MSU*16]. VTK-m provides a flexible particle advection algorithm that is optimized for a variety of processors [PYK*18].

Technically, a magnetic field is not a flow field; it does not describe the movement of matter. However, we are interested in extracting magnetic field lines: curves that are tangent to the magnetic field everywhere. This is the same property a particle advection trajectory has to its velocity field, and thus it is valid to extract these magnetic field lines by treating magnetism as a flow and leveraging the aforementioned particle advection algorithms. This paper often refers to flowing particles even though that does not match the physical process.

3. Methods

Computing the Poincaré map consists of two basic steps. First, compute field lines originating from each seed location. Second, intersect the fieldlines with the Poincaré section. Fieldlines, or streamlines, are generated by computing the trajectory of a massless particle through a vector field. This is a classic initial value problem, $\frac{dy}{dt} = f(t,y)$, where $y(t_0) = y_0$. Here, y_0 is the (seed) location at time t_0, and $f(t,y)$ is the vector field. For streamline computation, a common method for computing the trajectory of the particle $(y(t))$, is using the 4^{th} order Runge-Kutta iterative method [PTVF92]. For a given step size $h > 0$, positions along the trajectory (y_i) can be computed by:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1}$$

where

$$\begin{aligned} k_1 &= f(t_n, \quad y_n) \\ k_2 &= f(t_n + \frac{h}{2}, \, y_n + h\frac{k_1}{2}) \\ k_3 &= f(t_n + \frac{h}{2}, \, y_n + h\frac{k_2}{2}) \\ k_4 &= f(t_n + \frac{h}{2}, \, y_n + hk_3) \end{aligned}$$

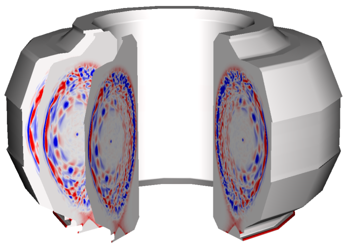

Figure 1: *Example of an XGC mesh that has been clipped to show the interior. 2D planes are equally spaced around the central axis of the tokamak. Two planes are shown in the clipped region to illustrate the semi-unstructured nature of the mesh.*

The key requirement of the RK4 method is to be able to evaluate $f(t,y)$ (i.e., the magnetic field) at each point within the domain.

The XGC code uses a semi-unstructured mesh to represent the physics of the plasma. Fundamentally, the mesh is cylindrical with periodic boundary conditions along the cylinder's axis. The mesh is structured along the axis of the cylinder and each toroidal cross-section is a 2D unstructured triangular mesh. The cross-section is rotated around the central axis to create the 3D mesh (see Figure 1). The mesh consists of wedge elements that are formed from pairs of triangles from adjacent planar cross sections.

3.1. Poincaré Algorithm

Algorithm 1 Algorithm for computing a Poincaré map.

```
1:  /* Initialization */
2:  $P$ = Poincaré section
3:  $S$ = Input seed locations
4:  $Max_p$ = Maximum number of punctures
5:  $Result = Empty$
6:
7:  for all $s$ in $S$ do
8:      /* Compute the Poincaré map for the field line at $s$ */
9:      $n = 0$
10:     $p_0 = s$
11:
12:     /* Iterate until max number of punctures computed */
13:     while $n < Max_p$ do
14:         $p_1 = \mathrm{RK4Solve}(p_0)$
15:         $L$ = line segment from $p_0$ to $p_1$
16:         if $L$ intersects $P$ then
17:             Add intersection of $L$ and $P$ to $Result$
18:             $n = n + 1$
19:         end if
20:         $p_0 = p_1$
21:     end while
22: end for
```

Algorithm 1 contains pseudocode for computing the Poincaré map. Initialization consists of specifying the initial seed locations, defining the Poincaré section (i.e., 2D plane), and the maximum number of punctures to compute as well as initializing a structure to store punctures (Lines 2 - 5). For each seed location s (Line 7), the fieldline is iteratively computed using a RK4 solver. At each iteration of the solver, the line segment between the current location and the previous location is checked against the Poincaré section to determine if an intersection occurred (Line 16). If so, the intersection point is added to the result. This process continues until the maximum number of punctures occurs, or the particle terminates. For brevity, the code to check for particle termination is not shown in the code listing.

3.2. Implementation in VTK-m

VTK-m uses a data parallel abstraction to achieve high performance on many-core devices [MMP*21]. Given a vector of data, VTK-m will apply a function to each element of the array. These functions often come in the form of "functors," which is an object that can be called like an ordinary function. The input data to the VTK-m implementation of the Poincaré algorithm is an array of initial particle positions. The functor object is initialized with the Poincaré section, RK4 step size and the maximum number of punctures. The execution portion of the functor consists of lines 7- 22 in Algorithm 1. At runtime, each input particle and the functor are mapped to a thread and executed on the specified device, effectively making the for loop parallel.

The bottleneck of the algorithm is not the arithmetic used to compute Equation 1 for the RK4 solution but rather the field evaluation to get the numbers to plug into the equation. At each iteration of the algorithm, the cell containing the current location of the point must be determined. Because the mesh is unstructured, this requires a cell search. This search can be optimized by taking advantage of the toroidal symmetry of the mesh. Using this symmetry, the point can be specified by (x, y, ϕ), where x and y are coordinates of the 2D cross-section and ϕ is the angle around the torus. The point can be easily projected onto the 2D cross section by setting $\phi = 0$ (i.e., $(x, y, 0)$). The cell-finding operation is now reduced to two dimensions.

To accelerate the cell-finding operation for unstructured grids, VTK-m provides two different acceleration structures. The first uses two levels of structured grids [KBS11]. The first level is a coarse grid of bins that covers the spatial extents of the data. Each bin within the first level contains a finer grid where the number of bins is proportional to the number of cells contained in the first level grid. Each bin in the fine-level grids contains a list of unstructured cells that are contained. The second acceleration structure is much simpler. It uses a single fine-structured grid. As before, each bin contains a list of the unstructured cells that are contained. For each method, the 2D cross-section mesh is passed in, and the acceleration structure is built. The acceleration structure is passed into the VTK-m functor to be used for field evaluations.

The major costs for either structure are the time to retrieve the appropriate bin and the time to search through the cells in the bin. The advantage of the simpler uniform bin approach is that the appropriate bin can be found with a single lookup rather than the two-part lookup of the two-level grid. However, the two-level grid avoids bins containing large numbers of unstructured cells when the unstructured cells are unevenly distributed.

In either case, whenever the algorithm looks up a cell, it remembers both the unstructured cell found and the bin containing it. Subsequent field evaluations will be nearby and are often in the same cell or a nearby one. Before searching for a bin, the algorithm first checks the saved unstructured cell for containment and then the saved bin.

3.3. Field Evaluation

For many applications, the vector field values are defined at the vertices of each cell. Evaluation at a location within the cell interpolates the values at each of the vertices. However, the magnetic field in XGC is much more complicated. Because calculating fieldlines contains many iterative steps, accuracy is paramount as small errors can rapidly grow and produce incorrect results.

In XGC, the magnetic field vector consists of an axisymmetric (i.e., defined at the cross-sectional plane and constant along the cylindrical axis) background field, B_0 and a non-axisymmetric perturbation, δB. For a particle located at (x, y, ϕ), the magnetic field is sum of the background field and the perturbation: $B_0(x, y) +$

$\delta B(x,y,\phi)$. The background field, B_0, is evaluated using cubic spline interpolation using the $2D$ coordinates (x,y). The perturbation is represented as the component of the magnetic vector potential parallel to the background magnetic field $A_{||}(x,y,\varphi)$, on a set of N_φ planar, unstructured triangle meshes, where N_φ is the toroidal resolution of the XGC simulation. In practice, for large runs, N_φ is between 32 and 64, depending on the physics being studied. The corresponding perturbed magnetic field is $\delta B = \nabla \times (A_{||}B_0/|B_0|)$. For interpolation of data between planes, the magnetic vector potential and its derivatives are interpolated linearly along the magnetic field line through the position of the particle. This requires interpolation on two cross-sectional planes, which is also linear. To minimize interpolation errors, the mesh in XGC is approximately aligned with the magnetic field such that a field line starting on a vertex on one plane maps along B_0 to a position at or very close to a mesh vertex on the adjacent plane. For a more detailed explanation is given in [MHC*19, HO13, HYK*16].

Accurately calculating the non-axisymmetric perturbation of the magnetic field, (δB), was the most challenging part of the collaboration. The XGC implementation of the Poincaré tool is embedded within the XGC simulation code. The Poincaré tool is run by using an input deck for the simulation that specifies how to configure and run the analysis. As such, the code for performing a full simulation and Poincaré analysis share some non-trivial overlap that must be carefully separated. Further, given that XGC is written in Fortran, and VTK-m is written in C++, the code separation must be done carefully and is easy to get wrong. We instrumented the XGC code with debug logs to check the calculation of key quantities in the derivation of the magnetic field at each step of a point along a field line. The coordinates of individual field lines were also saved so that they could be compared (both visually and textually) to the coordinates generated by the VTK-m implementation. When differences in coordinates were detected, we were able to compare, step by step, the computation of the different values associated with the calculation of B_0 and δB in both codes and determine where the errors occurred. Once these were identified and fixed, we ran both the XGC and VTK-m implementations of the code over many different seeding scenarios and physics parameters and compared the resulting Poincaré plots to ensure they were identical.

In an attempt to optimize the calculation of the perturbed magnet field, we pre-computed $B_0(x,y) + \delta B(x,y,\phi)$ at each vertex in the $3D$ mesh to avoid the complexity of the field evaluation at every RK4 iteration. However, the effects of linear interpolation of the values within the 3D elements of the mesh over a large number of iterations resulted in errors that compounded over thousands of RK4 steps and produced significantly inaccurate results. In the end, this optimization had to be abandoned.

4. Results

The experiments were performed on the Frontier [fro] supercomputer at Oak Ridge National Laboratory. Frontier is a 1.102 ExaFLOPS supercomputer consisting of 9472 nodes. Each node contains a 64-core AMD Epyc CPU and four Radeon Instinct MI250X GPUs.

To study the performance, we ran both the XGC and VTK-m

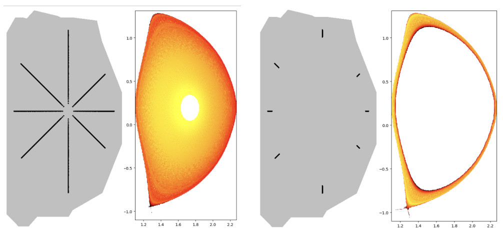
(a) Whole seeding (b) Edge seeding

Figure 2: *Seed placement allows for analysis of different parts of the plasma. Whole seeding along with the resulting Poincaré map is shown in (a), and the edge seeding and results are shown in (b).*

versions of the Poincaré tool. We use the XGC implementation as a baseline. While the XGC simulation code can run on GPUs, the XGC Poincaré analysis tool is only implemented on the CPUs. The VTK-m implementation was run on both the CPU and GPU. The results from the runs on CPUs give us a direct comparison between the XGC analysis code and the VTK-m equivalent. The runs on the GPU highlight the added benefits afforded by the portability of the VTK-m implementation.

The XGC implementation was run on the CPUs of five Frontier nodes and uses OpenMP for parallelization across the multi-core CPUs. The VTK-m implementation was run in two configurations – CPU and GPU. The CPU implementation was run on two Frontier nodes and the GPU implementation was run on a single node using two of the four GPUs.

We used two variations in the number of seeds and two variations in the placement of the seeds for a total of four different configurations. The seeds are placed along regular angular intervals around the center of the cross-section. We used a total of eight angular intervals and placed 1280 and 3200 seeds along each angular interval. This gives seed numbers of 10,240 and 25,600. We also varied the placement of the seeds along each angular interval. For whole seeding (see Figure 2(a)), the seeds are placed uniformly from the center of the cross-section to the edge. For edge seeding (see Figure 2(b)), the seeds are placed further away from the center, or nearer to the edge of the plasma. Whole seed placement is used to analyze the overall nature of the magnetic field in the plasma. The placement of seeds along the edge is a common way to analyze the very turbulent regions of the plasma that occur along the edge region. Figure 3 shows a zoomed-in view of Poincaré maps generated from two different time steps using edge seeding. The image is zoomed in to view the bottom left of the cross-section, called the ”X” point, where turbulence in the plasma is complex.

Table 1 contains the timing results for the XGC tool and VTK-m implementations for each configuration. We also ran the VTK-m implementations using both the Two-Level and Uniform Bins cell locators, but the performances were similar (within 1.5% of each other). The results we report are using the Uniform Bins cell locator. Because the triangles in the cross-sectional grid are uniformly distributed, the added complexity of the Two-Level grid does not provide any benefit. We performed some tuning on the optimal size of the Uniform Bin acceleration structure and found that a

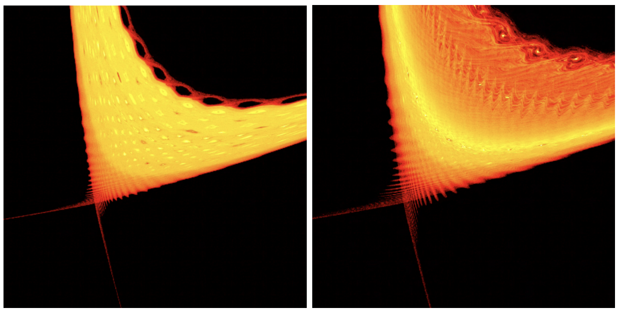

Figure 3: *Poincaré maps generated from two time steps using edge seeding. Zooming in to regions of interest makes it possible to see topological features that develop as the plasma develops.*

Table 1: *Time (in seconds) for XGC and VTK-m runs on CPUs and GPUs for both Whole and Edge seeding with* 10,240 *and* 25,600 *seeds. The time speedup factors for the VTK-m runs on the CPU and GPU are given in two right-most columns.*

Num Seeds	Seeding	XGC Time	VTK-m Time CPU	VTK-m Time GPU	Speedup CPU	Speedup GPU
10240	Whole	2001.8	249.0	171.5	**8.0**	**11.7**
10240	Edge	1762.6	197.0	154.3	**8.9**	**11.4**
25600	Whole	3785.8	655.5	245.7	**5.8**	**15.4**
25600	Edge	3435.3	522.0	223.1	**6.6**	**15.4**

size of 12,000×12,000 generally provided the best results. The speedups for the VTK-m implementation are significant. Running on the CPU, the VTK-m implementation provides between 5.8× and 8.9× speedup over the XGC implementation. On the GPU, VTK-m achieves between 11.4× and 15.4× speedup over the XGC implementation. Speedups on the GPU are limited by two factors. First, the number of threads that can be utilized. The computation is parallelized over the seeds, and the number of seeds commonly used for Poincaré maps is not enough to saturate the number of threads available on the GPUs. Second, and much more importantly, the cell-finding operation for unstructured grids is bound by the cost of memory access and *not* computation. As the seeds are traced, they circulate throughout the volume in non-uniform ways. Because the acceleration structure used for cell-finding is too large to be kept in fast GPU memory, there is significant time spent accessing data in slower memory spaces.

Table 2 contains a comparison of the total cost (in node-seconds) for the XGC and VTK-m implementations for each configuration. The XGC tool was run on five nodes of Frontier and so the XGC Cost column in Table 2 is 5× the runtime reported in Table 1. For the VTK-m implementation, the CPU runs were performed on two nodes of Frontier, and the GPU runs were performed on two GPUs within a single node. Because of this difference in the number of nodes used, the cost savings for the VTK-m implementation are more pronounced than the speedups. For the CPU runs, the VTK-m implementation running on two nodes, results in cost savings between 14.4× and 22.4×. For the GPU runs, the VTK-m implementation results in cost savings between 57.1× and 77.1×. We calculate the cost factor for VTK-m using one node, although it

Table 2: *Total cost (in node-seconds) for XGC and the savings factor for the VTK-m runs on the CPU and GPU. The XGC runs used five nodes and the VTK-m used two Frontier nodes for the CPU runs and one node for the GPU runs.*

Num Seeds	Seeding	XGC Cost	VTK-m Cost Savings CPU	VTK-m Cost Savings GPU
10240	Whole	10009.2	**20.1**	**58.4**
10240	Edge	8813.2	**22.4**	**57.1**
25600	Whole	18929.1	**14.4**	**77.1**
25600	Edge	17176.3	**16.5**	**77.0**

technically uses only half of the node, which results in cost savings between 114.2× and 154.2×.

The use of fewer nodes becomes even more important when doing in situ processing. XGC uses the in-transit processing model [CAA*20, CCD*18] for in situ analysis and visualization tasks. In this model, additional nodes are allocated for analysis and visualization. After the simulation completes a time step, the data are transferred to the additional nodes for asynchronous processing. For simulations that could run for days or weeks, the use of fewer nodes for analysis and visualization results in even more significant cost savings.

5. Summary and Future Work

In this paper, we have described a collaborative effort among computer scientists and physicists to develop a new tool for performing analysis and visualization of the magnetic fields in fusion devices. Poincaré plots are notoriously expensive to generate due to high computational costs. Because of the complexity of the magnetic fields in tokamaks, the definition of the vector field needed for generating field lines is often code-specific and complex. Working together with the XGC physics team, we developed a tool that reproduces the complex definition of the magnetic field and uses the concepts in VTK-m to efficiently map the computation onto both multi-core CPU and GPU devices. The development of this new tool has made it possible for physicists to perform unprecedented analysis on XGC simulations running on some of the most powerful supercomputers in the world. Previously, it was only possible to perform Poincaré analysis on a limited number of timesteps from a simulation. This work has made it possible to generate Poincaré plots in situ for every time step of the simulation. This provides the XGC team with a powerful new tool to study the physics of burning plasmas.

Although we have achieved significant speedups in computing Poincaré maps, there is still room for additional performance. The bottleneck in the algorithm is the cost of cell location. There are two major ways to address this problem. The first is to avoid performing cell locations using an adaptive step-sized solver. These techniques improve over the brute-force nature of fixed step-size methods to adjust according to the local changes in the vector field. We plan to explore the use of these solvers to increase performance while maintaining accuracy. The second is to improve the performance of the cell location on GPUs. The costs for memory access can vary dramatically on the GPU depending on where the memory is located. We plan to explore methods to achieve better caching of data in faster memory to improve overall performance.

Acknowledgements

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work was supported in part by the U.S. Department of Energy (DOE) RAPIDS SciDAC project under contract number DE-AC05-00OR22725 and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

[CAA*20] Childs H., Ahern S. D., Ahrens J., Bauer A. C., Bennett J., Bethel E. W., Bremer P.-T., Brugger E., Cottam J., Dorier M., Dutta S., Favre J. M., Fogal T., Frey S., Garth C., Geveci B., Godoy W. F., Hansen C. D., Harrison C., Hentschel B., Insley J., Johnson C. R., Klasky S., Knoll A., Kress J., Larsen M., Lofstead J., Ma K.-L., Malakar P., Meredith J., Moreland K., Navrátil P., O'Leary P., Parashar M., Pascucci V., Patchett J., Peterka T., Petruzza S., Podhorszki N., Pugmire D., Rasquin M., Rizzi S., Rogers D. H., Sane S., Sauer F., Sisneros R., Shen H.-W., Usher W., Vickery R., Vishwanath V., Wald I., Wang R., Weber G. H., Whitlock B., Wolf M., Yu H., Ziegeler S. B.: A terminology for in situ visualization and analysis systems. *The International Journal of High Performance Computing Applications 34*, 6 (2020), 676–691. URL: https://doi.org/10.1177/1094342020935991, doi:10.1177/1094342020935991. 5

[CCD*18] Choi J. Y., Chang C.-S., Dominski J., Klasky S., Merlo G., Suchyta E., Ainsworth M., Allen B., Cappello F., Churchill M., Davis P., Di S., Eisenhauer G., Ethier S., Foster I., Geveci B., Guo H., Huck K., Jenko F., Kim M., Kress J., Ku S.-H., Liu Q., Logan J., Malony A., Mehta K., Moreland K., Munson T., Parashar M., Peterka T., Podhorszki N., Pugmire D., Tugluk O., Wang R., Whitney B., Wolf M., Wood C.: Coupling exascale multiphysics applications: Methods and lessons learned. In *2018 IEEE 14th International Conference on e-Science (e-Science)* (2018), pp. 442–452. doi:10.1109/eScience.2018.00133. 5

[fro] Frontier supercomputer. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html. Accessed: 2024-02-13. 4

[GHP*16] Guo H., He W., Peterka T., Shen H.-W., Collis S. M., Helmus J. J.: Finite-time lyapunov exponents and lagrangian coherent structures in uncertain unsteady flows. *IEEE Transactions on Visualization and Computer Graphics 22*, 6 (June 2016), 1672–1682. doi:10.1109/TVCG.2016.2534560. 2

[HO13] Hariri F., Ottaviani M.: A flux-coordinate independent field-aligned approach to plasma turbulence simulations. *Computer Physics Communications 184*, 11 (Nov. 2013), 2419–2429. URL: http://dx.doi.org/10.1016/j.cpc.2013.06.005, doi:10.1016/j.cpc.2013.06.005. 4

[Hul92] Hultquist J. P. M.: Constructing stream surfaces in steady 3D vector fields. In *Proceedings Visualization '92* (Oct. 1992), pp. 171–178. doi:10.1109/VISUAL.1992.235211. 2

[HYK*16] Hager R., Yoon E., Ku S., D'Azevedo E., Worley P., Chang C.: A fully non-linear multi-species fokker–planck–landau collision operator for simulation of fusion plasma. *Journal of Computational Physics 315* (2016), 644–660. URL: https://www.sciencedirect.com/science/article/pii/S0021999116300298, doi:https://doi.org/10.1016/j.jcp.2016.03.064. 1, 4

[KBS11] Kalojanov J., Billeter M., Slusallek P.: Two-level grids for ray tracing on gpus. *Comput. Graph. Forum 30* (04 2011), 307–314. doi:10.1111/j.1467-8659.2011.01862.x. 3

[LKG97] Löffelmann H., Kucera T., Gröller E.: *Visualizing Poincare Maps together with the underlying flow*. Tech. Rep. TR-186-2-97-06, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/E193-02, A-1040 Vienna, Austria, Mar. 1997. human contact: technical-report@cg.tuwien.ac.at. URL: https://www.cg.tuwien.ac.at/research/publications/1997/Loeffelmann-1997-VPM/. 2

[MHC*19] Moritaka T., Hager R., Cole M., Lazerson S., Chang C.-S., Ku S.-H., Matsuoka S., Satake S., Ishiguro S.: Development of a gyrokinetic particle-in-cell code for whole-volume modeling of stellarators. *Plasma 2*, 2 (May 2019), 179–200. doi:10.3390/plasma2020014. 4

[MMP*21] Moreland K., Maynard R., Pugmire D., Yenpure A., Vacanti A., Larsen M., Childs H.: Minimizing development costs for efficient many-core visualization using mcd3. *Parallel Computing 108* (09 2021), 102834. doi:10.1016/j.parco.2021.102834. 3

[Mor00] Morrison P.: Magnetic field lines, hamiltonian dynamics, and nontwist systems. *Physics of Plasmas 7*, 6 (2000), 2279–2289. 2

[MSU*16] Moreland K., Sewell C., Usher W., Lo L.-t., Meredith J., Pugmire D., Kress J., Schroots H., Ma K.-L., Childs H., et al.: VTK-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE computer graphics and applications 36*, 3 (2016), 48–58. 2

[PTVF92] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes in C*, second ed. Cambridge University Press, Cambridge, USA, 1992. 2

[PYK*18] Pugmire D., Yenpure A., Kim M., Kress J., Maynard R., Childs H., Hentschel B.: Performance-portable particle advection with VTK-m. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (June 2018), pp. 45–55. doi:10.2312/pgv.20181094. 2

[SCT*10] Sanderson A., Chen G., Tricoche X., Pugmire D., Kruger S., Breslau J.: Analysis of recurrent patterns in toroidal magnetic fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1431–1440. doi:10.1109/TVCG.2010.133. 2

[SCTC12] Sanderson A., Chen G., Tricoche X., Cohen E.: Understanding quasi-periodic fieldlines and their topology in toroidal magnetic fields. *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications* (2012), 125–140. 2

[SKP*22] Suchyta E., Klasky S., Podhorszki N., Wolf M., Adesoji A., Chang C., Choi J., Davis P. E., Dominski J., Ethier S., Foster I., Germaschewski K., Geveci B., Harris C., Huck K. A., Liu Q., Logan J., Mehta K., Merlo G., Moore S. V., Munson T., Parashar M., Pugmire D., Shephard M. S., Smith C. W., Subedi P., Wan L., Wang R., Zhang S.: The exascale framework for high fidelity coupled simulations (EFFIS): Enabling whole device modeling in fusion science. *The International Journal of High Performance Computing Applications 36*, 1 (2022), 106–128. URL: https://doi.org/10.1177/10943420211019119, doi:10.1177/10943420211019119. 2

[SML04] Schroeder W., Martin K., Lorensen B.: *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*, fourth ed. Kitware Inc., 2004. ISBN 1-930934-19-X. 2

[TGS11] Tricoche X., Garth C., Sanderson A.: Visualization of topological structures in area-preserving maps. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1765–1774. doi:10.1109/TVCG.2011.254.

[YSB*23] YENPURE A., SANE S., BINYAHIB R., PUGMIRE D., GARTH C., CHILDS H.: State-of-the-art report on optimizing particle advection performance. *Computer Graphics Forum 42*, 3 (2023), 517–537. doi:https://doi.org/10.1111/cgf.14858. 2

[ZY18] ZHANG J., YUAN X.: A survey of parallel particle tracing algorithms in flow visualization. *Journal of Visualization 21* (Feb. 2018), 351–368. doi:10.1007/s12650-017-0470-2. 2