

Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms

S Ethier¹, W M Tang¹, Z Lin²

¹ Princeton Plasma Physics Laboratory, Princeton NJ 08543

² University of California - Irvine, Irvine CA 08543

Abstract. Since its introduction in the early 1980s, the gyrokinetic particle-in-cell (PIC) method has been very successfully applied to the exploration of many important kinetic stability issues in magnetically confined plasmas. Its self-consistent treatment of charged particles and the associated electromagnetic fluctuations makes this method appropriate for studying enhanced transport driven by plasma turbulence. Advances in algorithms and computer hardware have led to the development of a parallel, global, gyrokinetic code in full toroidal geometry, the Gyrokinetic Toroidal Code (GTC), developed at the Princeton Plasma Physics Laboratory. It has proven to be an invaluable tool to study key effects of low-frequency microturbulence in fusion plasmas. As a high-performance computing applications code, its flexible mixed-model parallel algorithm has allowed GTC to scale to over a thousand processors, which is routinely used for simulations. Improvements are continuously being made. As the U.S. ramps up its support for the International Tokamak Experimental Reactor (ITER), the need for understanding the impact of turbulent transport in burning plasma fusion devices is of utmost importance. Accordingly, the GTC code is at the forefront of the set of numerical tools being used to assess and predict the performance of ITER on critical issues such as the efficiency of energy confinement in reactors.

1. Introduction

Plasmas comprise over 99% of the visible universe and are rich in complex, collective phenomena. A major component of research in this area is the quest for harnessing fusion energy, the power source of the sun and other stars, which occurs when forms of the lightest atom, hydrogen, combine to make helium in a very hot (100 million degrees centigrade) ionized gas or "plasma." The development of a secure and reliable energy system that is environmentally and economically sustainable is a truly formidable scientific and technological challenge facing the world in the twenty-first century. This demands basic scientific understanding that can enable the innovations to make fusion energy practical. The "computational grand challenge" nature of fusion energy science is a consequence of the fact that in addition to dealing with vast ranges in space and time scales which can span over ten decades, the fusion-relevant problem involves extreme anisotropy, the interaction between large-scale fluid-like (macroscopic) physics and fine- scale kinetic (microscopic) physics, and the need to account for geometric detail. Moreover, the requirement of causality (inability to parallelize over time) makes this problem among the most challenging in computational physics.

In 2003, the US Secretary of Energy Spencer Abraham outlined the Department of Energy's Office of Science 20-year science facility plan [1]. This plan provides a roadmap for future scientific facilities to support the department's research missions and prioritizes 28 major

facilities, among which 12 are considered near-term and critical to the US scientific future. Within this context, the number one priority was identified as the ITER project [2], an international effort to build a fusion experiment that will study the so-called “burning plasma” regime (see Figure 1) - a regime corresponding to the expected operational conditions of real fusion power plants of the future with more energy being produced than what is provided to maintain the fusion reactions. Extrapolation of the performance of existing fusion experiments to

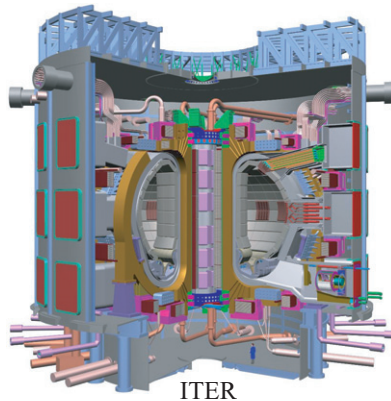


Figure 1. Current design of the ITER project tokamak. In 2003, ITER was identified as the number one DOE priority among major new scientific facilities.

future burning plasma systems such as ITER currently relies heavily on empirical scaling trends. Accelerated development of computational tools and techniques are needed to develop predictive models which can prove superior to empirical scaling. This will have a major impact on the fusion community’s ability to effectively harvest the key physics from ITER. The probability that ITER will achieve its goals can be significantly enhanced by development of the capability to numerically simulate the associated plasma behavior under realistic conditions. Unraveling the complex behavior of strongly nonlinear plasma systems is clearly a key component of the next frontier of computational fusion research and will advance the understanding of magnetically-confined plasmas to an exciting new level. Accelerated progress in the development of the needed codes with higher physics fidelity will be greatly aided by the interdisciplinary (computer science, applied mathematics, and specific applications) alliances championed by the SciDAC Program together with necessary access to the tremendous increase in compute cycles enabled by the impressive advances in supercomputer technology.

The complexity of the physical phenomena in fusion plasmas is due in part to the huge disparity of scales of interest. Scale lengths run from as small as the electron gyroradius (10^{-5} m) to as large as the device size or electron mean free path (10^2 m) while time scales go from the plasma wave period (10^{-11} sec) to confinement times (100 sec). Nevertheless, there has been excellent progress during the past decade in fundamental understanding of key individual phenomena in high temperature plasmas. Modern magnetic fusion experiments are typically not quiescent, but exhibit macroscopic motions that can affect their performance, and in some cases can lead to catastrophic termination of the discharge. Major advances have been achieved in the modeling of such dynamics, which require an integration of fluid and kinetic physics in complex magnetic geometry. Significant progress has also been made in addressing the dynamics governing the interactions between plasmas and electromagnetic waves, especially in the radio-frequency (RF) range of interest for plasma heating. Another key topic, where there have been exciting advances in understanding, is the degradation of confinement of energy and particles

in fusion plasmas caused by turbulence associated with small spatial- scale plasma instabilities driven by gradients in the plasma pressure. While progress has been impressive, the detailed physics of the growth and saturation of these instabilities, their impact on plasma confinement, and the knowledge of how such turbulence might be controlled remain major scientific challenges. This is the challenge currently being undertaken by the SCIDAC Gyrokinetic Particle Simulation Center (GPSC) team, which employs and continues to improve the state-of-the-art gyrokinetic particle-in-cell simulation code, GTC, as its lead code for studying plasma microturbulence.

This article is organized as follows. After providing an overview of the plasma physics to be modeled in the first section, the Gyrokinetic Toroidal Code GTC is described in the next. The most important features are explained, along with some discussion on its optimization. In the third section, results are presented from high-particle-resolution benchmarking tests performed on several of the most important supercomputers in the world, including the 40-Tflop Earth Simulator computer in Japan. In the final section, we discuss the implications of those high-resolution simulations on scientific discovery enabled by the ability to study problems that demand an extreme amount of computer time.

2. Magnetic confinement

The plasma temperature required to sustain the desirable level of fusion reactions in an Earth-bound device is extreme: about 100 million degrees Kelvin at a density of about 10^{20} ions per m^3 . Any interaction between this very hot plasma and the walls of a container invariably leads to a rapid quenching (sudden cooling) of the plasma and damage to the walls. To counter this, a strong magnetic field is applied, which essentially confines the particles by acting upon their trajectories in a way that prevents them from reaching the walls while maintaining the tremendously high pressure required for fusion reactions to occur. A charged particle moving within a magnetic field executes a helical orbit around an axis parallel to the local direction of the field. The particle “gyrates” around the field line while moving freely along it. A closed system in which the field lines are rendered endless by forcing them to loop around can ensure confinement for such particle motions. Null points, where the field vanishes, allows charged particles to readily escape and must be avoided. A closed geometry having all of these properties is the torus (see figure 2). Such closed confinement toroidal systems are readily achievable by appropriate placement of external magnetic coils.

Up to this point, the discussion has focused on how a single charged particle, ion or electron, moves in a uniform magnetic field. A toroidal field is of course non-uniform with the field strength varying inversely with the distance from the center of the torus. Associated gradients produce drifts that push the particles out of their confined trajectories. Minimization of the unfavorable drifts can be achieved by adding a twist to the magnetic field lines as particles move the “long way” around the torus in the toroidal direction. Figure 2 shows a typical field line topology (thin lines) for a toroidal fusion device. Charged particles follow the field lines as they twist around, moving from regions of bad curvature (drift towards the outside of the device) to regions of good curvature (drift towards the inside). In the tokamak device, the twist in the magnetic field lines is produced by a plasma current that flows toroidally, creating its own “poloidal” field that adds to the main field. It is important to mention that the amount of twist in the field lines varies from the center of the cross-section to the outside (minor radius). This variation in the twist is described by the “safety factor” (q) profile.

2.1. The importance of turbulent transport

Although the motion of a single charged particle in an external magnetic field is well understood, putting a large number of such particles together produces complex collective interactions. This behavior results from the self-consistent interactions between the electrically charged particles and the electromagnetic fields that they generate. The simplest approach to studying global

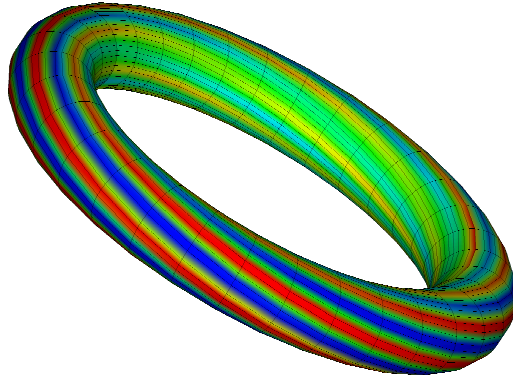


Figure 2. Toroidal flux surface showing the quasi-2D structure of the electrostatic potential generated by the plasma particles. The elongated structures follow the magnetic field lines (thin lines), indicating the prevalence of long wavelength modes in the direction parallel to the field. The magnetic field lines twist poloidally as one goes around the torus toroidally.

fluid-like waves and instabilities is to model the entire plasma as a magnetized fluid by following the familiar magneto- hydrodynamic (MHD) methodology. While reasonably successful in explaining and predicting the macroscopic state and stability of fusion plasmas under a variety of actual operating conditions, it is clear that some key phenomena demand a full kinetic description of the plasma. In particular, the complex motion of ions and electrons subject to the externally imposed magnetic field as well as to their own self-consistently generated electromagnetic fields can generate microturbulence in the presence of spatial gradients in the plasma. The associated turbulence is believed to be the main candidate accounting for the enhanced transport of energy and particles observed in tokamak experiments and has been the subject of intense studies over the past decade [3, 4]. Understanding how energy and particles transit from the hot inner core to the outside of the plasma is of highest importance since the associated confinement critically impacts the cost of sustaining fusion reactions.

The study of complex non-linear kinetic phenomena is best approached with numerical simulation. The Gyrokinetic Toroidal Code, GTC, has been developed as a first- principles physics capability and has been successfully applied to produce important breakthroughs in the understanding of key transport phenomena. For example, simulations of the onset of turbulence and the development of so-called “zonal flows,” which break up the eddies generated by microinstabilities, could explain the observation of enhanced confinement regimes in tokamaks [3]. This study by Lin et al. demonstrated how the particle-in-cell method could be used efficiently on large parallel supercomputers to gain new physical insight that would not have been possible without access to such powerful computational resources.

3. The Gyrokinetic Toroidal Code GTC

The GTC is a 3-D parallel particle-in-cell code that was originally developed by Z. Lin at the Princeton Plasma Physics Laboratory (PPPL) to simulate turbulent transport in fusion plasmas. PPPL has a long history of expertise in particle-in-cell methods, pioneered by John Dawson in the 1960s and improved over the years by researchers such as H. Okuda and W. W. Lee [5, 6]. Major advances in fusion PIC simulations were made during the past two decades with the development of the gyrokinetic simulation approach [6] and its implementation in numerical model. Improved features include the four-point average method to deal with finite ion gyro-radius effects [7], the delta-f approach to reduce computational requirements for noise suppression [8], and the split-weight scheme to treat kinetic effects of fast moving electrons

[9]. These advances have enabled further improvements in the capability of PIC simulations to include more complete physics with the modern version of GTC incorporating all of these advances together with full 3D toroidal geometry. Nevertheless, it should be noted that GTC is a relatively small code of only about 6000 lines. The following sections describe the main features of this featured code.

3.1. Particle-in-cell method

For the Particle-in-Cell (PIC) method, a finite number of charged particles are used to describe the plasma as they move according to the equations of motion along their characteristics. Apart from the forces due to the externally applied magnetic field, all of the other forces are self-consistently generated by the moving particles which create their own electrostatic and electromagnetic fields. While simple, calculating the electrostatic field by summing over the interaction of each pair of particles quickly becomes prohibitive because it is an N^2 calculation, where N is the number of particles in the simulation. Guided by the physics that the Coulomb interaction between the charged particles (ions and/or electrons) is long range, the proper approach is to use a grid and put down the charge density at each point due to the particles in the vicinity. This is called the “scatter” phase of the PIC simulation. The Poisson equation is then used to solve for the electrostatic potential, and the forces are subsequently gathered back to the particles’ positions during the “gather” phase of the simulation. This information is then used for advancing the particles by solving the equations of motion, or “push” phase of the simulation. With a number of particles of the order of $10^8 - 10^9$, this scheme is considerably more efficient since it scales as N rather than N^2 . Furthermore, the act of depositing the charges on a grid helps smooth out the small scale fluctuations in the field due to close encounters between the particles. The physics of interest is at much longer wavelengths and the small scale fluctuations ultimately get damped away by wave-particle interactions such as Landau damping in the system.

The scatter-solve-gather-push steps are repeated until the end of the PIC simulation. In GTC, particles that leave the simulation boundaries are assumed to follow zero order trajectories in the external magnetic field, bringing them back inside the plasma at an easy-to-calculate location. This boundary condition is very easy to apply and has no adverse effect on the calculation. As for the boundary conditions for the field solve, the electrostatic potential is set to zero since we make sure that all the turbulence-driving gradients go smoothly to zero as well at the boundaries.

3.2. Gyrokinetic theory and delta-f scheme

Since the most important kinetic effects that impact the stability and energy transport in tokamaks take place on a time scale much longer than the gyration period of an ion in the magnetic field, the gyration time scale can be eliminated from the system by “gyrophase” averaging the cyclotron motion in the Vlasov equation (collisionless kinetic equation). The physics of the full orbit of the ion is still retained but essentially averaged over one or more cyclotron periods. As shown in Figure (see Figure 3), the helical motion of the ion is basically replaced by a moving charged ring, which is followed via its guiding center. The equation resulting from the gyrophase average of the Vlasov equation is called the gyrokinetic equation and is valid as long as the scale length of the gradients in the system is much larger than the gyroradius of the ions (the radius of the ring described by the ion motion). It should be noted that the gyroradius varies with the velocity of the ion and the local magnetic field strength. GTC solves the gyrokinetic equation with charged rings taking the place of individual particles. To deposit the charge on the grid during the scatter phase, the four-point average technique [7] is used to account for finite Larmor radius effects. This involves choosing four points on the ring with each assigned a quarter of the total charge to deposit on their neighboring grid points. Each particle is followed from the motion of its guiding center but its actual trajectory and

range of influence involves its gyroradius. This way, one retains the non-local physics of the gyrating motion of the ions, while avoiding the difficulties associated with resolving this very fast motion, which would require an impractically small time step. It is not an overstatement to say that the gyrokinetic model made possible the otherwise computationally prohibitive PIC simulations of microturbulence in fusion plasmas.

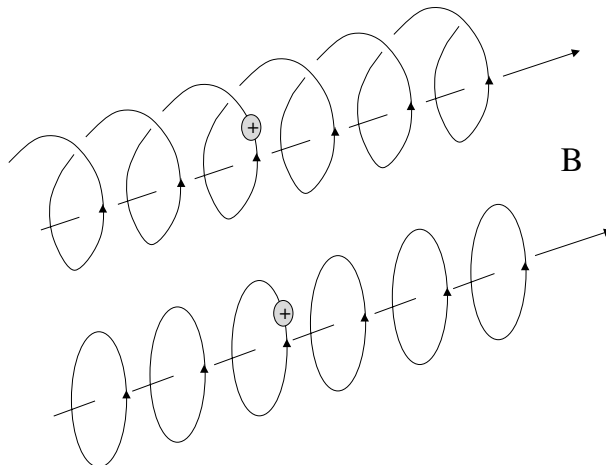


Figure 3. In the gyrokinetic equation, the gyrating motion of the charged particle (ion) is averaged over one or more cyclotron periods. The helical motion is replaced by a moving ring on which the charge is distributed.

The plasma turbulence studied with GTC is driven by spatial gradients in the equilibrium profiles of the plasma temperature and density. A PIC simulation using the full particle distribution function to describe the system would require a non-uniform loading of the particles within the simulation volume in order to produce the correct profiles. Regions of lower densities would have less particles and hence, less statistical resolution. The straightforward way to deal with this problem is to increase the overall number of particles although the computational requirement increases accordingly. To avoid this brute force approach, Parker and Lee [8] developed the δf scheme for gyrokinetic simulations. This method consists in choosing an appropriate initial equilibrium state for the system and keeping it fixed during the simulation while evolving only the perturbed part of the distribution function. We define $\delta f = F - F_0$, where F is the time-evolving distribution function of the system and F_0 , the equilibrium distribution function, which remains fixed and contains the initial density and temperature profiles. The gyrokinetic equation can then be rewritten so that $\delta f/F$ is now evolving in time instead of the full F . This approach is justified as long as the global temperature and density profiles change very little on the time scale of the turbulence being studied. By using this method, we can uniformly load the particles over the whole grid even when a density gradient is present. More importantly, the discrete particle noise associated with the equilibrium distribution function is removed in this method. This greatly reduces the statistical fluctuations in the system, allowing for cleaner simulations with fewer particles. The quantity $\delta f/F$ is called the “weight” (w), and the time step in the simulation is limited either by the electron parallel free motion along the magnetic field line or by the perpendicular drift motion of ion guiding centers. The latter is induced by the fluctuating electric field when electron dynamics is not taken into account.

3.3. Straight field line flux coordinates

The topology of the magnetic field in a tokamak is quite complex, even in its simplest form (limit of large aspect ratio field). In spite of its complex structure, a natural system of coordinates defined by the magnetic field can be chosen such that the field lines are straight instead of twisted, as they appear in normal Cartesian or cylindrical coordinates [10]. Forming such a coordinate system involves using the magnetic fluxes, toroidal (ψ) or poloidal (ψ_p), as the “radial” coordinate, and choosing the appropriate angle coordinates, θ and ζ , that make the field lines straight. This can be applied to any general tokamak equilibria, not just the simple large aspect ratio field model. It is then appropriate to specify the equations of motion for this particular coordinate system [11]. Such an approach has a significant impact on the lowest order of the numerical method that can be used to evolve the particles in the PIC code. Specifically, a simple second-order Runge-Kutta scheme can safely and accurately handle the time-advancing equations since the particles move mainly in a straight line in this coordinate system. Moreover, it can be shown that the trajectories of the magnetic field lines have a Hamiltonian character, with $\psi_p(\psi, \theta, \zeta)$ being the Hamiltonian and ψ, θ, ζ , the canonical coordinates. This allows the use of a very efficient guiding center Hamiltonian method to describe the equations of motion [12].

3.4. GTC grid

In constructing the grid that holds the field information, GTC also takes advantage of the magnetic field topology and electrostatic potential structure. Because of the fast motion of the charged particles along the field lines compared to the slow drift across them, the wave patterns are very different in these two directions. They split naturally between the “parallel” and “perpendicular” directions to the magnetic field (\mathbf{B}). The modes in the parallel direction, which twist around the field lines, have long wavelengths ($k_{\parallel} \ll 1$) as can be seen in Figure 2 where the structure of the electrostatic potential is shown on one of the “flux” surfaces described by the field. Moving in the direction perpendicular to the field lines, one encounters finer scale variations, of the order of the gyroradius, indicating a prevalence of shorter wavelength modes compared to the parallel direction. The code uses this to its advantage by having a grid that follows the same pattern, with the grid following the magnetic field lines as they twist around. The potential is then represented in field-line-following coordinates, which are a special case of the straight field line coordinates. In this system, the electrostatic potential is quasi-2D, having a slow variation in one direction, the one parallel to the field lines, and a faster variation in the other. This quasi-2D structure considerably reduces the number of grid points needed in the toroidal direction. The reduction is of the order of 100, which is also a measure of the savings in computational time. Figure 4 shows a schematic of the GTC grid for a constant value of the radial coordinate (flux surface). Using around 64 grid points or planes in the toroidal direction is found to be adequate in dealing with all sizes of devices. It should be pointed out that the twist in the field lines varies with the radial coordinate, as does the grid.

If one follows a magnetic field line around the torus, this line does not come back to its original position after one cycle unless it lies precisely on a rational surface. As a result, a grid point at $\zeta = 0$ does not necessarily match up with another grid point at $\zeta = 2\pi$. Two methods for enforcing the toroidal periodicity have been implemented in GTC. One method is to map the grids at $\zeta = 0$ to grids at $\zeta = 2\pi$ using interpolations, which results in some spatial damping. A second method is to allow the grid to slightly depart from the magnetic field lines in order to match the grid points, which requires a chain rule in calculating the parallel derivatives. In either approach, there is no approximation in the representation of the toroidal geometry, in contrast to the situation for flux-tube codes.

On the “poloidal” planes perpendicular to the toroidal direction, it is necessary to resolve the electrostatic potential fluctuations of the order of the ion gyroradius. The grid is accordingly

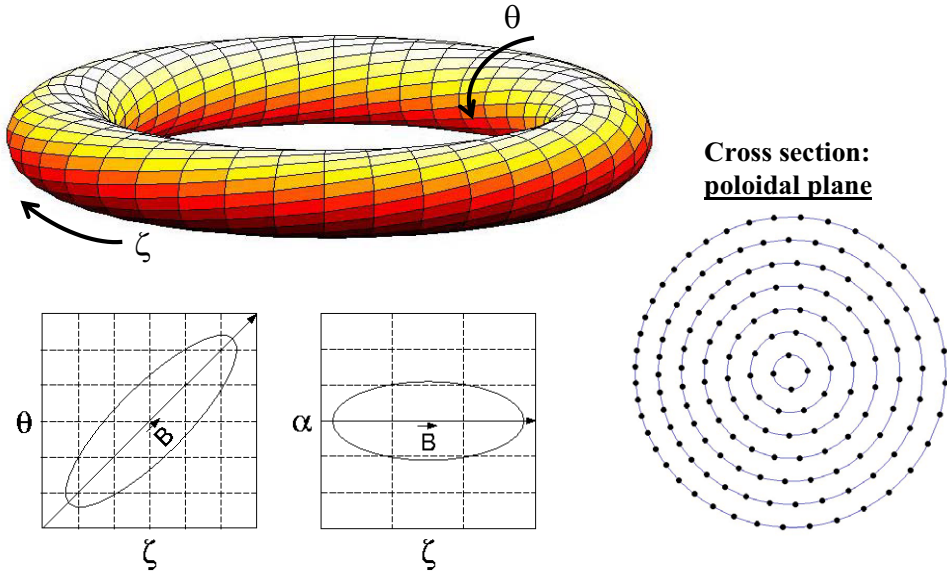


Figure 4. Field-line-following grid used in GTC. The lines drawn between the grid points show how the grid follows the field lines as they twist around the torus. A cross-section of the grid, or poloidal plane, shows how the distance between the points is kept constant on the whole plane in order to have a uniform resolution. It should be noted that the real GTC grid has a much larger number of poloidal grid points than what is depicted here. A system considered “small” will have about 30,000 poloidal grid points.

much finer, and the planes are not exactly perpendicular to the magnetic field lines because of the twist in the field. The quasi-2D potential can be easily mapped from the perpendicular planes onto the poloidal planes in field-line-following coordinates. In order to keep a uniform density of grid points to describe the simulation volume, a non-uniform mesh is used on the poloidal planes. Regularly spaced radial surfaces are used, but more points on these surfaces are used to describe the poloidal direction as one goes from the inner boundary to the outer (see figure 4). The distance between the points can then be easily kept close to the average thermal gyroradius distance required to properly resolve the drift waves in the system.

3.5. Parallel model

There are 3 different parallel schemes currently implemented in GTC. The original scheme consists of a one-dimensional domain decomposition in the toroidal direction using standard message passing calls for communication. A large part of this communication takes place between nearest neighbors, when particles move out of one domain to enter another. GTC uses an efficient two-step algorithm where no single processor is sending to or receiving from more than one processor at a time. During the first step, each processor sends to the “right” the particles that crossed the right boundary while receiving from the “left” the particles that entered from the left boundary. The second step proceeds in the reverse order, sending to the left the particles that crossed the left boundary while receiving the particles that entered from the right. These two steps may need to be repeated since particles with high parallel velocity can move across more than one domain during a time step. This scheme scales remarkably well but is limited in concurrency by the fact that one only needs about 64 toroidal planes to resolve all the physics in the direction parallel to the field lines, as explained in the previous section. Using more planes leaves the results unchanged due to the damping of waves of shorter wavelengths.

To increase the concurrency of GTC, a second level of parallelism was implemented. Since GTC's production platform was, and still is, the large IBM SP Power 3 "Seaborg" at the National Energy Research Scientific Computing Center (NERSC), a shared-memory loop-level hybrid model was implemented to take advantage of the shared memory nodes on the IBM-SP. Each node consists of 16 processors sharing the same large memory in a symmetric way (SMP). OpenMP directives were employed to carry out this fine-grained parallelization, leaving most of the source code unchanged except for the added comment lines that are the directives. The charge deposition algorithm (scatter) did require some special treatment, however. Temporary work copies of the local grid needed to be used for each OpenMP thread to avoid race conditions and the overwriting of array elements by the different threads. Recently, the new OpenMP version 2.0 standard introduced reduction operations on arrays, which removes the need for the explicit creation of temporary work arrays in the scatter routines. However, not all Fortran compilers have implemented the new standard. As the MPI domain decomposition, this algorithm also scales very well, as long as there is enough work for the threads. In GTC, this translates to a large number of particles and a large grid (large fusion device). GTC was one of the first codes to run in hybrid mode MPI-OpenMP on Seaborg at NERSC, and this allowed an increase in concurrency from 64-processor simulations to the full use of 1,024 processors. This tremendous increase in computational capability made possible, for the first time, the simulation of an ITER-size device, which led to an important study of size scaling of turbulent transport in magnetically confined plasmas [13]. The unprecedented ITER-size device simulation used, at the time, 125 million grid points and 1 billion particles.

When GTC was recently ported to the modern parallel vector machines, such as the CRAY X1 and the Earth Simulator, it became clear that more MPI-based parallelization would be required to achieve the desired concurrency. The loop-level shared memory algorithm in GTC competes directly with the vectorization on both machines and also with the streaming on the X1. Vectorization is crucial to achieve high performance on those computers so it is more efficient to run without the shared memory model in this case. In order to regain the lost in concurrency, a new MPI-based particle decomposition scheme was implemented in the code. All the particles located in a toroidal domain can now be split between several processors, each one having a copy of the local grid (poloidal plane). Some of the operations are executed redundantly by those processors but that overhead can be negligible when using a large number of particles per grid point.

These 3 parallel algorithms give GTC great flexibility in the ways it can effectively utilize a variety of available computational hardware.

4. Porting GTC to modern computing platforms

The GTC code is very easy to port to different computers because of its relatively small size, about 6,000 lines, and its use of standard Fortran 90/95 and MPI. It does not require any external libraries although advantage is taken of the optimized FFT library routines available on the system. FFT calls are used for diagnostics and we also have a portable FFT routine for benchmark purposes.

GTC is very well-known in the fusion community due to the scientific breakthroughs that it has generated since its inception, leading to an increased understanding of the transport in tokamaks [3, 13, 14]. Moreover, this code has also achieved some prominence in the HPC community due to its flexible algorithm and its capability to efficiently utilize over a thousand processors in mixed-mode OpenMP-MPI or in MPI-only mode. It is the Fusion Energy Science (FES) representative in the NERSC benchmark suite of applications to evaluate new HPC platforms as part of procurement activities. It was also selected to participate in an application-driven performance study of the most advanced HPC platforms. This work, led by Dr. Leonid Oliker of the Future Technologies group at LBL/NERSC, involved some of the most compute-

intensive scientific codes that can scale to thousands of processors. The study included both superscalar and vector machines, such as the IBM SP Power3, the SGI Altix, the Earth Simulator in Yokohama, Japan, and the CRAY X1 at ORNL [16, 17].

4.1. Optimization on superscalar architecture

While being the basis for the success and performance of the PIC method, the grid-based charge deposition operation is also the source of the PIC codes limited processor efficiency observed on all superscalar and vector computers. The gather and scatter steps between the particles and the grid lead to gather/scatter in memory, requiring a large number of indirect addressing that decreases performance.

GTC was initially developed on the CRAY T3E, although the current version was modified and optimized on the IBM SP Power 3 computer. The two most time consuming routines are the charge deposition and the push routines, accounting for a little more than 80% of the run time, with 40% each. For the charge deposition, a loop over the particles (charged rings) retrieves the position of the guiding center of each one of those particles and calculates the positions and distances to the closest grid points of the four chosen points on the ring. Although the particle array gets scanned sequentially, the particles are randomly distributed within the simulation volume so that the closest grid points jump from one location to another within the charge density array. This leads to a low cache reuse on cache-based architecture since two successive stores to the grid array are usually in different cache lines. However, making the charge density array a two-dimensional array, with the first dimension being the four points of the four-point average method, helps improve the cache reuse since some of the grid points will be common between the four points on the ring.

The particle pushing routine also consists of a big loop over the number of particles in the local domain. For each particle, the value of the field at its location is gathered from the neighboring grid points and the Hamiltonian equations of motion are used to advance all the particle quantities: positions, velocities, and weights. For efficiency and cache optimization, a 2D array, `zion(6,mi)`, is used to hold the particle quantities, with the first dimension being the actual values and the second dimension, the particle index. It was found that better performance could be achieved in the push loop by pre-gathering the field values at the particle positions in a separate loop. These field values are stored in a cache-friendly array that speeds up the expensive push phase.

The key to good performance on a superscalar computer is efficient data layout to maximize cache reuse and data access speed. This is also true for cache-less vector machines. In GTC, the particle and grid arrays have been built in such a way as to reach the best compromise. Sorting the data can sometimes improve the data access and speeds up parts of the code. However, the extra work required to sort often cancels the benefits.

4.2. Modern parallel-vector computers

The GTC vectorization work started on the single node SX-6 at the Arctic Region Supercomputing Center as part of the performance study led by LBNL. The SX-6 was the ideal platform to prepare the code for the Earth Simulator since they share the same basic hardware and software, although the ES has high speed memory and custom-made interconnect. Since a large fraction of the vectorization work on GTC had already been carried out on the single-node NEC SX-6 computer [18], recompiling the code on the Earth Simulator (ES) was straightforward because the compilers were essentially the same on both machines.

4.2.1. Vectorizing the scatter routine As mentioned above, the scatter phase in GTC consists of randomly localized particles that deposit their charge on the grid. This leads to a low cache reuse on cache-based architecture since two successive stores to the grid array are usually in

different cache lines. The situation becomes more critical on vector architecture when two or more particles contributing to the same grid point are part of the same vector operation. There is a memory dependency that prevents vectorization, which would otherwise lead to an incorrect value of the total charge on that grid point. Fortunately, several methods have been developed, over the past two decades, to take care of this issue, and of these, the work-vector method [19] was chosen and implemented. In this approach, a temporary copy of the grid array is given an extra dimension corresponding to the vector length, so that each element of the current vector operation writes to a different memory address, entirely avoiding memory dependencies. After the main loop, the results accumulated in the work-vector array are gathered or reduced to the final grid array. The only drawback of this method is the increase in memory used by the code. The final version, including other temporary arrays created by the compiler, has a memory usage two to eight times higher than the superscalar version of the code, depending on the number of particles per grid point. Other methods to solve this memory dependency problem involve sorting the particles in one way or another, increasing the amount of computation instead of memory, and resulting in a longer run time compared to the work-vector method.

4.2.2. Porting GTC to the Earth Simulator The work-vector method was successfully implemented in the code and led to the full vectorization of the scatter loop with near perfect average vector length (maximum 256). Other improvements in the charge deposition routine consisted of eliminating some memory bank conflicts caused by an access concentration to some particular addresses. This arises, in our case, because the same elements of a few small 1D arrays are accessed several times inside the same loop. A processor needs to wait for a memory bank recovery time when trying to access a memory address that has just been read, and this leads to poor memory access performance when it happens too frequently. By using the “duplicate” directive provided by the ES compiler, several copies of the short 1D arrays can be laid out in memory so that 2 or more memory banks have the same element of the array at different locations in main memory. A processor can then get that element from any one of several banks, thus avoiding the recovery time delay. This approach can be hand-coded by adding an extra dimension to the arrays, but the “duplicate” directive is easier to use and more efficient. This method significantly reduced the bank conflicts in the charge deposition routine and increased its performance by 36.5%. Optimization changes to other subroutines were not nearly as extensive as what was done in the charge deposition. For instance, adding a single compiler directive along with inverting the dimensions of a few arrays was sufficient to achieve 99.4% of vector operation ratio with an average vector length of 255.9 in the push routine. During the limited time spent at the Earth Simulator Center, the last version of the code achieved an overall vector operation ratio of 99% and an average vector length of 242 on the most efficient tests.

4.2.3. Porting GTC to the CRAY X1 As a start, the Earth Simulator version of GTC was put on the CRAY X1 and compiled without changes. The compilation in MSP mode was straightforward and the loop-mark listing generated by the compiler was very informative. The use of the work-vector method in the charge deposition routine was just as necessary on the X1 as on the ES to achieve full vectorization of the scatter loop. In MSP mode, this loop also streams nicely, although a few directives had to be inserted to allow it. The size of the work-vector array for the scatter loop is the same on both machines despite the fact that the vector length on the X1 is 64 compared to 256 for the ES. The scatter loop is being split in 4 streams and each individual stream is being vectorized, so we need 4×64 copies of the grid array to avoid all possible memory dependencies in the loop.

By adding a compiler directive to prevent the vectorization of a short inner loop, the push routine achieved the highest performance of all the subroutines in the code, which was also the case on the Earth Simulator. However, the most time consuming routine on the X1 became

the “shift” subroutine, which was unexpected since this routine had a relatively low percentage (11%) of run time on the ES for the same test case. It was now using more than 54% of the time on the X1. The subroutine “shift” verifies the new coordinates of each particle after the “push” step to check if they moved outside of their current domain and should thus be moved to another processor. The main loop over the particles in this routine contains nested “if” statements that prevent vectorization on both the ES and the X1. The difference in percentage of time spent in “shift” between the two computers is, in part, an indication of the relative speed and efficiency of their respective scalar processor. Since the loop is also not streamed on the X1, it uses only 1 of the 4 scalar processors that are part of an MSP. In this situation, the ES has a clear advantage with its 500MHz 4-way super-scalar processor compared to the 400MHz 2-way super-scalar processor on the X1. By converting the nested “if” statements into two successive condition blocks easily recognizable by the compiler, the main loop could now be streamed and vectorized, decreasing the percentage of time spent in the routine from 54% to only 4%. This change was also implemented on the Earth Simulator during a second visit to Yokohama in 2004, but the impact was not as dramatic as for the X1 due to the faster scalar processor on the ES. Nonetheless, the percentage of time spent in shift dropped from 11% to about 7%.

Codes can be compiled and run in SSP mode on the X1. This eliminates multi-streaming in the code and the execution. Each MSP is then used as four single-streaming vector processors. The drawback is that one now has four processors doing MPI communications and sharing the bandwidth instead of one. Tests were done in SSP mode with GTC but the performance was lower than in the MSP mode. At this point, the best test case on the X1 has a 99.7% vector operation ratio with an average vector length (AVL) of 62.4 (perfect AVL is 64).

5. Performance results

The characteristic of GTC, which distinguishes it from all the other existing microturbulence codes, is the way it performs the entire calculation in “real space.” All of the other codes known to the authors carry out a large fraction of the calculations in Fourier space, which requires many global communications within their parallel algorithm. In GTC, everything is done locally in configuration space, including solving the Poisson equation. This results in high parallel scalability, with communications taking only 10 to 15% of the wall clock time during a typical simulation.

Due to the random gather/scatter that is fundamental to its PIC algorithm, GTC is sensitive to memory latency. This has an impact on both superscalar and vector machines. Low memory bandwidth also affects the performance of the code, especially when the problem size per processor becomes very large. Fortunately, production platforms such as the 6,080-processor IBM SP at NERSC have enough processors to handle the current simulations being performed.

GTC routinely runs on 1,024 processors on Seaborg, at NERSC, with problem sizes ranging from 10 million grid points with 100 million particles on the low end to 100 million grid points with 1 billion particles on the high end. The code achieves between 100 and 150 Mflops/sec per processor on Seaborg, or about 10% of theoretical peak performance. This is typical of most scientific codes, and the trend has been for that ratio to decrease due to the ever increasing processor speed while access to memory remains practically unchanged. GTC gets 5 to 6% of peak on the IBM Power4 and Itanium II processors, although the number of flops per second increases by about a factor of three. With their huge memory bandwidth, low latency, and efficient vector processing, the Earth Simulator and CRAY X1 give GTC an impressive boost in performance. The code runs ten times faster on those two vector computers than on Seaborg. During the first visit to the Earth Simulator Center, GTC included only the 1D domain decomposition and the shared memory model. Since it was decided not to use the OpenMP parallelization in order to maximize the vector efficiency, the code concurrency was limited to 64 processors. Even with this “relatively” small number of processors, GTC ran 20%

faster on both the ES and the X1 than on 1,024 Seaborg processors using the mixed-model. For the second visit a year later, the MPI-based particle decomposition had been implemented in the code and was put to the real test. The model surpassed all expectations for simulations with a large number of particles per cell. With this new algorithm in place, GTC fulfilled the very strict scaling and performance requirements needed to gain access to a large number of processors on the Earth Simulator. The code achieved an unprecedented 3.7 Teraflops/sec sustained on 2,048 processors, with an efficiency of 24% of peak.

The impressive level of performance achieved opens exciting new possibilities in terms of scientific simulations which have never been previously attempted. Phase space resolution in PIC codes is determined by the total number of particles used for the calculation while grid spacing is governed only by the shortest wavelength that the underlying model is able to treat. For the gyrokinetic model, the ion gyroradius is the shortest length in the system and thus corresponds to the grid spacing. To increase the velocity space resolution in a grid-based kinetic code, one must increase the number of velocity grid points for a fixed velocity range and reduce the time step to satisfy the Courant condition. However, these restrictions are not applicable to GTC. Specifically, increasing the number of particles has no effect on the time step but increases the resolution in both the velocity and configuration spaces. High resolution simulations can reveal sharp resonances in the interaction between waves and particles. These powerful new capabilities can also enable systematic investigations of the often neglected velocity-space non-linearity in gyrokinetic simulations and also in keeping the fluctuation level low during long-time simulations. To assess the feasibility of such calculations and see how far we could push the number of particles while keeping the same run time, a series of high resolution tests runs were performed during the performance study of the top HPC platforms. Figure 5 shows a scaling plot of the “compute power”. Here we define the compute power as the number of particles being moved one time step in one second. The faster the code runs on a given computer, the more particles can be moved per second. The plot clearly shows the superior performance of the two vector machines, the Earth Simulator and the CRAY X1. Using 2,048 processors, the Earth Simulator can move 2.6 billion particles \times step per second while Seaborg can move only 180 million. Thunder, the Itanium II cluster with a Quadrics interconnect, can push 3 times more particles than Seaborg but almost 5 times less than the Earth Simulator. The X1 is almost at the level of the ES but has only 512 processors, or MSPs, compared to the ES, which has ten times more. Fortunately, the X1 is currently being upgraded to X1E, where the final configuration will have twice the number of MSPs running 40% faster than the old ones. From a “time to solution” perspective for very challenging scientific problems, these powerful new resources should provide enough compute power to enable performing long-time, high resolution simulations within a very reasonable time.

6. Conclusions

The Gyrokinetic Toroidal Code has been very successful in contributing to increased understanding of turbulent transport in fusion plasmas. Its self-consistent treatment of particles and fields at the kinetic level in a global toroidal geometry is key to further scientific progress in dealing with the complex dynamics and non-linearities in fusion energy systems. With its demonstrated ability to effectively utilize the most advanced high-performance computing resources available, GTC has made possible the simulation of some of the most challenging scientific problems. For example, the implementation of the mixed-model OpenMP-MPI on the IBM SP 3 has enabled the “scientific discovery” type of simulation of new confinement trends in a global ITER-size system. This large-problem-sized simulation was carried out in a reasonable amount of time using over a thousand processors, something never previously achieved in fusion computations. Continued improvements in the parallel algorithms and optimization are expected to help maintain GTC’s proficiency on all the new HPC platforms. Nevertheless, it should be

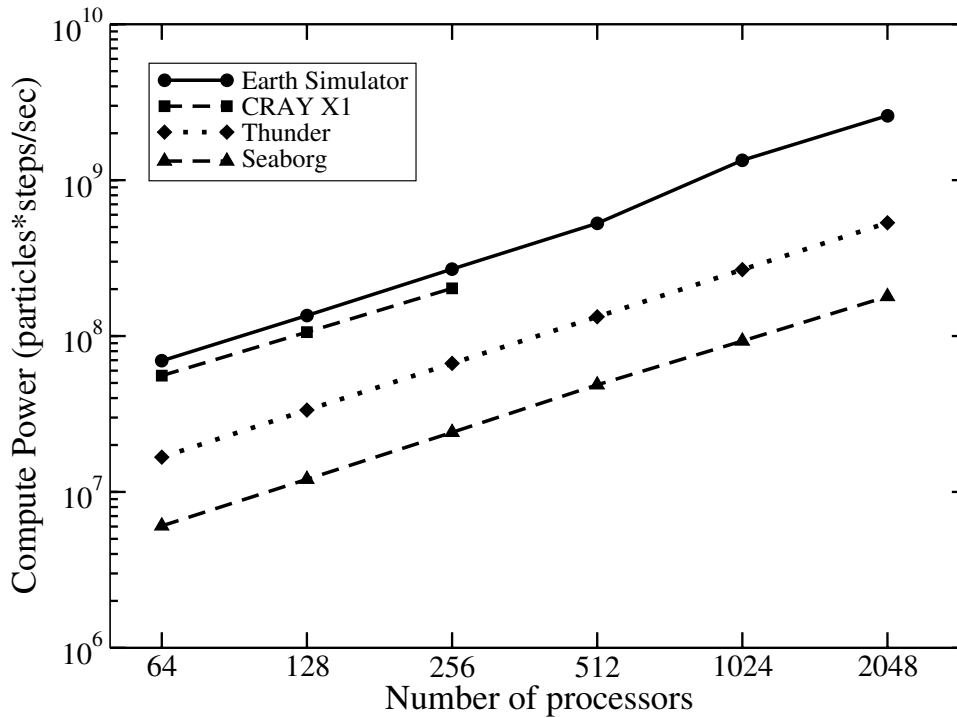


Figure 5. Compute power of GTC on four of largest HPC platforms used in our study. The compute power is defined as the number of particles being moved on time step in one second. The highest performance is achieved by the Earth Simulator vector computer, moving 2.6 billion particles for 1 step in 1 second.

kept in mind that this is not a static code and it is constantly being improved and upgraded with more physics capabilities. New features which have either recently been incorporated or are under active development include: full electron kinetic dynamics, interface with realistic general magnetic equilibria, and electromagnetic effects. With these additional physics capabilities, GTC holds promise for delivering more outstanding fusion plasma scientific discoveries through advanced computing.

Acknowledgments

S. Ethier would like to thank Dr. Leonid Oliker of the LBL Futures Technologies group and Dr. Horst Simon, director of NERSC, for the opportunity to participate in the advanced computing platforms collaboration. Thank you to Dr. Sato, director-general of the Earth Simulator Center, Dr. Tsuda and Mr. Kitawaki for their help and hospitality during the two visits at the ES Center. Thank you to David Park of NEC USA for his great help in optimizing GTC on the SX-6 and Earth Simulator. Many thanks also to Dr. James Schwarzmeier and Nathan Wichmann of CRAY for their help porting GTC to the CRAY X1. This work was supported by the Center for Gyrokinetic Particle Simulation of Turbulent Transport in Burning Plasmas under the SciDAC program of the Department of Energy, and by DOE Contract No. DE-AC02-76-CHO-3073 (PPPL).

References

- [1] http://www.sc.doe.gov/Sub/Facilities_for_future/20-Year-Outlook-screen.pdf
- [2] <http://www.iter.org/>

- [3] Lin Z, Hahm T S, Lee W W, Tang W M and White R B 1998 *Science* **281** 1835.
- [4] Dimits A M et al. 2000 *Phys. Plasmas* **7** 969.
- [5] Dawson J M 1983 *Rev. Modern Phys.* **55** 403.
- [6] Lee W W 1983 *Phys. Fluids* **26** 556.
- [7] Lee W W 1987 *J. Comp. Phys.* **72** 243.
- [8] Parker S E and Lee W W 1993 *Phys. Fluids B* **77** 1237.
- [9] Manuilskiy I and Lee W W 2000 *Phys. Plasma B* **7** 1381.
- [10] Boozer A H 1980 *Phys. Fluids* **23** 904.
- [11] White R B and Chance M S 1984 *Phys. Fluids* **27** 2455.
- [12] White R B 1990 *Phys. Fluids B* **2** 845.
- [13] Lin Z, Ethier S, Hahm T S and Tang W M 2002 *Phys. Rev. Lett.* **88** 195004.
- [14] Lin Z, Hahm T S, Lee W W, Tang W M, and Diamond P H 1999 *Phys. Rev. Lett.* **83** 3645.
- [15] Dimits A M 1993 *Phys. Rev. E* **48** 4070.
- [16] Oliner L et al. 2003 Evaluation of Cache-based Superscalar and Cacheless Vector Architectures for Scientific Computations in *Proceedings of SC2003 conference* Phoenix, AZ.
- [17] Oliner L et al. 2004 Scientific Computations on Modern Parallel Vector Systems in *Proceedings of SC2004 conference* Pittsburgh, PA.
- [18] Ethier S, Lin Z 2004 *Comp. Phys. Comm.* **164** 456.
- [19] Nishigushi A, Orii S, and Yabe T 1985 *J. Comp. Phys* **61** 519