COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **4**, No. 3, pp. 675-702

Commun. Comput. Phys.
September 2008

Plasma Edge Kinetic-MHD Modeling in Tokamaks Using Kepler Workflow for Code Coupling, Data Management and Visualization

J. Cummings[1,*], A. Pankin[2], N. Podhorszki[3], G. Park[4], S. Ku[4], R. Barreto[5], S. Klasky[5], C. S. Chang[4], H. Strauss[4], L. Sugiyama[6], P. Snyder[7], D. Pearlstein[8], B. Ludäscher[3], G. Bateman[2], A. Kritz[2] and the CPES Team[9]

[1] *California Institute of Technology, Pasadena, CA 91125.*
[2] *Lehigh University, Bethlehem, PA 18015.*
[3] *University of California at Davis, Davis, CA 95616.*
[4] *Courant Institute of Mathematical Sciences, New York University, NY 10012.*
[5] *Oak Ridge National Laboratory, Oak Ridge, TN 37830.*
[6] *Massachusetts Institute of Technology, Cambridge, MA 02139.*
[7] *General Atomics, San Diego, CA 92186.*
[8] *Lawrence Livermore National Laboratory, Livermore, CA 94550.*
[9] *SciDAC FSP Center for Plasma Edge Simulation.*

Received 2 November 2007; Accepted (in revised version) 28 January 2008

Available online 21 April 2008

Abstract. A new predictive computer simulation tool targeting the development of the H-mode pedestal at the plasma edge in tokamaks and the triggering and dynamics of edge localized modes (ELMs) is presented in this report. This tool brings together, in a coordinated and effective manner, several first-principles physics simulation codes, stability analysis packages, and data processing and visualization tools. A Kepler workflow is used in order to carry out an edge plasma simulation that loosely couples the kinetic code, XGC0, with an ideal MHD linear stability analysis code, ELITE, and an extended MHD initial value code such as M3D or NIMROD. XGC0 includes the neoclassical ion-electron-neutral dynamics needed to simulate pedestal growth near the separatrix. The Kepler workflow processes the XGC0 simulation results into simple images that can be selected and displayed via the Dashboard, a monitoring tool implemented in AJAX allowing the scientist to track computational resources, examine running and archived jobs, and view key physics data, all within a standard Web

*Corresponding author. *Email addresses:* `cummings@cacr.caltech.edu` (J. Cummings), `pankin@lehigh.edu` (A. Pankin), `pnorbert@cs.ucdavis.edu` (N. Podhorszki), `gypark@courant.nyu.edu` (G. Park), `sku@cims.nyu.edu` (S. Ku), `barreto@ornl.gov` (R. Barreto), `sklasky@ornl.gov` (S. Klasky), `cschang@cims.nyu.edu` (C. S. Chang), `strauss@courant.nyu.edu` (H. Strauss), `sugiyama@psfc.mit.edu` (L. Sugiyama), `snyder@fusion.gat.com` (P. Snyder), `ldp@llnl.gov` (D. Pearlstein), `ludaesch@ucdavis.edu` (B. Ludäscher), `bateman@lehigh.edu` (G. Bateman), `kritz@lehigh.edu` (A. Kritz)

browser. The XGC0 simulation is monitored for the conditions needed to trigger an ELM crash by periodically assessing the edge plasma pressure and current density profiles using the ELITE code. If an ELM crash is triggered, the Kepler workflow launches the M3D code on a moderate-size Opteron cluster to simulate the nonlinear ELM crash and to compute the relaxation of plasma profiles after the crash. This process is monitored through periodic outputs of plasma fluid quantities that are automatically visualized with AVS/Express and may be displayed on the Dashboard. Finally, the Kepler workflow archives all data outputs and processed images using HPSS, as well as provenance information about the software and hardware used to create the simulation. The complete process of preparing, executing and monitoring a coupled-code simulation of the edge pressure pedestal buildup and the ELM cycle using the Kepler scientific workflow system is described in this paper.

PACS: 52.65.-y, 52.65.Kj, 52.65.Tt, 52.65.Ww

Key words: Plasma simulation, magnetohydrodynamic and fluid equation, gyrofluid and gyrokinetic simulations, hybrid methods.

1 Introduction

In a tokamak fusion reactor, if the hot edge plasma, which has a density and temperature around $1\times10^{20}\ m^{-3}$ and 5 keV, respectively, is allowed to touch the material wall in an uncontrolled way, it can sputter the wall material into the plasma, which may degrade or extinguish the fusion burn and may shorten the wall lifetime to an unacceptable level. In an attempt to control this problem, all the modern tokamaks, including the planned ITER (International Thermonuclear Experimental Reactor), have been designed to divert the escaping edge plasma to a specific location called the "divertor chamber" by means of a magnetic field produced by external coils. The wall plates in the divertor chamber are then expected to be periodically replaced in a tokamak reactor.

The magnetic field lines inside the tokamak chamber are divided into two groups: one forming nested closed surfaces in the main chamber without touching the material wall, and the other leading to the divertor chamber. The boundary separating these two groups is the magnetic separatrix surface. The region outside the separatrix surface containing diverted field lines is called the "scrape-off" region, and the region inside the separatrix containing nested magnetic surfaces is called the "core" region. The plasma in the core region is hot and dense, while the plasma in the scrape-off region is cold and diluted (except just in front of the divertor plates).

Tokamak plasma transport is normally anomalous and is thought to be associated with small-scale plasma turbulence. When the heating power to the core plasma is above some threshold, it has been observed experimentally that there forms a thin plasma layer just inside the separatrix surface in which the plasma is almost free of turbulence; the cross-field transport rate is reduced to the neoclassical level [1] (neoclassical transport is a collisionally driven transport in an inhomogeneous magnetic field). This layer is called the "H-mode" layer, where H-mode is an abbreviation for "high-confinement mode." A tokamak plasma without an H-mode layer is called "L-mode" (low-confinement mode), in which the plasma transport is anomalous and dominated by small-scale turbulence. Since the cross-field transport rate in the thin H-mode layer is much lower than the ambient transport rate, the local plasma gradient becomes very steep, fed by the particle source from ionization of incoming neutral particles from the wall and the heat source provided by the outward flow from the core region. As a result, the plasma forms a distinct pedestal extending from the scrape-off region a short distance into the core, with most of the gradient existing in the H-mode layer. This pedestal raises the fusion probability dramatically by raising the central plasma temperature and density.

Under normal conditions, the rise of the pedestal triggers magnetohydrodynamic instabilities known as edge localized modes (ELMs), which both constrain the pedestal pressure (limiting the fusion power in the core) and eject heat and particles onto material surfaces (reducing the wall lifetime of a fusion reactor). The onset of ELMs is known to be dependent upon the physical properties of the edge pedestal. Neither the pedestal growth properties nor the ELM crash physics is sufficiently known at the present time for a predictive capability. Success of next-generation burning plasma experiments such as ITER is heavily dependent upon achieving H-mode operation and obtaining an edge pedestal of sufficient height without triggering large-scale ELMs. The physical understanding and prediction capability of the edge plasma pedestal and the accompanying ELMs are therefore at the highest priority in fusion plasma research.

Understanding the pedestal structure requires a first-principles, full distribution function, kinetic simulation due to the steep pressure gradient, low collisionality, unconventional particle orbits, and non-Maxwellian ions. A fluid code cannot properly describe these edge-specific features. The kinetic code should include neoclassical ion-electron-neutral dynamics, as well as the self-consistent electromagnetic perturbations and turbulence. This is a difficult, long-term effort, requiring intense collaboration between physicists, applied mathematicians, and computer scientists, working on high-performance computing platforms. Such work is the focus of the Center for Plasma Edge Simulation (CPES), a SciDAC (Scientific Discovery through Advanced Computing) project that is jointly funded by the U.S. DOE Office of Fusion Energy Sciences and Office of Advanced Scientific Computing Research.

The kinetic code XGC0, which computes plasma equilibrium evolution in the presence of neoclassical ion-electron-neutral dynamics, is presently in operation in the CPES project. This code is capable of simulating pedestal growth across the separatrix from neutral ionization for the first time. The ion neoclassical transport in XGC0 can be enhanced to include turbulent transport using a phenomenological model. For a complete kinetic simulation of edge turbulence and transport properties, the gyrokinetic edge code XGC1 is under development. The electrostatic turbulence capability of XGC1 is now undergoing rigorous verification; developments will then focus on the drift time scale, electromagnetic turbulence and transport across the separatrix. Once the code is complete, XGC1 will be coupled to XGC0 to replace the phenomenological turbulent transport model. Both XGC0 and XGC1 grew out of the original XGC ion guiding center code [2].

Inclusion of the high-frequency MHD phenomena into the XGC1 gyrokinetic framework is a longer term goal. In the interim while XGC1 establishes this MHD capability, simulations of ELM crashes on the MHD time scale are being carried out using the nonlinear MHD code M3D [3], coupling it to the kinetic code XGC0 to incorporate kinetic information. M3D is an advanced, resistive MHD code funded by a SciDAC project (CEMM, the Center for Extended Magnetohydrodynamic Modeling), with its capability extended to include two-fluid effects (i.e., Extended MHD). M3D can model the diverted magnetic separatrix geometry. The vacuum region is simulated as a cold plasma with high resistivity. M3D is parallelized for use on a moderate number of processors. NIMROD is another nonlinear extended MHD code in the CEMM project with similar features. The NIMROD code may also be coupled to XGC0 for cross verification. The pedestal growth in XGC0 is monitored for ELM instability by a linear ideal MHD code ELITE [4,5]. ELITE has been validated against many experimental data sets for the large-scale Type I ELM instability onset.

A coupled simulation containing a kinetic model of edge pedestal build up and an extended MHD model of ELM behavior is a necessary component in the understanding of the pedestal-ELM cycle physics. Such a comprehensive model of this physics will yield the predicted pedestal height, which is related to the core confinement, and the expected wall load, which is related to material lifetime. In this report, we describe our efforts to perform this type of coupled simulation for the first time, using a novel approach that employs scientific workflows and a dashboard monitoring tool to automate and simplify the process.

2 Code components for coupled kinetic-MHD edge simulation

Simulation of the full ELM cycle requires the cooperation of several types of physics models, operating in a loosely coupled manner. Edge transport physics, including neoclassical effects, ion orbit loss, neutral recycling at the boundaries, heating from the core plasma, turbulent transport, and self-consistent electromagnetic fields, require a fully kinetic plasma model such as a particle-in-cell code. A Grad-Shafranov solver is needed to update periodically the magnetic equilibrium in which the kinetic plasma model operates. Detection of potentially unstable edge localized modes is provided by an ideal MHD linear stability analysis code. If an ELM is driven unstable, a fully nonlinear MHD initial value code can be used to simulate the evolution of the ELM and its modification of the edge plasma profiles. As the ELM crash completes and dissipates, the new plasma equilibrium that is established would be taken as the starting point for the next ELM cycle. A brief description is presented here for each of the simulation models that have been employed in the coupled simulations of the ELM cycle being presented here.

2.1 XGC0

A particle-in-cell, guiding center code working in a five-dimensional phase space (three in physical space and two in velocity space), XGC0 has specifically been developed for the study of plasma edge dynamics [2]. XGC0 can handle neoclassical transport of both ions and electrons self-consistently with radial electric field physics and Monte Carlo neutral atom penetration physics. Charged particle motion in XGC0 is governed by the well-known Hamiltonian guiding center equations of motion [6], and XGC0 calculates radial electric field dynamics self-consistently using the neoclassical current balance equation. In XGC0, the effect of the parallel current on the radial electric field dynamics in the scrape-off region is accounted for by using the logical sheath method [7], which removes the fastest electrons periodically to satisfy plasma quasineutrality.

XGC0 can be operated with or without electrons; in the case without electrons, the electrostatic potential profile in the scrape-off region is externally prescribed. Simulation of both ions and electrons can be used to compute radial E-field in the scrape-off region and self-consistent bootstrap current evolution, which is caused in part by the electron particle banana orbits, and thus requires following electron orbits accurately in the presence of Coulomb collision effects. The typical XGC0 simulation with electrons uses a reduced ion-electron mass ratio (such as 400) along with a subcycling technique.

There are two types of particle collisions in XGC0: Coulomb collisions and plasma-neutral collision processes such as ionization, charge exchange, and elastic collisions. Collisions in XGC0 are implemented using a Monte-Carlo method; thus, particle velocities are randomly changed periodically according to the collision process, new electron-ion pairs are created through neutral ionization, and plasma temperature is reduced through inward diffusion of neutrals and ion-neutral charge exchange. The Monte-Carlo Coulomb collision model in XGC0 employs linear collisions with a background Maxwellian distribution [8] in the frame of the rotating plasma. For ion-ion collisions, a scheme conserving plasma momentum and energy is used [9].

For an initialization of the plasma particles, a shifted Maxwellian distribution is used with given density, temperature, and toroidal rotation profiles. The profiles can be modeled by an analytic equation, such as the modified hyperbolic tangent form, or by experimental data contained in a specific data format such as g-eqdsk. The electrons and ions are followed until the edge plasma reaches either neoclassical steady state or the ELM instability boundary. Because plasma particles and their energy are gradually lost to the wall boundary, particle and heat sources are needed for reaching steady state, and these roles are played by neutral recycling and a plasma heating model, respectively. Although the XGC0 code is used primarily for the study of the edge ion-electron neoclassical transport, a simple phenomenological turbulent diffusion model, based on the random walk superposed on the Lagrangian motion, is also included to simulate the effect of turbulent diffusion on the pedestal dynamics.

Both XGC0 and the gyrokinetic edge turbulence code XGC1 have shown good scalability with number of processors. The greatest number of processors used by XGC1 to date is 16,384 for a production run. For test runs, XGC1 typically uses about 4,000 processors, while an XGC0 run without electrons may use as few as 128 processors. The best computing speed of XGC1 achieved to date on the Cray XT3/XT4 at Oak Ridge National Laboratory (ORNL) is 440 Mflops per processor core, which corresponds to about 9% of the theoretical peak processing speed.

2.2 TEQ

Each XGC0 simulation starts typically with an experimental EFIT equilibrium. As the plasma profiles advance in the XGC0 code, it generally becomes necessary to compute a new equilibrium. In particular, during the transition from low- to high-confinement regimes (L-H transition), the plasma shaping parameters such as triangularity and elongation and the separatrix location typically change. In tokamak experiments, the plasma shaping and separatrix location are controlled by coil feedback loops. In order to reproduce accurately experimental conditions, the XGC0 code needs to resolve the evolving equilibrium in time. New equilibrium profiles are also needed by all MHD codes that are part of the CPES kinetic-MHD coupling framework.

The TEQ equilibrium solver [10] is an NTCC module [11] that was originally developed as a part of the Corsica transport code [12]. The module can be used to compute both prescribed-boundary and free-boundary equilibria. Prescribed-boundary equilibrium solvers are often referred to as *inverse equilibrium solvers*, and free-boundary equilibrium solvers are often referred as to *direct equilibrium solvers*. For the TEQ inverse equilibrium solver, the solution for the magnetic configuration is given in terms of the major radius R and vertical position Z, as a function of the magnetic flux stream function ψ(which is the poloidal magnetic flux divided by 2π) and a poloidal angle-like variable θ. This inverse representation is applicable only to the core plasma in which magnetic surfaces are simply nested closed contours. For the TEQ direct equilibrium solver, solution for the magnetic configuration is represented in the form $\psi = \psi(R,Z)$. Magnetic surfaces are determined by tracing out contours of constant ψ. The TEQ module contains initialization information for virtually any existing tokamak, and it includes "dead-start" procedures to generate initialization information for new tokamaks.

The TEQ equilibrium solver has been recently benchmarked against other equilibrium solvers [13]. It has been demonstrated that the TEQ direct equilibrium solver provides a robust solution with small residuals at the plasma edge, which particularly suits the needs of edge plasma simulations. In addition, the TEQ code is capable of generating the *eqdsk* files that are recognized by the ideal MHD stability code ELITE and the extended MHD codes M3D and NIMROD. The TEQ direct and inverse equilibrium solvers are implemented in the XGC0 code. The direct equilibrium solver yields g-eqdsk files used by the NIMROD code, and the inverse equilibrium solver yields t-eqdsk files used by the ELITE code. In the CPES framework, the TEQ direct equilibrium solution provides boundary conditions (e.g., separatrix location) for the TEQ inverse equilibrium solver. The TEQ module has been tested for several DIII-D discharges. Fig. 1 shows the magnetic flux surfaces, coil locations, and TEQ grid for DIII-D discharge 113317.

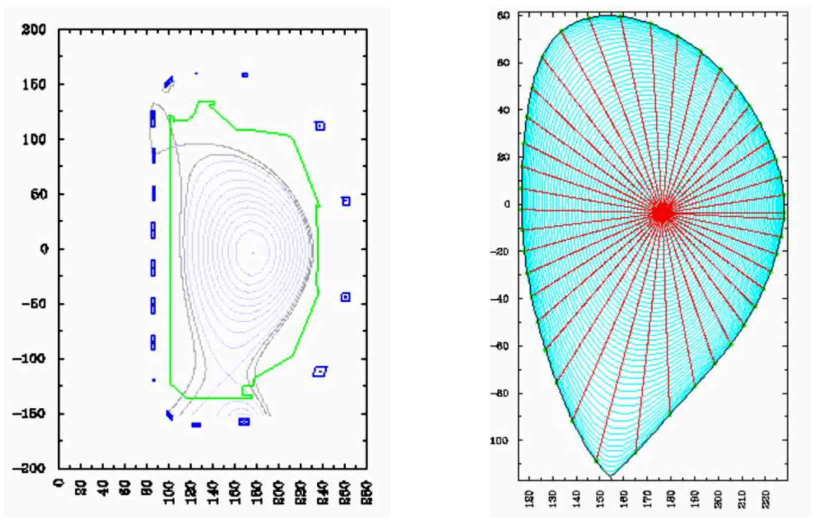


Figure 1: Magnetic flux surfaces and computational grid for a DIII-D equilibrium that is used in CPES studies. The shape of the wall is shown just outside the separatrix in the left panel.

2.3 ELITE

The sharp pressure gradient and resulting large bootstrap current in the edge transport barrier or "pedestal" region of the plasma provide free energy, which can drive magnetohydrodynamic modes unstable. The most unstable MHD modes can be predominantly pressure-driven ballooning modes, current-driven kink/peeling modes, or, most commonly, coupled peeling-ballooning modes driven by both pressure gradient and current. The "peeling-ballooning" theory of ELMs was first developed in the local high-n (toroidal mode number) limit [14], and later extended to incorporate intermediate-n and non-locality [4, 5]. A highly efficient $2D$ numerical code, ELITE [4, 5], has been developed to study these non-local, finite-n instabilities. ELITE has been successfully benchmarked against the MISHKA, GATO, DCON, CASTOR, MARS, MARG2D, and BALMSC codes [5,15,16] and has been used in comparisons with a number of tokamak experiments including DIII-D [5,15,17–23], MAST [24], Alcator C-Mod [25], and JT-60U [26,27]. In general, standard "Type I" ELMs are observed when the peeling-ballooning stability constraint is exceeded, and that the most unstable modes are typically $n \sim 4-30$ peeling-ballooning modes.

ELITE employs a high-n expansion, taken to sufficiently high order so as to allow accurate calculations for $n \geq 4$. It also takes advantage of the radial localization of poloidal modes to achieve very high numerical efficiency. ELITE calculates growth rates and mode structure, allowing it to be used both with a diamagnetic stabilization criterion (here we use $\gamma > \omega *_{pi}/2$) and in comparisons with observed ELM structure [19]. ELITE's efficiency and robustness render it a useful tool for integrated studies of ELM onset such as those conducted here. ELITE consists of a suite of three Fortran 90 executables, normally run via a set of scripts. While ELITE itself is a serial code, the natural parallelism

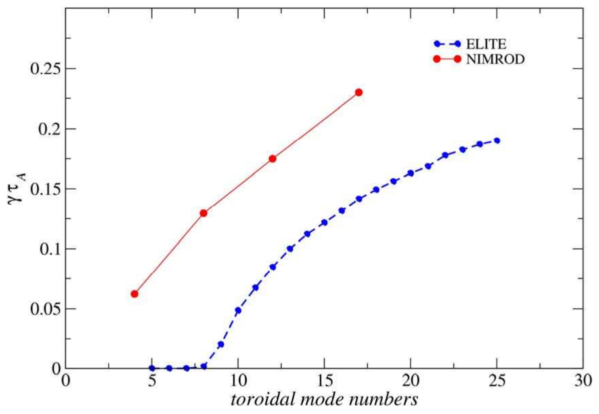


Figure 2: Growth rates as a function of toroidal mode number computed with the NIMROD and ELITE codes for the DIII-D discharge 113317.

across toroidal mode number, which is a good quantum number in a linear calculation, allows ELITE to be invoked as a set of distributed processes, simultaneously evaluating stability for a range of n values on a single equilibrium to find the ELM onset time.

2.4 NIMROD

The extended $3D$ MHD NIMROD code [28–33] advances the well known set of extended MHD equations as a function of space and time. This set includes continuity equations for mass and momentum, the temperature evolution equation, and Maxwell's equations in the presence of finite resistivity, Hall current, electron pressure gradients and finite Larmor radii (FLR) effects. The gyro-viscous stress tensor and Hall current effects tend to stabilize high mode numbers [34, 35]. The stabilization of the high-n modes, which have a relatively fine spatial scale, should make the corresponding nonlinear computation easier. However, the two-fluid effects cause the modes to rotate with a poloidal drift frequency comparable to the diamagnetic frequency, and this mode rotation makes it more difficult to advance the equations in time.

Numerous verification studies have been carried out in which NIMROD results have been compared with the results of other MHD codes as well as analytic scalings [32]. In order to assure that the growth rates and peeling-ballooning stability thresholds found with the ELITE code qualitatively agree with the stability threshold found with the NIMROD code, the two MHD codes have been benchmarked against each other for a sample equilibrium from the CPES kinetic-MHD coupling studies. The equilibrium that is based on the DIII-D discharge 113317 is selected for the comparison. The growth rates as a function of toroidal mode number computed with the NIMROD and ELITE codes are shown in Fig. 2. The growth rates computed by NIMROD are approximately two times higher than the growth rates computed with the ELITE code, and the peeling-ballooning threshold computed with NIMROD is lower. The differences in results can be explained by differences in the resistivity profiles in ELITE and NIMROD. The ELITE code is an ideal MHD code with zero resistivity in the plasma core and infinite resistivity in the vacuum region. For linear NIMROD runs, the resistivity in the "vacuum" region is up to 1×10^6 larger than in the core plasma, approaching the ideal limit. For the free-boundary simulations, the resistivity has the Spitzer temperature dependence $T^{3/2}$. For the DIII-D simulations shown here, the temperature ratio between the core and edge is approximately 500, which yields a resistivity ratio of approximately 10^5. For the lower resistivities in the plasma region and higher resistivities in the vacuum region, NIMROD yields results that are in better agreement with the ELITE results. Despite the differences in the growth rates, the eigenfunctions computed with NIMROD and ELITE agree reasonably well [34, 36]. This makes it possible to use the NIMROD code for nonlinear evolution studies of ELM crashes and to use the ELITE code, which is better validated against the experimental data, for the linear peeling-ballooning stability analysis of plasma profiles computed with the XGC0 kinetic code.

2.5 M3D

The M3D code [3] solves the extended MHD equations [37]. It is based on discretization with linear finite elements [38] in poloidal planes, with either pseudospectral or finite difference representation in the toroidal direction. It has been implemented in two packages with different parallelizations: M3D-OMP uses OpenMP and runs on shared memory computers, while M3D-MPP runs on distributed memory computers.

M3D-MPP employs domain decomposition within and among the poloidal planes and uses MPI and distributed matrix solvers from the PETSc library [39]. In this application of the M3D code to ELM simulations, M3D-OMP is essentially used as a serial preprocessor for M3D-MPP. M3D-OMP is used to read an *eqdsk* file of equilibrium profile data, generate an unstructured FE mesh that is aligned with closed magnetic flux surfaces, and interpolate the primary M3D variables (poloidal magnetic flux ψ, pressure p, toroidal current, mass density ρ, and toroidal magnetic field) to the mesh. M3D-OMP then writes this data to a file that is used to initialize M3D-MPP. M3D-OMP can also read in the plasma profiles generated by the XGC0 kinetic code and produce a new *eqdsk* file.

M3D has been extensively benchmarked. Originally it used a different spatial discretization, with radial finite differences and spectral representations in the poloidal and toroidal directions. That version of M3D was benchmarked against PEST and compared with theory for a number of different problems. The present FEM version uses essentially the same top-level physics code, but provides more geometric flexibility as well as nonlinear robustness. It has been benchmarked against the original spectral implementation and against the MARS and NOVA-K codes for internal MHD modes, with and without fast particles [40]. Recently a linear ELM benchmarking test that includes the vacuum region was carried out, comparing M3D and NIMROD with ELITE and GATO, which in turn have been extensively benchmarked. Growth rates (see Fig. 3) and mode structures agree reasonably well. Some level of disagreement is expected, since ELITE and

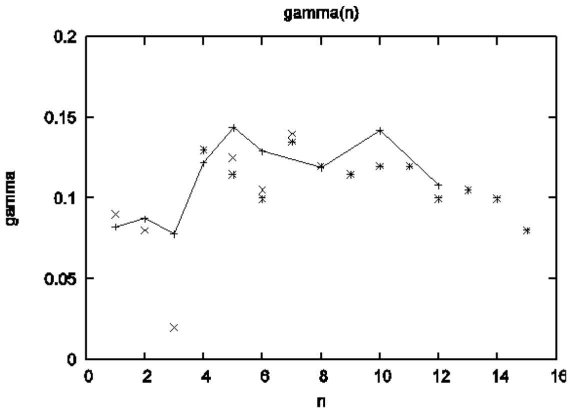


Figure 3: Linear ELM growth rate as a function of mode number n. Solid curve is M3D results, points marked with $\times$ are from GATO, and points marked with * are from ELITE.

GATO are ideal MHD stability codes, while M3D is a resistive MHD nonlinear code. The vacuum region outside the separatrix is modeled in M3D as a highly resistive, low density region. Similarly, the M3D plasma is moderately resistive, much like the real edge plasma. Both of these assumptions about the plasma resistivity will affect the results.

To permit the long-time simulations required for ELM relaxation, M3D was modified slightly with a semi-implicit treatment of advection, in order to ensure that the density and temperature remain positive. The vacuum region model strongly affects the ELM relaxation. Extra dissipation can be added near the wall boundary to provide a sink for density and temperature. The M3D grid for ELM simulations usually includes a central hole in the core plasma because the central region is not affected by the ELM. It has been found that shrinking the central hole is helpful in nonlinear simulations, allowing a more complete relaxation of the plasma temperature profile. Future ELM simulations using M3D may be done without a central hole in the grid, using a relatively coarse FE mesh in this central region instead.

3 Code coupling scenarios for an ELM cycle simulation

Given the simulation codes described above, there are many ways to couple them and provide a complete ELM cycle simulation. Several different combinations of these and other similar physics models have been explored, examining such factors as ease of integration and performance of individual code components on our preferred computing platform. Through this process, the relative ease of integrating and managing such code components has been demonstrated using the Kepler framework for automated workflows. Here two basic code coupling scenarios that were successfully employed for a comprehensive simulation of the ELM crash will be highlighted.

Scenario 1 involves the coupling of four code components, as illustrated in Fig. 4: XGC0, M3D-OMP, ELITE, and M3D-MPP. The XGC0 kinetic code is initialized with ana-

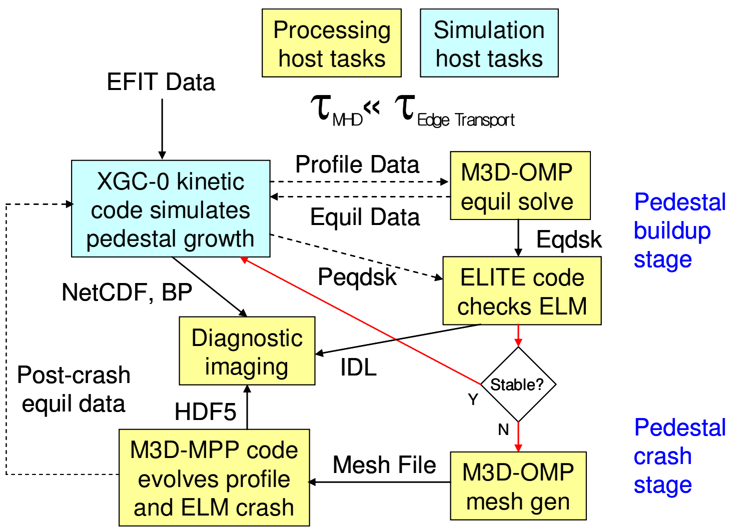


Figure 4: Schematic of a sample kinetic-MHD code coupling scenario.

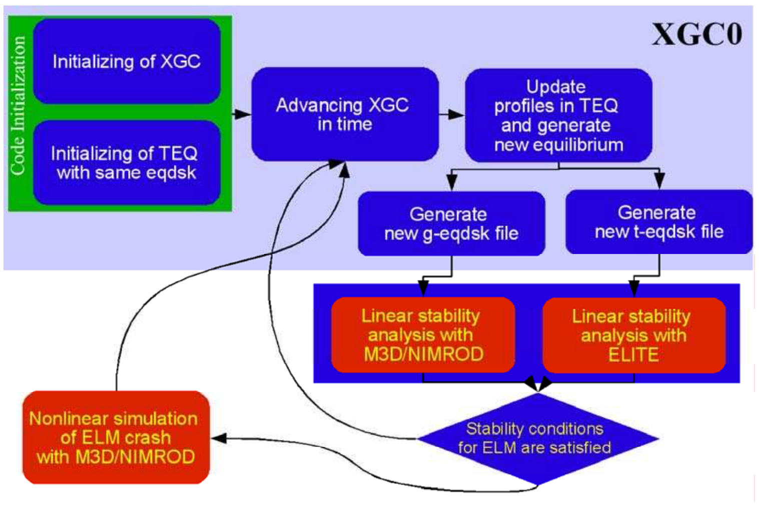

XGC0
Code Initialization
Initializing of XGC
Initializing of TEQ with same eqdsk
Advancing XGC in time
Update profiles in TEQ and generate new equilibrium
Generate new g-eqdsk file
Generate new t-eqdsk file
Linear stability analysis with M3D/NIMROD
Linear stability analysis with ELITE
Stability conditions for ELM are satisfied
Nonlinear simulation of ELM crash with M3D/NIMROD

Figure 5: CPES data flow uses magnetic flux surfaces for ITER equilibrium that are used in MHD studies.

lytic profiles for the edge plasma density and temperature and magnetic equilibrium data given in the form of a g-eqdsk file generated by the EFIT program [41] for a particular set of observations from a tokamak plasma experiment of interest. XGC0 simulates neoclassical transport in the presence of a self-consistent radial electric field and computes the plasma bootstrap current. A simple model for a source of neutrals at the wall boundary and an anomalous diffusive transport are included. The simulation host for XGC0 is typically the Cray XT3/XT4 supercomputer at ORNL named *jaguar*, although in these particular coupled simulations the computing resources required are not truly massive.

(More resources are needed when running XGC0 in conjunction with the related code XGC1, which simulates turbulent transport in the plasma edge.) At prescribed intervals, XGC0 outputs an *m3d.in* file containing radial profiles of the edge plasma density, temperature, and parallel current. This file is transferred to a processing host, which is usually the Infiniband cluster of dual Opteron nodes at ORNL named *ewok*, where it is read by the code M3D-OMP and used to produce an updated, high-resolution magnetic equilibrium eqdsk file. (We have also demonstrated the code coupling described here using an alternate simulation host such as the IBM SP-4 at NERSC named *seaborg* or an alternate processing host such as a local computing cluster.)

The updated magnetic equilibrium data is transferred back to XGC0 on the simulation host and read in to maintain self-consistency. Additionally, the eqdsk file is passed to the ELITE code for linear MHD stability analysis. The ELITE code also reads in edge plasma density data in the form of a p-eqdsk file, which is produced directly by the XGC0 code and must be transferred to the processing host. ELITE is launched using a script that runs each of the three components of the analysis, scans the output files for possible errors or problems, graphs some of the basic outputs such as the linear eigenmode, and checks whether ELITE has found the mode to be unstable or not. Each run of ELITE checks linear stability for a particular toroidal mode number n, so one can perform multiple, concurrent runs of ELITE to examine a range of typical intermediate n values for ELM instabilities.

If the ELITE code finds a mode with a linear growth rate that is above the typical threshold for ELM instability (usually taken to be one half of the ion diamagnetic drift frequency), then the kinetic XGC0 code is halted on the simulation host and the parallel M3D-MPP code is launched to perform a simulation of the nonlinear evolution of the ELM. The M3D-MPP code is usually run on the processing cluster because the nodes are normally available immediately and the resource needs are not so large, but this code could be launched on the simulation host instead. M3D-MPP requires an *omp.out* file produced by the M3D-OMP code that contains the initial plasma fluid quantities on the finite element mesh used by the MHD solver. Consequently, M3D-OMP must be run one more time, with input parameters that request production of the mesh file. Then the M3D-MPP code is launched using the batch system, and the diagnostic variable outputs are plotted as the ELM grows and expels particles and heat towards the boundary, allowing the plasma to relax and evolve towards a new equilibrium. Once the plasma profiles reach a quasi steady state in the M3D-MPP simulation, a new eqdsk file is produced and the cycle is complete. At this point, the M3D-MPP code can be halted and a new kinetic XGC0 simulation can be launched using the final equilibrium data from the previous ELM cycle.

The second code coupling scenario takes advantage of TEQ direct and inverse equilibrium solvers implemented in the XGC0 code. The TEQ module produces two eqdsk files at specific times that are controlled through the XGC0 input file parameters. The first file is a t-eqdsk file that contains the fixed-boundary equilibrium solution. This file is directly recognized by the ELITE code. The inverse equilibrium solution generated by TEQ has high poloidal resolution, which is a critical factor for the ELITE code. The second file generated by the TEQ module in XGC0 is a g-eqdsk file. This file contains the free-boundary equilibrium solution. It is used only if the ELITE code indicates that triggering conditions for an ELM crash are satisfied. In this case, the g-eqdsk file is used together with appropriate namelists in the *fluxgrid.in* and *nimrod.in* files by a NIMROD preprocessing utility called *nimset*. The *nimset* utility generates a new NIMROD grid and stores these equilibrium fields and initial perturbations on this grid in a binary file named *dump.00000*. This binary file together with namelists from *nimrod.in* is used by the NIMROD code, which runs typically on an IBM p575 POWER 5 system named *bassi* at NERSC. The details of Scenario 2 that include use of the TEQ and NIMROD codes are illustrated in Fig. 5.

4 Kepler framework for workflow automation

The kinetic-MHD coupling using the ORNL and other HPC resources poses multiple challenges for us. There are several simulation codes that should be executed on different resources in a coordinated and repetitive manner, delivering data from one to another in several stages, with the number of iterations depending on the status of the simulation. Certainly, manual control is impossible due to the large number of iterations. A completely script-based solution would fail because we have to get access to resources that are protected by One-Time Passwords (OTP) several times during one run. Additionally, we have different "tasks" to perform concurrently: processing NetCDF and other binary file output of XGC0, computing the new equilibrium in each coupling step, checking the linear MHD stability with ELITE, and executing M3D-MPP and NIMROD at the same time. This requires a parallel programming environment, for which scripting languages are cumbersome. Moreover, the tasks above are themselves long processing pipelines, applying several processing steps (transfer, conversion, image generation, etc.) for a stream of data items. Exploiting the performance improvement coming from pipeline parallel processing is necessary for us to be able to monitor the simulations and drive the coupling in a timely fashion.

We have chosen to use Kepler [42] due to its support of advanced computational models such as pipeline-parallel execution (discussed below). We prefer Kepler over other scientific workflow systems because the majority of them are grid-oriented (allowing only the creation of workflows where each component is realized as a job) or web-based (where a component is realized as a web service call). The resources at hand are not part of any grid or web-based software infrastructure. Even if they were, in a grid infrastructure there is no access to a job's output on the fly unless all output is put into external storage elements (a difficult requirement for a project with so many existing simulation codes). The quick access and fast processing of data and the fast turnaround in the execution of the many operations are essential for the kinetic-MHD coupling's timing requirements and are not achievable by using grid-based job submission. Another advantage over many alternative systems is that Kepler development is community-driven. Thus, different projects and teams such as CPES and SDM (Scientific Data Management) that use Kepler can collaborate to share development tasks and system improvements.

Another requirement of this code coupling task was that the workflow execution should not be constrained to the workflow system's graphical user interface. In the future, CPES workflows will be launched from the project dashboard (described in Section 5); meanwhile, they are executed from the command line or launched by a script. Kepler allows workflows to be executed from the command line, reading parameters from a text file, and thus is ideal for launching workflows either from scripts or web-based tools.

Kepler is an open-source scientific workflow system developed in the collaboration of several projects in different scientific and technology areas. The examples presented in [42] highlight the different application areas of scientific workflows. Roughly speaking, we can distinguish between several broad classes of workflows. There are *knowledge discovery workflows* that combine custom algorithms and analysis tools to elucidate new information from existing data. A workflow can be an *automation* of an otherwise manual procedure, or the *re-engineering* of a custom-built tool for a more generic, extensible and changing environment of resources and tools, thus forming the basis of a general "toolkit" for an application area. Other workflows are created to employ *high-performance* computing resources in a coordinated manner to achieve computationally intensive tasks.

Kepler is based on Ptolemy II [43], a modeling tool for heterogeneous, concurrent systems. One advantage of PtolemyII lies in a modeling and design paradigm called *actor-oriented modeling* [44]. Within this paradigm, a workflow is viewed as a composition of independent components called *actors*. Communication between actors happens through interfaces called *ports* by sending data encapsulated in *tokens*. In addition to these ports, actors have parameters that configure and customize their behavior. Given an interconnection of actors, however, there are many possible execution semantics that one could assign to the diagram. For example, actors might have their own thread of control, or their execution might be triggered by the availability of new inputs in a sequential order. A key property of PtolemyII is that the execution semantics is specified by a separate object called a *director*. The director defines how actors are executed and how they communicate with one another. Consequently, the execution model is less an emergent side-effect of the various interconnected actors and their (possibly ad-hoc) orchestration, and more a prescribed concurrent semantics as one might find in a well-defined concurrent programming language. The execution model defined by the director is called *a model of computation*. Patterns of concurrent interaction are factored out into the design of the directors, rather than being individually constructed by the designer of the workflow.

There are several models of computation realized in directors. The PN (Process Network) director is based on Kahn process networks [45] and allows actors to run concurrently in their own threads. In principle, write (or send) operations never block (the queue sizes are adjusted automatically if necessary). In contrast, actors block on read (or receive) operations (i.e., when an actor has not received all required input tokens). The

SDF (Synchronous Data-Flow) director performs static analysis on a workflow that guarantees absence of deadlocks, determines required buffer sizes, and optimizes the static and sequential scheduling of actor execution, but it poses certain restrictions on the possible process networks in order to be able to do that. Specifically, the token production and consumption rate of an actor must be known before runtime to allow SDF scheduling. In contrast, the DDF (Dynamic Data-Flow) director can be used for sequential (single-threaded) execution of a process network where the actors dynamically change the pattern of input consumption and output generation, thus allowing for data-dependent routing (e.g., conditional branches) and iterations (e.g., while loops). Other directors have been constructed for modeling Discrete Event (DE) systems, Continuous-Time (CT) models, Communicating Sequential Processes (CSP), and Finite State Machines (FSM) to mention just a few. Workflows can be nested in other workflows within a hierarchical model, where subcomponents can have an entirely different model of computation. In the CPES workflows, Process Networks that allow concurrent execution of independent operations are combined with DDF components that are inherently sequential operations. To the best of our knowledge, Kepler, through the underlying Ptolemy II directors, is the only scientific workflow system that allows such flexible combinations of models of computation.

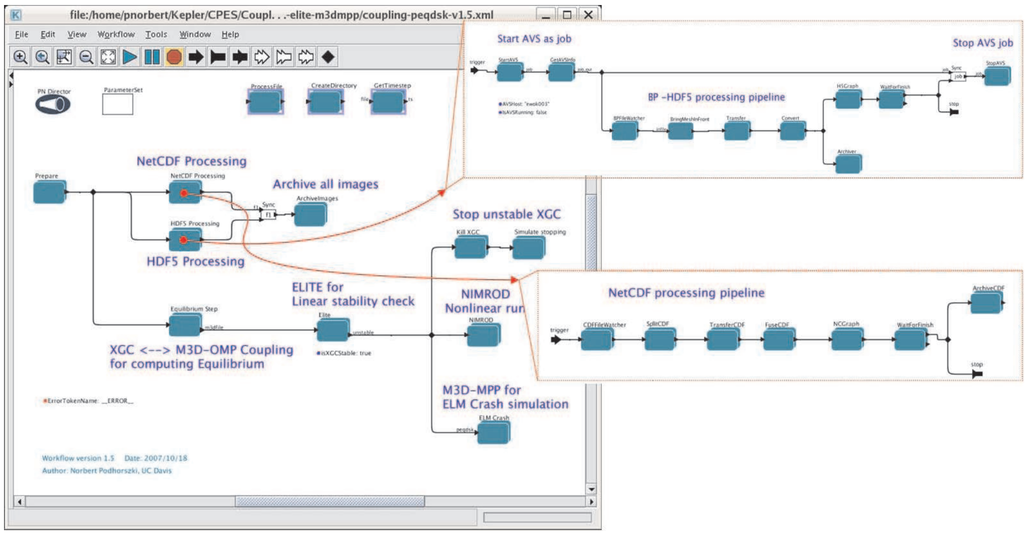

file:/home/pnorbert/Kepler/CPES/Coupl. . .-elite-m3dmpp/coupling-peqdsk-v1.5.xml
File Edit View Workflow Tools Window Help
PN Director
ParameterSet
ProcessFile
CreateDirectory
GetTimestep
Prepare
NetCDF Processing
Archive all images
HDF5 Processing
Stop unstable XGC
ELITE for Linear stability check
NIMROD Nonlinear run
XGC <--> M3D-OMP Coupling for computing Equilibrium
M3D-MPP for ELM Crash simulation
Start AVS as job
Stop AVS job
BP -HDF5 processing pipeline
NetCDF processing pipeline
Workflow version 1.5 Date: 2007/10/18
Author: Norbert Podhorszki, UC Davis

Figure 6: CPES kinetic-MHD coupling workflow in Kepler realizing the XGC0 + M3D-OMP + ELITE + M3D-MPP scenario. Each component is itself a hierarchical subworkflow. On the right side of the figure, the HDF5 Processing and the NetCDF Processing steps are shown.

Fig. 6 shows the top level of the Kepler workflow that coordinates the kinetic-MHD coupling. Among the important elements on the Kepler canvas is the PN Director (upper left-hand corner) that defines task-parallel execution for independent actors on different branches or pipelines and pipeline-parallel processing for actors connected into a pipeline. The *ParameterSet* actor reads workflow parameters from a text file, thus providing a practical way to deploy a workflow system-wide and allow it to be used with different settings. The three components on the top show the class/instance concept of Kepler. They are subworkflows designated as classes, so they can be instantiated several times in the workflow graph while using the same code (just like the atomic Java actors are used). The instances can be parameterized differently, but any change to the workflow graph of the actor means a global change for all instances.

ProcessFile is the most important actor for the CPES project. It is a generic actor that executes a command on a remote machine using an (already established) SSH connection or, alternatively, on the local machine using the Java Runtime environment. The command is given as a string parameter containing macro definitions to denote the actual input file and directory, the output file and directory, and the host. The actor can store the command string on successful execution and can check all future commands against it, skipping commands if they have already been executed. This "lightweight checkpointing" feature allows us to stop and restart the workflow any time, without affecting the simulation, and to continue processing the output where we left off. The actor also passes special tokens that denote the end of the simulation (*StopFile*) or a failed operation upstream on the pipeline. With appropriate parameterization we can realize all remote operations of the workflow by using this actor, transferring files from the simulation site to the processing site using scp or bbcp (third-party transfer), and executing all external codes and scripts on the processing site. Note, for example, that five of the *NetCDF Processing* pipeline's seven components (shown on the right side of Fig. 6) are simply different instances of the same *ProcessFile* actor. The transparency of local and remote sites in the workflow allows us to execute the workflow on the data processing site for production use or on a laptop/desktop for testing, demonstration or tutorials (with all simulations executed and data stored at ORNL), without ever changing the workflow itself. Other CPES actors provide SSH session operations, remote file operations (file copy, directory creation, remote directory listing), archival operations, session logging, etc. Those actors, along with the restarting capability of workflows and techniques used to ensure robustness of the workflows, are described in detail in [46].

The workflow screenshot graphically presents the coarse steps of the coupling. XGC0 is not represented, since the simulation is currently submitted independently by the user, and the workflow is started afterwards. After a number of preparation steps (logins to all involved machines, creation of directories on the processing site, and copying of input and configuration files of each code used in the workflow into the appropriate places), three independent pipelines are started, ready to monitor and react to output from the XGC0 simulation. These pipelines start processing output of each coupling or diagnostic procedure as soon as they find it.

Monitoring of XGC0 is realized by two pipelines: *NetCDF Processing* starts with watching for 1D diagnostic variables of XGC0 stored in NetCDF files. As each file grows (they are extended after every diagnostic period), the workflow splits (takes the most recent data entry), transfers and merges the data to mirror XGC0's output on the processing site efficiently. Finally, images are generated using *xmgrace* for all variables in the output

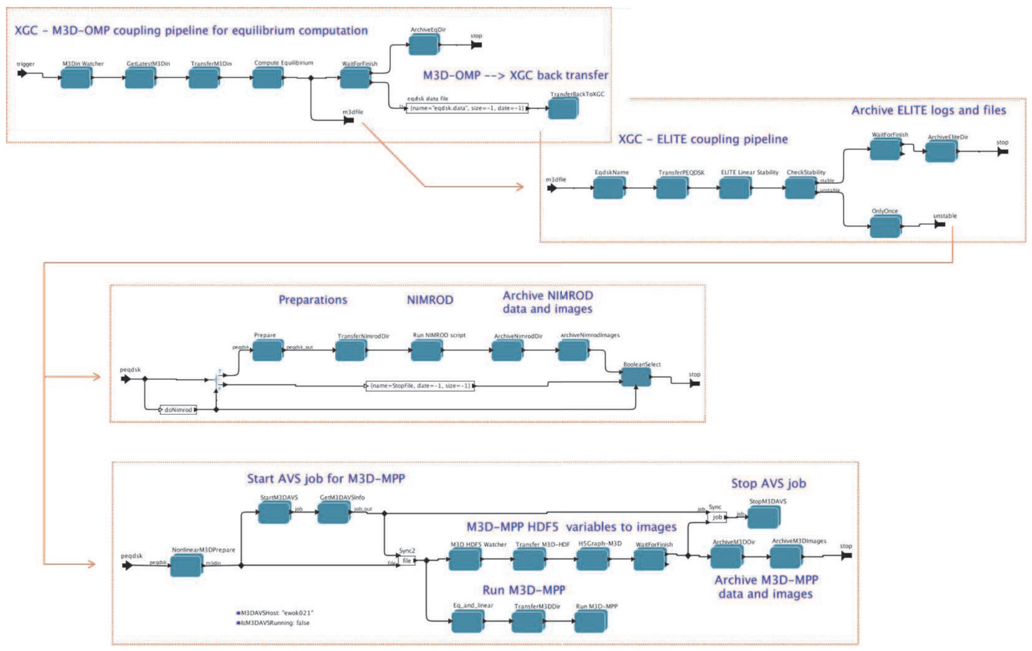


Figure 7: The coupling pipelines performing the M3D-OMP, ELITE, NIMROD and M3D-MPP steps. The additional arrows indicate the connections between those steps.

for each diagnostic time step and placed into a predefined directory, which is shown by the dashboard. The *HDF5 Processing* pipeline's role is similar, but there are important differences. For each diagnostic time step, XGC creates new BP files (a custom binary format for efficient I/O writing); hence, there are no split and merge steps when transferring them to the processing site. The BP files are converted to HDF5 using an external code and then images are created for all 2*D* slices of the 3*D* data stored in those files using an *AVS/Express* network. For this purpose, the pipeline starts AVS/Express as a job on the processing site and then uses it as a (private) service by making imaging requests to it.

The M3D-OMP pipeline (top of Fig. 7) realizes the loose cyclic coupling between XGC0 and M3D-OMP. From XGC0, a small data file *m3d.in* is transferred to the processing site where M3D-OMP is executed (submitted as a job). The new equilibrium is then transferred back to the simulation's directory. The result also triggers the execution of the *ELITE* pipeline, which transfers the p-eqdsk data (plasma density profile) from XGC0 and runs ELITE, again submitted as a job. A wrapping script around ELITE performs all the tasks: prepare job, submit job and wait for results, check all output files for errors or problems, generate output plots using IDL, and determine whether the XGC0 simulation has become MHD unstable. All these subcomponents could also be realized as subworkflows in Kepler if more detailed control were needed inside the workflow. However, scripts are still the easiest way to implement these individual tasks involving several system calls and program executions. The result of the script is a simple "control file" containing the string *stable* or *unstable*, which is checked in the last actor of this subworkflow. If XGC0 is determined to be MHD unstable, both NIMROD and M3D-MPP are launched, while the workflow stops XGC0. The *NIMROD* pipeline is simple because, similar to ELITE, the majority of the work is performed by an external script that does the following tasks: prepare a NIMROD job, submit it and wait for it to run, then generate images from its output using *xdraw*. The M3D-MPP pipeline, however, is more complex. It can be used as a stand-alone workflow as well, to execute M3D-MPP and monitor it. The workflow generates images from its HDF5 output using AVS/Express, so the structure is similar to the *HDF5 Processing* pipeline described above. It is worth noting that the AVS/Express job is submitted directly from the workflow, not from a script. Thus, the workflow has the power to wait for the AVS/Express job start-up, perform other tasks while the job is running, and stop the job at the end of the processing. This capability is enabled by a set of job-related actors in Kepler that support several job managers (PBS, LoadLeveler, SGE, Condor and "process fork") and use SSH to connect to remote computing sites.

5 Dashboard tool for simulation data and resource monitoring

The need for dashboards for monitoring the scientific process and collecting provenance data has been recognized for many years [47]. Monitoring can be partly the collection of the provenance and meta-data, but it also may require collection of additional information and display of information in a very user-friendly format. A dashboard allows us to provide run-time tracking, problem determination, computational steering, and other workflow-related feedback mechanisms to the users of this system.This information may be useful to provide fault-tolerance information, performance monitoring, and monitoring of the actual data created in the simulation.

As part of the CPES project, we have been creating a scientific dashboard to simplify the number of tasks our users need to perform to monitor their workflows. We view the dashboard as a place where a user can learn about DOE computing resources and query these resources, so that they can determine when their simulations will run and how much time they have used on these large systems. Users can also access on the dashboard all of the information that was generated from their current and previous simulations.

Because of their evolutionary and exploratory nature, frequent changes are often an integral part of a scientific workflow lifecycle for the CPES suite of codes. Therefore, is critical to record provenance information (e.g., the lineage of data and processes) in a way that is consistent, persistent, and easily retrievable and auditable. Related to this is the ability to steer the workflows and the associated computational tasks through use of run-time dashboards, analytics and process feedback loops. The CPES dashboard system allows the display of all of the meta-data that is generated either by the workflow or by the user's interaction with the dashboard itself. Part of this system includes methods to annotate the information on the dashboard and save this in a database.

In order for the system to be useable by the computational scientists who are our targeted users, we wanted to use Web2 technologies [48]. Here data is the driving force of the technologies delivered to the user. Web2 technologies generally refer to web-based services and communities that have a strong reliance on data access and aim to facilitate collaboration and the sharing of data while adhering to the concept of lightweight programming models. Beyond these basic goals of Web2 technologies, we want to provide a system that uses the most common web browsers (e.g., Firefox, Internet Explorer) and has a low memory overhead. We also want a system that would work fast, which implies that we need asynchronous movement of information from our back-end system to the dashboard. We choose not to use Java because we feel that the momentum in industry is moving towards Asynchronous Java and XML (AJAX) [49], and Flash [50] for fast delivery of content-rich information. Another aspect of choosing these technologies of Java is the ease-of-programming with AJAX and Flash. Initial development and hardening of our dashboard took less than one person-year of development with the help of the CPES and SDM teams.

Another very important aspect of the dashboard is to provide a secure environment in which users can access their data and create electronic notes for all of the data associated with their simulations. Initially when a user logs onto our system, they must log in with a One-Time Password. Since the users must be able to submit jobs from the system, we need to use a technique that issues an X509 certificate based on the OTP. This certificate allows users to submit jobs, query the Cray supercomputer at ORNL, and access data from the system. We are currently running the system without certificates, so that we don't have access to these feedback mechanisms. We plan to put these mechanisms in place in the future, and to provide additional mechanisms for accessing data on these systems.

Back-end architecture

The ORNL Dashboard is mainly using PHP at the back end. Information is pushed into the MySQL database at ORNL using shell scripts and python scripts, and then later retrieved via PHP to be displayed using AJAX or Flash on the client side.

1. Python scripts. Some of these scripts are running as cron jobs at the back end, pushing frequently updated information such as the status of the machine queues into the database. This is the simple case of machine monitoring, where the information is always available and current as long as the dashboard back-end machine (*ewok*) is up and running. In the case of simulation monitoring, the first step is to run the script pushing the "cheat sheet" for a specific run into the database. The same script that starts the simulation workflow also calls a python routine that inserts metadata about this simulation into a MySQL table (i.e., the "cheat sheet"). From that point on, the Dashboard knows where to look for data associated with a particular username, machine name, and job ID listed in the machine queues. The additional python scripts are called from PHP to pre-process information such as the table of variables minima and maxima for the run. This file contains data pertaining to all time steps, but only information for one time step is displayed at a time.

2. Database. As previously mentioned, the database tables may be filled by cron jobs running on *ewok* or by the initial script that starts the workflow or by PHP scripts directly from the Dashboard. Hence the current information stored in the database is of the following types:

   (a) System information – These tables contain the machine queues for seven DOE machines; i.e., username, job ID, job status, start time, number of processors used, etc. The individual usage on each machine is also stored here.

   (b) Simulation meta-data – The simulation "cheat sheet" is unique to a username, a machine name, and a job id. Each job has a single entry or cheat sheet ID in the database. This ID is linked to a series of crucial details about the run including the simulation name, the path to the simulation data, and the path to archived data in the High Performance Storage System at ORNL.

   (c) User tables – This information is stored while users are using the Dashboard to create user sessions, and it allows a level of customization of the Dashboard. In machine monitoring mode, the user can save quick annotations to their runs. For simulation monitoring, these database tables store information such as details about the Flash graphics viewer: number and name of variables loaded in each space (or *desktop*), etc.

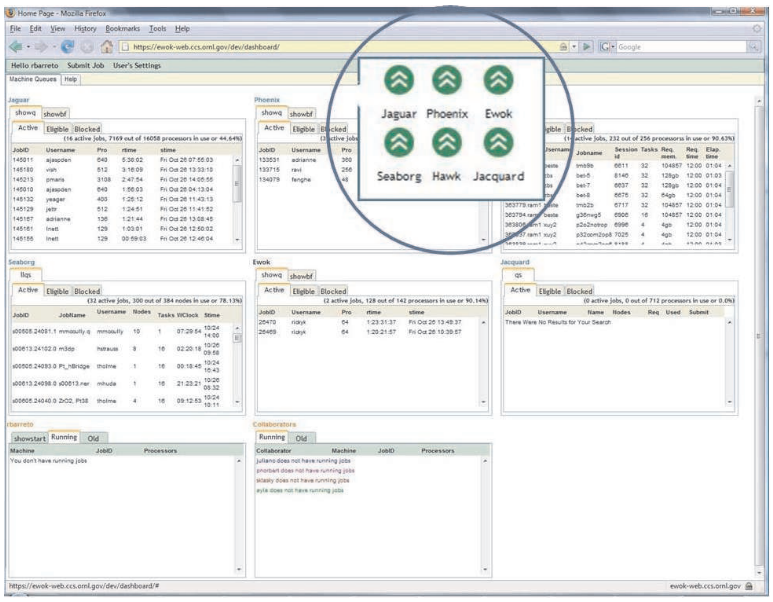
   (d) Administrator tables – Additionally, data is entered and updated manually by super users who determine access permissions on data stored in the database. For instance, users are allowed to see other users' runs as long as their usernames are stored in the collaborators table in the database.

Naturally, several of the previously mentioned sections of the database continue to be developed over time, but the only supplementary information currently missing from the database is the provenance information (especially the data provenance).

Machine monitoring

The Dashboard homepage displays a detailed view of the status of the various machines at the user's request. In this mode, the technology used is AJAX and PHP. The link at the top center of this page displays images in an AJAX tooltip that indicate whether a machine is up or down (see Fig. 8). The AJAX tabs and tables below the link contain more detailed information. They hold the machine queues arranged by active (running), eligible (queued), or blocked (on hold) jobs. This page also gives information on machine usage and availability. The user has the possibility to query machines (currently only *ewok*) for estimated start time before submitting a job. The Dashboard also offers the option to directly log into a machine using the web-based terminal *AjaxTerm* [51]. Finally, the two additional tabs in the machine monitoring display the result of a query to retrieve the user's running and previous jobs, as well as those of his or her collaborators.

As more machines become available to users, the machine monitoring page will be updated to include them. Permissions may need to be set to establish which machines users are allowed to see. Some scientists that use the Dashboard don't have accounts


Home Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://ewok-web.ccs.ornl.gov/dev/dashboard/
Google
Hello rbarreto Submit Job User's Settings
Machine Queues Help
Jaguar Phoenix Ewok
Seaborg Hawk Jacquard

Jaguar — showq | showbf — Active | Eligible | Blocked

(16 active jobs, 7169 out of 16058 processors in use or 44.64%)

JobID	Username	Pro	rtime	stime
145011	ajaspden	640	5:38:02	Fri Oct 26 07:55:03
145180	vish	512	3:16:09	Fri Oct 26 13:33:10
145213	pmaris	3108	2:47:54	Fri Oct 26 14:05:55
145010	ajaspden	640	1:56:03	Fri Oct 26 04:13:04
145132	yeager	400	1:25:12	Fri Oct 26 11:43:13
145129	jettr	512	1:24:51	Fri Oct 26 11:41:52
145167	adrianne	136	1:21:44	Fri Oct 26 13:08:45
145161	lnett	129	1:03:01	Fri Oct 26 12:50:02
145155	lnett	129	00:59:03	Fri Oct 26 12:46:04

Phoenix — showq | showbf — Active | Eligible | Blocked

JobID	Username	Pro
133531	adrianne	360
133715	ravi	256
134079	fenghe	48

Hawk — Eligible | Blocked

(1 active jobs, 232 out of 256 processorss in use or 90.63%)

JobID	Username	Jobname	Session id	Tasks	Req. mem.	Req. time	Elap. time
	beste	tmb9b	6611	32	104857	12:00	01:04
	tbs	bet-5	8146	32	128gb	12:00	01:03
	tbs	bet-7	6637	32	128gb	12:00	01:04
	tbs	bet-8	6676	32	64gb	12:00	01:04
363779.ram1	beste	tmb2b	6717	32	104857	12:00	01:04
363794.ram	beste	g36nwg5	6906	16	104857	12:00	01:04
363806.ram1	xuy2	p2o2notrop	6996	4	4gb	12:00	01:04
363 37.ram1	xuy2	p32oom2op8	7025	4	4gb	12:00	01:04

Seaborg — llqs — Active | Eligible | Blocked

(32 active jobs, 300 out of 384 nodes in use or 78.13%)

JobID	JobName	Username	Nodes	Tasks	WClock	Stime
s00505.24081.1	mmccully.q	mmccully	10	1	07:29:54	10/24 14:00
s00613.24102.0	m3dp	hstrauss	8	16	02:20:18	10/26 09:58
s00505.24093.0	Pt_hBridge	tholme	1	16	00:18:45	10/24 16:43
s00613.24098.0	s00613.ner	mhuda	1	16	21:23:21	10/26 08:32
s00605.24040.0	ZrO2, Pt38	tholme	4	16	09:12:53	10/24 10:11

Ewok — showq | showbf — Active | Eligible | Blocked

(2 active jobs, 128 out of 142 processors in use or 90.14%)

JobID	Username	Pro	rtime	stime
26470	rickyk	64	1:23:31:37	Fri Oct 26 13:49:37
26469	rickyk	64	1:20:21:57	Fri Oct 26 10:38:57

Jacquard — qs — Active | Eligible | Blocked

(0 active jobs, 0 out of 712 processors in use or 0.0%)

JobID	Username	Name	Nodes	Req	Used	Submit
There Were No Results for Your Search						

rbarreto — showstart | Running | Old

Machine	JobID	Processors
You don't have running jobs		

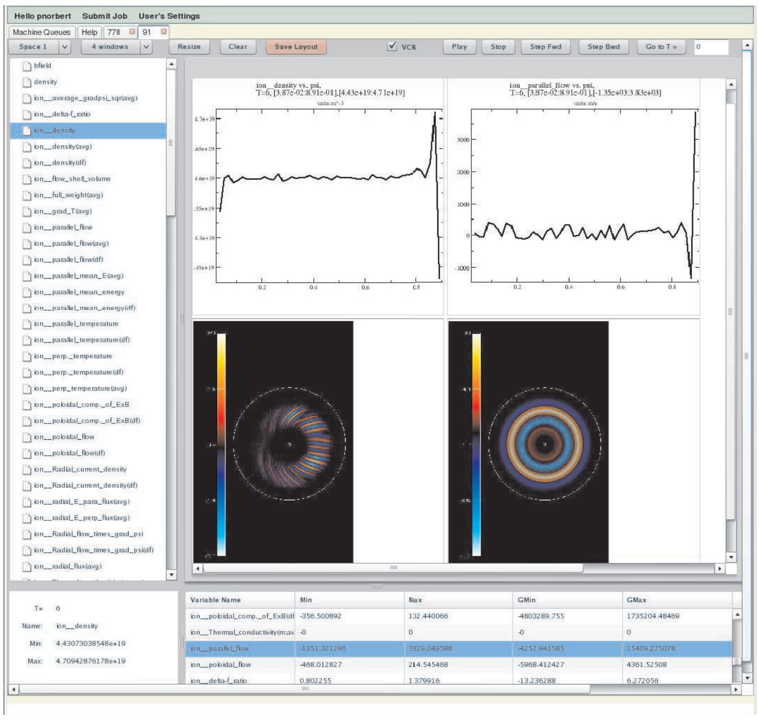
Collaborators — Running | Old

Collaborator	Machine	JobID	Processors
juliano does not have running jobs			
pnorbert does not have running jobs			
sklasky does not have running jobs			
ayla does not have running jobs			

https://ewok-web.ccs.ornl.gov/dev/dashboard/#

ewok-web.ccs.ornl.gov

Figure 8: Machine Monitoring Page – The first six tabs display DOE machine queue command outputs. The bottom two tabs parse the runs to extract the user's job and the jobs of his or her collaborators. The green arrows indicate that the machines are up.


Hello pnorbert Submit Job User's Settings
Machine Queues Help 778 91
Space 1
4 windows
Resize
Clear
Save Layout
VCR
Play
Stop
Step Fwd
Step Bwd
Go to T =
0
ion__density vs. psi,
T=6, [3.87e-02:8.91e-01],[4.43e+19:4.71e+19]
units m^-3
ion__parallel_flow vs. psi,
T=6, [3.87e-02:8.91e-01],[-1.35e+03:3.83e+03]
units m/s

bfield
density
ion__average_gradpsi_sqr(avg)
ion__delta-f_ratio
ion__density
ion__density(avg)
ion__density(df)
ion__flow_shell_volume
ion__full_weight(avg)
ion__grad_T(avg)
ion__parallel_flow
ion__parallel_flow(avg)
ion__parallel_flow(df)
ion__parallel_mean_E(avg)
ion__parallel_mean_energy
ion__parallel_mean_energy(df)
ion__parallel_temperature
ion__parallel_temperature(df)
ion__perp._temperature
ion__perp._temperature(df)
ion__perp_temperature(avg)
ion__poloidal_comp._of_ExB
ion__poloidal_comp._of_ExB(df)
ion__poloidal_flow
ion__poloidal_flow(df)
ion__Radial_current_density
ion__Radial_current_density(df)
ion__radial_E_para_flux(avg)
ion__radial_E_perp_flux(avg)
ion__Radial_flow_times_grad_psi
ion__Radial_flow_times_grad_psi(df)
ion__radial_flux(avg)

T=	6
Name:	ion__density
Min:	4.43073038548e+19
Max:	4.70942876178e+19

Variable Name	Min	Max	GMin	GMax
ion__poloidal_comp._of_ExB(df	-356.500892	132.440066	-4803289.755	1735204.48469
ion__Thermal_conductivity(m:av	-0	0	-0	0
ion__parallel_flow	-1351.321296	3329.049588	-4252.941585	15409.275078
ion__poloidal_flow	-468.012827	214.545468	-5968.412427	4361.52508
ion__delta-f_ratio	0.802255	1.379916	-13.236288	6.272056

Figure 9: Simulation Monitoring Page – The variables to the left can be dropped into the graphics tiles. The user can view graphs, movies, or text files, in addition to the table of variables minima and maxima at the bottom of the page.

on all the supported systems. Even in the case where a user has an account on every machine displayed on the main page, this information could become overwhelming and unnecessary. A mechanism can be provided to display and rearrange only computers of special interest to the user.

Simulation monitoring

The simulation page is mainly based on Flash and PHP. The basic idea is to load a tree view of the variables for a specific run and visualize the evolution of the associated data through time. The Graphical User Interface for this mode includes four spaces (desktops), each including up to 16 graphics frames. Users can visualize images or movies by dragging and dropping variables from the tree-view to these tiles (see Fig. 9). Other types of data include variables local and global minima and maxima, ASCII files associated with special variables, etc. The list will expand and change with different simulations.

The Dashboard will include provenance information in the near future. Provenance is the history of the data, execution and conditions applied to a workflow run [47]. This relevant information should be displayed in a user-friendly format on the Dashboard. Considering that provenance includes process, system, workflow and simulation data provenance, some information might be more important for the developer than for the user, and the Dashboard should be aware of these differences. Another important piece not yet included in the current Dashboard is a data-delivery system from the simulation directory or from HPSS to the user's machine or to a user-specified remote computer. A third focus being developed along with provenance and data delivery is user annotations of images and movies. These annotations will be stored in the database and linked to the applicable simulation data, so that they can later be jointly queried. To take the annotations further, the Dashboard should include an e-book (electronic notebook). This option will provide the user with a way to insert notes and images in an expanding document associated with a particular simulation name. As these different pieces are added to the Dashboard, its user friendliness and performance should continually improve.

6 Application of Kepler workflow to coupled code simulation of ELM cycle

Let us now bring together the preceding discussion of the required physics code components and computer science tools and technologies by examining the application of a Kepler scientific workflow and use of the ORNL Dashboard tool to simulate H-mode pedestal growth and the onset of ELM instability. For this particular example, we focus on the first code coupling scenario described in Section 3 of this paper and illustrated in Fig. 4. An EFIT equilibrium data file for DIII-D shot number 096333 from time 0.337 msec is selected as the starting point for our XGC0 kinetic code, along with model profiles for the initial edge plasma density and temperature. The XGC0 code is configured to run with 320,000 ion particles distributed across 128 processing cores on the Cray $XT3/XT4$

jaguar at ORNL. The simulation end time is chosen to be roughly 100 ion toroidal transit periods; in the coupled simulation scenario, however, the kinetic code will be halted whenever linear instability of ELMs is detected. A particular set of XGC0 input file parameters determine whether the code runs in a coupled mode, and, if so, how often coupling data is to be written out. (It should be noted that there is a practical limit on how frequently code coupling data can actually be exchanged, due to the finite amount of time required to perform ELM linear stability analysis.) In this instance, a coupled simulation is chosen with the *m3d.in* plasma profile data and *peqdsk* density data being output once every 500 time steps (equal to roughly one ion transit period), along with standard XGC0 diagnostic variables such as the plasma density and temperature profiles and the self-consistent radial electric field. An equivalent *uncoupled* XGC0 simulation would require about 75 minutes of wall clock time to complete.

This XGC0 production run is launched on *jaguar* in the conventional manner using a PBS job script. Job submission can be done either in a standard terminal window or via the Dashboard using AjaxTerm. (It is planned for the near future to have Kepler workflows that include XGC0 job submission as one of the specified tasks.) The Dashboard environment offers the user a very convenient means of examining the distributed computing resources that are available prior to starting a simulation, as well as monitoring the job queue to determine when execution begins. At any time after XGC0 job submission, the Kepler coupling workflow is launched (once again on the simulation host machine *jaguar*) using a shell script that takes two arguments: a shot ID number that acts as a unique label for this run, and the job ID number associated with the XGC0 job that has been submitted. The workflow actually runs on the processing host *ewok*, so a login with OTP is required in this case. By convention, all of the CPES workflows create a run-specific simulation output directory labeled by the shot ID number within the "workflow" directory of a user's workspace on the processing host, so the user must ensure that such a "workflow" directory exists prior to the first workflow run. The coupling workflow here has a graphical representation similar to that displayed in Fig. 6. Each workflow has its own configuration file that specifies details about which computing hosts and directories are used, how often certain tasks are performed, which utility commands are invoked, and so forth. (These parameters should remain mostly constant from one run to the next, so that the configuration files are rarely edited.)

Once the Kepler coupling workflow is running, it creates a hierarchy of output directories on the processing host for the different stages/components of the coupled simulation and then begins looking for diagnostic or coupling data to process. The user can monitor this activity in real time, either by looking in the output directory hierarchy on the processing host or, more conveniently, by using the Dashboard to examine the running job on the simulation host as described in Section 5. (Note that the Kepler workflow running on *ewok* is not itself visible on the Dashboard because it is not run via the job queue; nevertheless, the outputs of tasks driven by the workflow are associated with the XGC0 job and can be monitored on the Dashboard.) XGC0 diagnostic variables that are plotted on the Dashboard will have their plots updated automatically as new

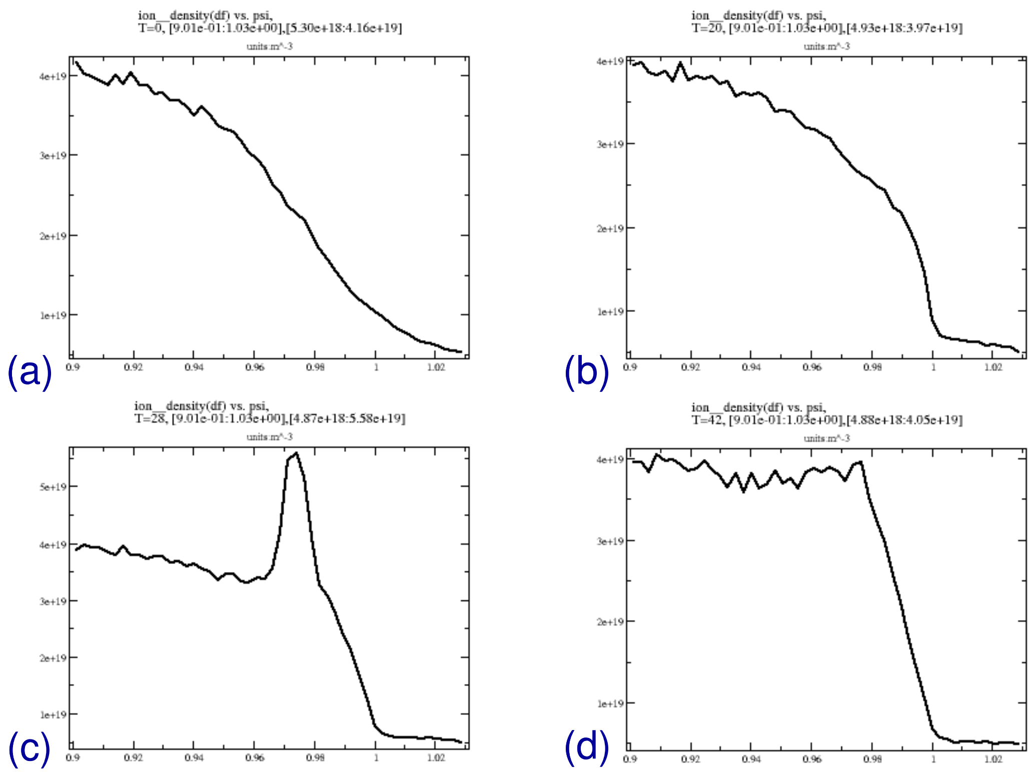


Figure 10: Evolution of perturbed ion density, from initial profile (a) until just prior to ELM onset (d).

data is written out by XGC0 and then processed by the relevant pipeline in the Kepler workflow. The Dashboard can be placed in "VCR" mode, in which it will record all such updates and create a Flash animation file containing the history of all variables displayed on the Dashboard. For the sample coupled simulation we are studying here, a sequence of plots of the perturbed ion density shows the natural buildup of the edge pedestal due to the interplay of neutral penetration from the wall boundary and ion orbit loss near the separatrix in the presence of a self-consistent radial electric field (Fig. 10).

As the XGC0 run proceeds, a series of *m3d_NNN.in* and *peqdsk.NNN* files are written out, where *NNN* here represents the coupling period number. The Kepler workflow looks for the most recent such files and transfers them to *ewok* for processing. (Note that this procedure results in a non-deterministic set of coupling times being analyzed. We are currently working on an approach that will allow one to specify within the workflow precisely which coupling times are to be processed.) M3D-OMP is run on the *m3d.in* file to produce an updated *eqdsk* file, which is transferred back to XGC0 but also given to ELITE along with the *peqdsk* file for linear stability analysis. Output variables from this analysis such as the normalized pressure gradient and plasma current are plotted automatically using IDL scripts and made available to the Dashboard.

The most crucial output data from ELITE is the computed linear growth rate of the

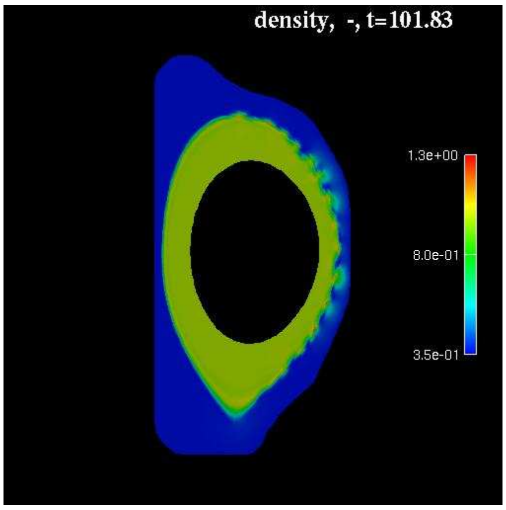


Figure 11: Density in M3D nonlinear MHD simulation.

eigenmode, which is compared with a threshold value (based on local theory) of one-half the ion diamagnetic drift frequency to determine the moment of ELM instability. For this example run, the $n=8$ mode tends to have the strongest linear growth rate and reaches the instability threshold first. The XGC0 simulation is configured to output coupling data 100 times during the entire run, which in this case is about once every 45 seconds. However, the time required for the M3D-OMP run to update the *eqdsk* file followed by the ELITE run to check stability turns out to be about 8 minutes on a single *ewok* processing core. Thus, the coupled system as configured can update the equilibrium and test for stability only every 10^{th} or 11^{th} coupling period. In this case, four series of M3D-OMP and ELITE runs were performed by the Kepler workflow before a linear growth rate exceeding the instability threshold was reported for coupling data set number 40.

At this point, the XGC0 job is terminated by the workflow, and a single-step linear run of M3D-OMP is executed in order to produce the *omp.out* mesh file needed by the parallel M3D-MPP code. Once this preliminary step is complete, the workflow submits a job script to the batch queue on *ewok* (or alternatively on *jaguar* or *seaborg*, for example) to run the nonlinear ELM simulation. In this example, a 128-process M3D-MPP job was run with 16 separate poloidal planes and approximately 8,000 elements in each plane of the M3D mesh. The code performed 10,000 time steps (equivalent to more than 100 Alfvén periods of simulated time) in about 100 minutes of wall clock time. As it turns out, this was enough time for the ELM to grow and saturate (see Fig. 11), but longer runs show that a simulation several times longer is needed for complete nonlinear saturation and dissipation of the mode energy to occur. Our goal is for the nonlinear simulation to reach a quasi steady-state in which the magnetic geometry near the separatrix has healed and the edge plasma profiles cease to evolve. This can be challenging with a resistive MHD initial value code like M3D because the nonlinear ELM crash can generate large density and heat releases from the plasma that take a long time to dissipate. The resistive MHD model is inexact in the vacuum region, and there is no "absorbing" boundary condition at the outer wall (although you can mock this behavior by adjusting heat dissipation characteristics in the input parameters). Furthermore, the ELM crash generates fine-scale structures in the perturbed density that can cause numerical difficulties related to grid resolution in certain cases. Nevertheless, we have been able to address these challenges and produce stable nonlinear ELM simulations that eventually reach a steady state. We are currently working to create an automated means of detecting the attainment of quasi steady-state in the M3D-MPP nonlinear ELM simulation, so that a final equilibrium can be produced and used to begin a new ELM cycle.

7 Conclusion and discussion

A coupled simulation has been successfully demonstrated involving advanced kinetic and MHD code components that are utilized in a coordinated fashion to model the edge pedestal buildup of an H-mode plasma and the subsequent ELM crash that can occur when extremely sharp gradients develop. This system loosely couples the kinetic edge code XGC0 with a variety of MHD equilibrium solvers and analysis packages, including TEQ, ELITE, M3D, and NIMROD. A set of scientific workflows developed in the Kepler framework are used to orchestrate the actions of the coupled simulation, launching code components as needed and performing the requisite data transfers between codes and analysis of simulation outputs. A web-based dashboard tool developed at ORNL is used to check for computational resources and monitor the coupled simulation results in real time.

This coupled simulation will be used to facilitate the exploration of the peeling- ballooning mode instability boundary in (j,α) parameter space, where j represents the normalized plasma current density in the edge region and α is the normalized pressure gradient. As the physics studies proceed, improvements will continue to be made to the computational tools that have been developed. Further refinements to the coupling workflows are needed, for example, to allow for prescribing in advance which coupling data sets are to be analyzed, which is needed to guarantee reproducibility of the coupled simulation. Straightforward modifications should make it possible for concurrent runs of ELITE to be used to scan multiple n values at once for linear stability, or to launch simultaneously nonlinear ELM simulations from the same initial profiles with the M3D-MPP and NIMROD codes and perform cross-verification of the models. Continued developments of the ORNL Dashboard tool will include an e-book feature for annotating current and previously archived simulation results and an interface with the MDS+ database format that is used to store experimental data from many present-day tokamak devices. The latter feature should greatly ease the process of configuring simulations of specific experimental shots and conditions, as well as performing code validation against observational data.

Acknowledgments

This work is part of the ongoing research activities within the SciDAC Fusion Simulation Prototype (FSP) Center for Plasma Edge Simulation, which is supported by the Office of Fusion Energy Sciences and the Office of Advanced Scientific Computing Research within the U.S. Department of Energy. We are grateful to the National Center for Computational Science at Oak Ridge National Laboratory and the National Energy Research Supercomputer Center at Lawrence Berkeley National Laboratory for access to and support of their computing resources. Finally, we would like to thank the members of the M3D, ELITE, TEQ, and NIMROD development teams for providing code access and their expertise in utilizing each of these code components effectively.

References

[1] R. J. Groebner, K. H. Burrell and R. P. Seraydarian, Phys. Rev. Lett. 64, 3015 (1990).
[2] C. S. Chang and S. Ku, Phys. Plasma 11, 5626 (2004).
[3] W. Park, E. V. Belova, G. Y. Fu, X. Tang, H. R. Strauss and L. E. Sugiyama, Phys. Plasmas 6, 1796 (1999).
[4] H. R. Wilson, P. B. Snyder, G. T. A. Huysmans and R. L. Miller, Phys. Plasmas 9, 1277 (2002).
[5] P. B. Snyder, H. R. Wilson et al., Phys. Plasmas 9, 2037 (2002).
[6] R. G. Littlejohn, J. Plasma Phys. 20, 111 (1983).
[7] R. J. Procassini, C. K. Birdsall and B. I. Cohen, Nucl. Fusion 30, 2329 (1990).
[8] A. H. Boozer and G. Kuo-Petravic, Phys. Fluids 24, 851 (1981).
[9] W. X. Wang, N. Nakajima, M. Okamoto and S. Murakami, Plasma Phys. Control. Fusion 41, 1091 (1999).
[10] L.L. LoDestro and L. D. Pearlstein, Phys. Plasmas 1, 90 (1994).
[11] A. H. Kritz, G. Bateman, J. Kinsey et al., Comput. Phys. Commun. 164, 108 (2004).
[12] J. A. Crotinger, L. LoDestro, L. D. Pearlstein, A. Tarditi, T. A. Casper and E. B. Hooper, CORSICA: A Comprehensive Simulation of Toroidal Magnetic-Fusion Devices, Technical Report UCRL-ID-126284 (Lawrence Livermore National Laboratory, April 1997).
[13] R. Andre, D. McCune, J. Menard, D. Pearlstein, L. Lodestro and J. Carlsson, "New MHD Equilibrium Solver Options in TRANSP", APS Meeting Abstracts, page 1129P (October 2006).
[14] J. W. Connor, R. J. Hastie, H. R. Wilson and R. L. Miller, Phys. Plasmas 5, 2687 (1998).
[15] P. B. Snyder, H. R. Wilson et al., Nucl. Fusion 44, 320 (2004).
[16] P. B. Snyder, K. H. Burrell, H. R. Wilson et al., Nucl. Fusion 47, 961 (2007).
[17] A. W. Leonard, T. H. Osborne et al., Phys. Plasmas 10, 1765 (2003).
[18] T. W. Petrie et al., Nucl. Fusion 43, 910 (2003).
[19] P. B. Snyder, H. R Wilson and X. Q. Xu, Phys. Plasmas 12, 056115 (2005).
[20] W. P. West et al., Nucl. Fusion 45, 1708 (2005).
[21] K. H. Burrell et al., Phys. Plasmas 12, 056121 (2005).
[22] M. R. Wade et al., Phys. Rev. Lett. 94, 2250011 (2005).
[23] T. Evans et al., Nature Physics 2, 414 (2006).
[24] A. Kirk et al., Plasma Phys. Control. Fusion 46, 551 (2004).
[25] D. A. Mossessian, P. Snyder, A. Hubbard et al., Phys. Plasmas 10, 1720 (2003).
[26] L. L. Lao, Y. Kamada, T. Okawa et al., Nucl. Fusion 41, 295 (2001).

[27] N. Oyama et al., Nucl. Fusion 45, 871 (2005).
[28] C. R. Sovinec, A. H. Glasser, T. A. Gianakon, D. C. Barnes, R. A. Nebel, S. E. Kruger, D. D. Schnack, S. J. Plimpton, A. Tarditi, M. S. Chu and the NIMROD Team, J. Comput. Phys. 195, 355 (2004).
[29] D. D. Schnack, D. C. Barnes, D. P. Brennan, C. C. Hegna, E. Held, C. C. Kim, S. E. Kruger, A. Y. Pankin and C. R. Sovinec, Phys. Plasmas 13, 058103 (2006).
[30] S. E. Kruger, D. D. Schnack and C. R. Sovinec, Phys. Plasmas 12, 056113 (2005).
[31] E. D. Held, J. D. Callen, C. C. Hegna, C. R. Sovinec, T. A. Gianakon and S. E. Kruger, Phys. Plasmas 11, 2419 (2004).
[32] C. R. Sovinec, T. A. Gianakon, E. D. Held, S. E. Kruger, D. D. Schnack and the NIMROD Team, Phys. Plasmas 10, 1727 (2003).
[33] A. H. Glasser, C. R. Sovinec, R. A. Nebel, T. A. Gianakon, S. J. Plimpton, M. S. Chu, D. D. Schnack and the NIMROD Team, Plasma Phys. Control. Fusion 41, A747 (1999).
[34] A. Y. Pankin, G. Bateman, D. P. Brennan, A. H. Kritz et al., Plasma. Phys. Control. Fusion 49, S63 (2007).
[35] C. R. Sovinec, D. C. Barnes, R. A. Bayliss et al., J. Phys.: Conf. Ser. 78, 012070 (2007).
[36] D. P. Brennan, S. E. Kruger, D. D. Schnack, C. R. Sovinec and A. Y. Pankin, J. Phys.: Conf. Ser. 46, 63 (2006).
[37] L. E. Sugiyama and W. Park, Phys. Plasmas 7, 4644 (2000).
[38] H. R. Strauss and W. Longcope, J. Comput. Phys. 147, 318 (1998).
[39] Portable extensible toolkit for scientific computation; `http://www-unix.mcs.anl.gov/petsc`.
[40] G. Y. Fu, W. Park, H. R. Strauss, J. Breslau, J. Chen, S. Jardin and L. Sugiyama, Phys. Plasmas 13, 052517 (2006).
[41] Equilibrium fitting software; `http://fusion.gat.com/theory/Efit`.
[42] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao and Y. Zhao, Concurrency and Computation: Practice & Experience 18(10), 1039 (August 2006).
[43] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs and Y. Xiong, Proceedings of the IEEE 91(1), 127 (January 2003).
[44] E. A. Lee and S. Neuendorffer, Actor-oriented models for codesign: Balancing re-use and performance, Formal methods and models for system design: a system level perspective, ISBN 1-4020-8051-4 (2004).
[45] G. Kahn and D. B. MacQueen, Proceedings of the IFIP 77, 993 (1977).
[46] N. Podhorszki, B. Ludǎscher and S. Klasky, "Workflow Automation for Processing Plasma Fusion Simulation Data", 2nd Workshop on Workflows in Support of Large-Scale Science (Monterey, CA, June 2007).
[47] R. I. Balay, M. A. Vouk, H. Perros, "Performance of Network-Based Problem-Solving Environments," Enabling Technologies for Computational Science Frameworks, Middleware and Environments, editors Elias N. Houstis, John R. Rice, Efstratios Gallopoulos, and Randall Bramley, ISBN 0-7923-7809-1 (2000).
[48] Web2 technologies; `http://en.wikipedia.org/wiki/Web_2`.
[49] B. McLaughlin, Head Rush Ajax (O'Reilly, March 2006), ISBN-10 0596102259.
[50] T. Perkins, Adobe Flash CS3 Professional Hands-On Training (Peachpit Press, September 2007), ISBN-10 0321509838.
[51] Ajax terminal; `http://antony.lesuisse.org/qweb/trac/wiki/AjaxTerm`.