ComPat Framework for Multiscale Simulations Applied to Fusion Plasmas

O. O. Luk,[1] O. Hoenen,[1] A. Bottino,[1] B. D. Scott,[1] and D. P. Coster[1, a)]

Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, D-85748, Garching, Germany

Abstract:

Understanding the dependency between plasma turbulence and overall transport is essential in predicting the performance of fusion devices. This dependency is not fully understood because of the highly disparate spatio-temporal scales involved, which prohibits a fully-resolved turbulence simulation at transport scales. The Computer Patterns for High Performance Multiscale Computing (ComPat) project takes the component based approach for multiscale simulations to connect existing single-scale models into a workflow. In this paper, we present the ComPat's approach in building a multiscale fusion application: it brings equilibrium, transport and turbulence models together, where the turbulence is described by a 3D gyrofluid code. Initial results and challenges encountered with such approach are also presented and discussed. One of the challenges is to ensure numerical stability, for which adaptive time step size and the search for quasi-steady state are implemented as possible solutions. Another challenge is to improve the overall performance of multiscale simulation, and that is addressed by increasing the level of parallelism to the workflow.

[a)]Email: {*onnie.luk*, *olivier.hoenen*, *bottino*, *bds*, *david.coster*}@ipp.mpg.de

I. INTRODUCTION

The dynamics of a thermonuclear fusion plasma is very complex. Such complexity arises from the instabilities driven by plasma turbulence, which in turn can destroy plasma confinement. Fully simulating the impact of turbulence on the performance of a fusion device such as ITER[1] is challenging, and doing so solely with a turbulence code is projected to exceed even the next generation of exascale computers. Therefore, a multiscale approach will become necessary, for fusion plasmas exhibit highly disparate spatio-temporal scales. For example, in the continuity equation:

$$\frac{\partial}{\partial t} n_\alpha + \nabla \cdot \Gamma_\alpha = S,$$

the particle density of species α (n_α), can be obtained from a plasma transport code; the particle flux of species α (Γ_α), is obtained from turbulence calculations. All quantities rely on the equilibrium state, including the prescribed particle source (S). This is a multiscale problem, for turbulence spans roughly microseconds in time and millimeters in space, while transport describing energy confinement spans in the order of few seconds in time and a few meters in space.

While there exist single-scale models to study turbulence (e.g. gyrokinetic models) and transport (large-scale simplified models) separately, efforts are made in building multiscale models to study fusion plasmas in the past.[2–5] Some of the more recent efforts[6,7] include integrated modeling framework that takes users' feedback and inputs to improve and broaden the multiscale physics model[8]; transport solver that takes steady state turbulent fluxes from a turbulence code[9]; and, transport manager that runs turbulent transport and neoclassical codes in parallel[10]. In particular, the European Integrated Tokamak[11] Modelling Task Force (EFDA ITM-TF) constructed a generic platform for both tokamak modeling and framework to build complex workflows[12]. These workflows use the component based approach and are established on a standardized interface. A significant advantage to this approach is that it allows simple replacement of any individual component with another that has the same interface, therefore increases extensibility and eases the benchmarks and verification efforts. Currently, this platform is sustained through the EUROfusion consortium[13] and contributes extensively to the Integrated Modelling and Analysis Suite[14], which is developed specifi-

cally for ITER. In the MAPPER project[15], the same standardized interface were used with different framework (Multiscale Modelling and Simulation Framework, or MMSF) and technologies (Multiscale Coupling Library and Environment, or MUSCLE2[16]) for applications that require distributed or high-performance computing (HPC) resources[17,18]. This parallel effort is continued within the ComPat project[19], in which the MMSF is extended with the concept of Multiscale Computing Patterns[20], or MCP. The project aims to run applications more efficiently, especially when one or several single-scale models require petascale computing resources.

The developed MMSF and the available technologies within the ComPat project can be extended to build a multiscale workflow for the fusion application. In this paper, we apply such workflow to study multiscale fusion plasmas in a tokamak scenario. We also introduce and implement methods that improve numerical stability and increase overall runtime performance to the workflow. The layout of the paper is as follows: specifics on the ComPat project and the implementation of the fusion application using such framework are discussed in Section II and III, respectively. Challenges encountered in this simulation framework and solutions we enact, including alleviate time-scale mismatch between small-scale turbulence and large-scale plasma transport, terminate simulation at appropriate time (e.g. quasi-steady state), and increase level of parallelism to the workflow, are detailed in Section IV. Finally, we conclude with summary and future plans in Section V.

II. COMPAT PROJECT

Nowadays there is a growing demand for heavy HPC resources in multiscale applications. Traditionally, the monolithic approach is taken to construct a multiscale application, in which all scales of interests are included within one single code (Fig. 1, left). However, the MMSF takes the component based approach, in which a group of single-scale models (or *submodels*, with each simulating a shorter range of scales) are coupled together to form a multiscale simulation (Fig. 1, right). In such scale separation process, the submodels can have either partial or complete overlap in time, space or both. It is an interesting approach from the software engineering point of view, for it leads to a simpler algorithm and code base for each submodel. In return, this allows for faster and better-optimized implementations that are easy to debug and maintain. However, despite the benefits that appear at the

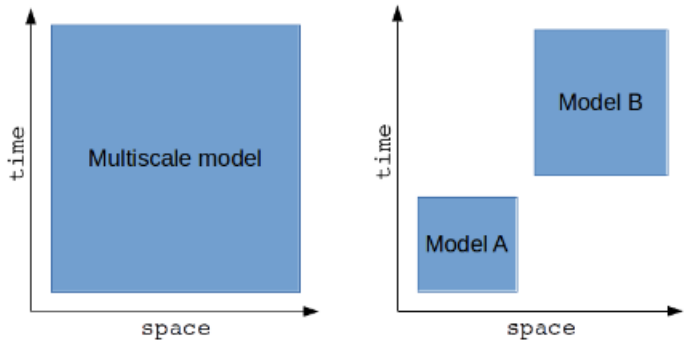

FIG. 1. Comparison between the range of space and time covered by the traditional, monolithic multiscale model (left) and the component based, multiscale model (right).

submodel level, this approach implies added complexity in binding all submodels within a coupled multiscale application. This in turn can generate runtime overheads. To alleviate the efforts required from the application developers, many generic and domain specific projects are developing coupling libraries, frameworks and execution platforms. An overview is given by Groen et. al.[21]

The ComPat project represents one such effort: its purpose is to develop domain-agnostic solutions for multiscale simulations. Its first major objective is to provide a collection of methods and software that can ease the development of component based, multiscale simulations. The other major objective is to create generic transformation and optimization methods (e.g smart scheduling and load-balancing of the components) that can improve the simulations' performance on a targeted set of execution platforms, despite the coupling overheads. A simulation's performance can be measured by metric such as runtime, time to completion, efficiency, and energy consumption. The underlying idea is to first estimate the computing requirement for each submodel, and then to understand how and when the submodels interact with each other. With these information, the multiscale application can be categorized in terms of one of the three MCPs shown in Fig. 2, for each pattern has its own transformations and optimization techniques. In ComPat, multiscale applications from several domains (e.g. thermonuclear fusion, astrophysics, material science, biomedicine, and social science) are mapped to the most suitable patterns to demonstrate how generic these MCPs are. In this paper, the fusion application we constructed (Section III) falls into the Extreme Scaling MCP, in which a primary submodel (computationally intensive) is coupled

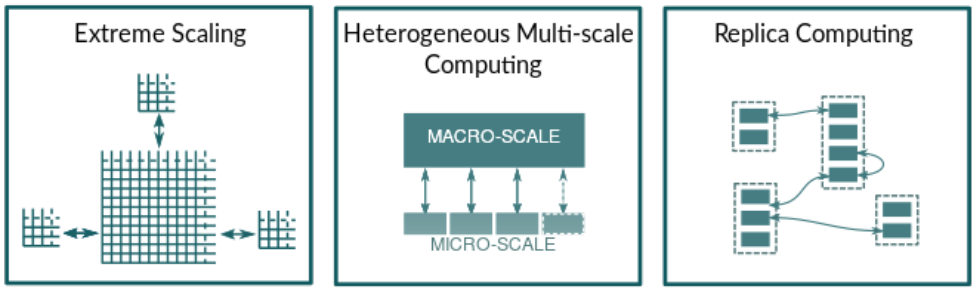

FIG. 2. Graphic illustrations of the three Multiscale Computing Patterns: Extreme Scaling (left), Heterogeneous Multiscale Computing (center), and Replica Computing (right) patterns.

to several auxiliary submodels (computationally inexpensive).

III. IMPLEMENTATION OF THE COUPLED MULTISCALE APPLICATION

A. A generic and non-intrusive implementation

In order to minimize the intrusiveness of this approach to the existing single scale models, each one of these models has to go through several layers of transformations (called *wrappers*) to become a component of the multiscale application, as illustrated in Fig. 3. First, a *data wrapper* is implemented around a physics routine (can be a legacy code) to ensure both inputs and outputs are using a common interface (called consistent physical objects CPOs). This layer is independent to the component-coupling and execution environment, therefore the data wrappers can be reused in different applications using different technologies. Around this data wrapper, a *component* layer is implemented for a given coupling environment. In our present fusion application, we use MUSCLE2 as the coupling environment. The instantiation of a component (or *kernel* in MUSCLE2 terminology) requires the calling of MUSCLE2 API to perform different generic steps. These steps include initializing kernel execution, getting parameters' values, implementing the parameterized time loop (internal to each kernel), receiving and sending data, and finalizing the execution of the kernel. The MUSCLE2 is available for C++/Fortran/Java/Python/Matlab codes and its API is very similar to MPI's, so the learning curve is not very steep for HPC-experienced developers. Each kernel is compiled into a single executable, and it runs through the MUSCLE2

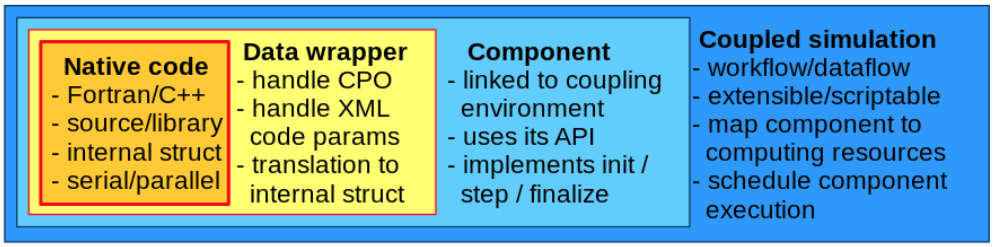

FIG. 3. The layers of wrappers apply around the native code to create a non-intrusive, component based approach.

execution environment. The MUSCLE2 requires an input configuration file that describes the *coupled simulation*. This file is written in the Ruby programming language, and contains the list, types and path to executable of all kernels present in this application, some of kernels' parameters (global or local), and the coupling topology (i.e. how kernels are linked through their respective inputs/outputs). Such description is easy to extend, modify, and parameterized. For example, kernels with the same interface can be interchanged by updating a single line of file. This makes the implementation of toy submodels and the benchmarking process (with different implementations) during the application development phase very easy and straightforward. In our current application, which is described in detail in Section III B, the components are tightly-coupled due to frequent data exchange and feedback loops. The tightly-coupled simulation requires memory-based or network-based data exchange, and MUSCLE2 handles such exchange over TCP/IP. However, data exchange using MPI-like communications is also possible.

B. Details of the fusion application

The multiscale application we propose in this work is composed of four major submodels:

- TRA: a 1D transport model that evolves temperature and other profiles in time
- EQU: a 2D fixed boundary equilibrium model that updates geometrical information
- TUR: a 3D gyrofluid turbulence model that computes heat and particle fluxes
- FDV: a module that converts fluxes coming from turbulence submodel into transport coefficients compatible with transport submodels

Native codes for the targeted submodels have been developed independently; their respective data wrappers are available through the ITM community in Europe, and the common data structures are defined by Imbeaux et. al.[22]. In the current implementation, the TRA submodel is described by the ETS module,[23] which solves for equation describing poloidal flux, density equation for every ion species, temperature equations for both ions and electrons, and the toroidal rotation equation for ions. It has temperature fixed at the boundary. The EQU is described by CHEASE[24], which is a high-resolution, fixed-boundary Grad-Shafranov solver. The TUR is described by GEM[25], which is a gyrofluid electromagnetic flux-tube model that determines heat and particle fluxes at each flux-tube in a field-aligned shifted-metric coordinate system. It takes straight fieldline coordinate metric and builds a field-aligned shifted-metric coordinate system. The numerical grids are based on field-aligned coordinate system (x,y,s): one axis (s) runs in parallel to the magnetic field, one (x) points radially across toroidal flux surfaces, and one (y) has vanishing projections to both equilibrium magnetic field and background gradient. We use fixed number of grids throughout the entire simulation, with 128 x 128 x 32 grids in x- ,y-, s-direction, respectively. Dirichlet boundary conditions are applied to the x-direction.[26] Initial fluctuations are launched as single Maxwellian density structure localized at nonlinear amplitude with Gaussian profiles in drift plane and along field lines. Finite electron pressure launches shear-Alfvén waves and then drift wave field at nonlinear amplitude. The system proceeds to full-turbulence unless nonlinearly stable. The calculations at each flux-tube are done concurrently. The FDV is described by a module based on dynamical alignment[26]. As mentioned in Section III A, a data wrapper surrounds the submodels and a MUSCLE2 kernel is implemented around each of these data wrappers to form components in the coupled multiscale application. A few additional components are required and are depicted in the coupling topology presented in Fig. 4: the INI component imports initial data into the system from various sources (files, databases, etc.) and generic duplication data mappers dup (a part of the MUSCLE2 standard library) scatter data to several destinations.

Fig. 4 also shows the communication flow of CPOs among the components. The arrows in blue show the flow of equilibrium CPOs: the EQU outputs the updated geometrical information (in the form of CPOs), then MUSCLE2 duplicates and distributes the CPOs to TUR, FDV, and TRA as inputs. The arrows in red show the flow of core plasma profiles CPOs: the TRA outputs the updated profiles such as electron temperature T_e, then MUS-

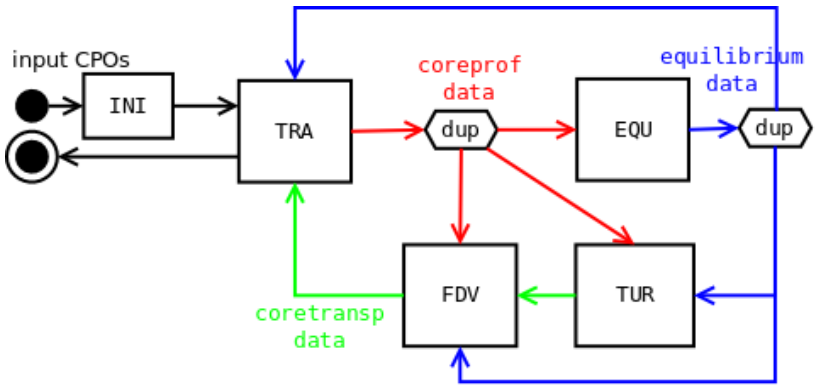

FIG. 4. Topology of the fusion application. Each box represents a component of the application, and colored arrows represent the data flow within the application.

CLE2 duplicates and distributes the CPOs to EQU, TUR, and FDV as inputs. The arrows in green show the flow of core plasma transport coefficients as CPOs: the TUR outputs updated transport coefficients such as electron heat flux q_e, then MUSCLE2 duplicates and distributes the CPOs to FDV as inputs. The FDV then converts fluxes from TUR to coefficients that are required by the transport model, and passes these coefficients to TRA as inputs.

There are computing challenges arising from this workflow that need to be addressed. One issue is that only the TUR submodel is described by a parallel code, and it needs more computing resources than those required by other submodels. Hence, the application fits into the Extreme Scaling MCP. Another issue with our workflow topology is that the data dependencies exhibited in this application prevent concurrent execution of submodels: the TUR submodel needs to wait for updated data from other serial submodels, which decreases the overall parallel efficiency to the application. In the next section, we discuss the challenges arising from the workflow topology and the methods we use in attempt to resolve these challenges.

IV. CHALLENGES IN MULTISCALE, COMPONENT BASED APPROACH

There are two types of challenges in building a multiscale fusion simulation using the component based approach: physics ones and computation ones. Some of these physics challenges include conversion of fluxes into diffusivity and convection coefficients (D and v,

respectively). However, we will not elaborate the physics challenges any further. Instead, we will focus on several computational challenges in this paper. This includes time bridging between turbulence and transport models of disparate scales, finding quasi-steady state, and optimizing the overall performance of the workflow.

A. Time Bridging Technique

Using the workflow we constructed (see Fig. 4) and initial parameters based on an ASDEX Upgrade tokamak-like scenario, a simulation was performed using 1024 cores with 8 flux-tubes in GEM, while particle density is fixed in time. The results are plotted in Fig. 5. In this figure, T_e, which is calculated by ETS, experiences large fluctuations in time. The T_e fluctuations do not allow for the plasma to converge to a steady state value. This is especially apparent for flux-tubes closer to the core of the plasma. On the other hand, q_e, directly calculated by GEM, does not change much until T_e drops significantly. Whenever T_e drops significantly, q_e rises and drops in a short period of time, forming bursts on q_e. The time evolution of T_e displayed in Fig. 5 is not physical, but instead, it is artificially produced from the time-scale mismatch between turbulence and transport codes.

Every time we call GEM within the workflow, it runs 5000 internal time steps. Each one of these steps is 0.002 $L_\perp/c_s$, in which $L_\perp$ is the background profile length scale and c_s is the speed of sound. The vorticity scale is approximately $c_s/L_\perp$. Therefore, 5000 GEM internal time steps equate to 10.0 $L_\perp/c_s$. A call to the ETS comes after GEM, and each ETS call has time step size Δt (a predefined parameter) fixed at 0.01 s. As a result, the q_e cannot correctly adapt to the T_e profile, especially when the profile evolves too quickly within one ETS time step.

We performed another simulation with Δt set to 0.001 s, with the time evolution of q_e and T_e shown in Fig. 6. The figure shows that, with the time step size lowered by a factor of 10, the time-scale mismatch between turbulence and transport calculations diminishes. Therefore, we need the transport code to evolve the profiles at a lower rate compared to the turbulence code, such that the turbulence code can adapt to the changes from the profile.

To carry such idea further, a time-bridging technique is introduced to alleviate the time-scale mismatch in the current workflow. Instead of having a fixed Δt, it becomes adaptive and dependent on two parameters: $\Delta T_{e,lim}$ and $\Delta \partial_\rho T_{e,lim}$. The $\Delta T_{e,lim}$ is an upper limit to

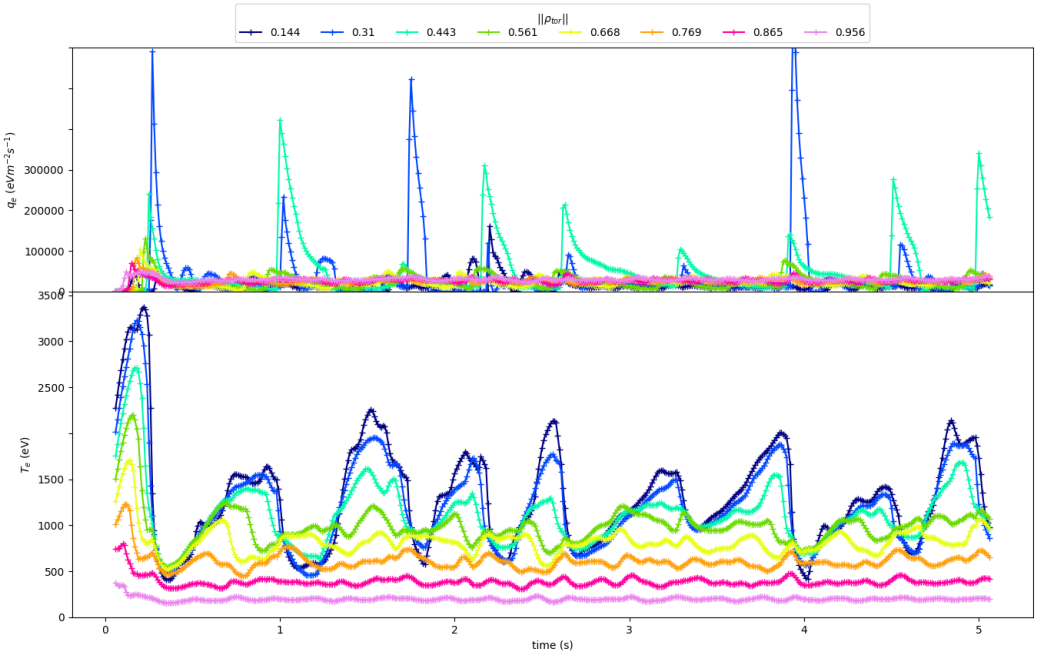

FIG. 5. Time history of electron heat flux q_e (top panel, in the unit of $\mathrm{eVm^{-3}s^{-1}}$) and electron temperature T_e (bottom panel, in the unit of eV) at eight flux-tube locations. Colored traces represent flux-tube locations in the normalized toroidal flux coordinate value $||\rho_{tor}||$. The ETS time step size Δt is 0.01 s.

the change on T_e at time t ($\Delta T_e(t)$), which is expressed as

$$\Delta T_{e,lim} > \Delta T_e(t) = \left| \frac{T_e(t-\Delta t) - T_e(t)}{T_e(t-\Delta t)} \right|. \tag{1}$$

The other parameter, $\Delta\partial_\rho T_{e,lim}$, is an upper limit to the change on radial gradient of T_e at time t ($\Delta\partial_\rho T_e(t)$), which is expressed as

$$\Delta\partial_\rho T_{e,lim} > \Delta\partial_\rho T_e(t) = \left| \frac{\partial_\rho T_e(t-\Delta t) - \partial_\rho T_e(t)}{|\partial_\rho T_e(t-\Delta t)| + \frac{T_e(t-\Delta t)}{a}} \right|. \tag{2}$$

Here, $\partial_\rho T_e$ is the derivative of T_e with respect to the toroidal flux coordinate ρ and a is the maximum toroidal flux coordinate value. The reason we set Δt to be dependent on $\Delta T_e(t)$ and $\Delta\partial_\rho T_e(t)$ is to make sure the temperature profile is evolving slowly enough, such

FIG. 6. Time history of electron heat flux q_e (top panel, in the unit of $\mathrm{eVm^{-3}s^{-1}}$) and electron temperature T_e (bottom panel, in the unit of eV) at eight flux-tube locations. Colored traces represent flux-tube locations in the normalized toroidal flux coordinate value $||\rho_{tor}||$. The ETS time step size Δt is 0.001 s.

that the turbulence code can adapt to the changes from the profile. If Δt is too large and therefore does not allow the deviations $\Delta T_e(t)$ and $\Delta\partial_\rho T_e(t)$ to go below the limits set by the user, then the algorithm would use a smaller Δt and try again, until the deviations satisfy the corresponding limits. However, if the limits are not satisfied after several iterations, namely when Δt has become smaller than the turbulence time scale, then the simulation would terminate. This is a process implemented within the data wrapper around ETS to ensure the temperature profiles calculated by the transport submodel evolve at a pace more adaptable for the turbulence submodel.

A simulation using the described time bridging technique has been tested. The time history of the q_e and T_e at various flux-tube locations, along with time history of time step size is shown in Fig. 7, with $a = 0.695\mathrm{m}$. The deviation limits are fixed at $\Delta T_{e,lim} = 0.2$ and $\Delta\partial_\rho T_{e,lim} = 0.1$.

In comparison to Fig. 5, where Δt is fixed at 0.01 s, T_e does not have large-scale fluc-

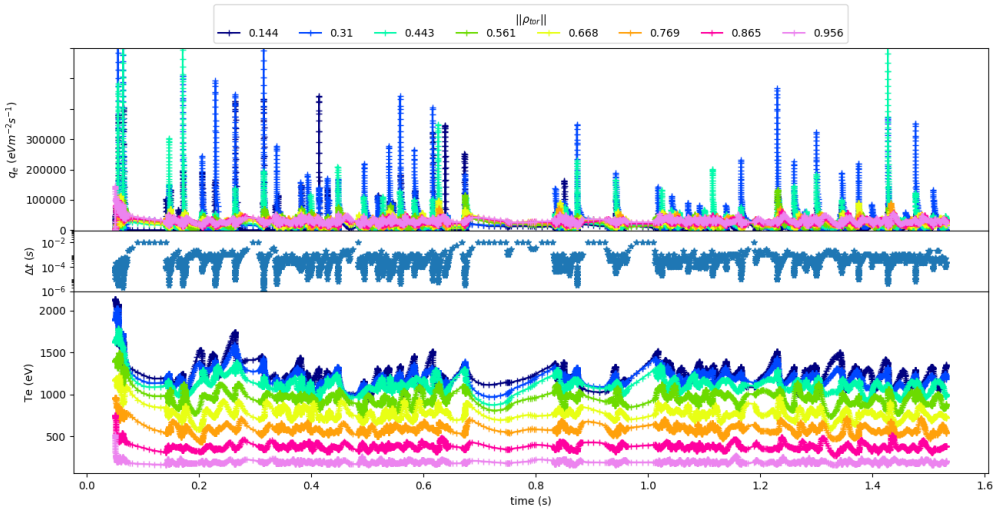

FIG. 7. The time history of q_e (top panel, in the unit of $\mathrm{eVm}^{-3}\mathrm{s}^{-1}$) and T_e (bottom panel, in the unit of eV) of each flux-tube, along with Δt (middle panel, in the unit of s). Every ETS time step ranges from 1e-8 to 1e-2 s (see middle panel).

tuations, even though q_e continues to have bursts in time, especially with the inner most flux-tubes. The time history of T_e indicates that it is converging to a steady-state value.

B. Determine Quasi-Steady State

The turbulence code uses the majority of the computing resources allocated to the multiscale simulation, which makes it the most costly component to run in terms of both wall clock time and number of cores, when compared to the equilibrium, transport, and the FDV module calculations. For example, when we run one iteration of the workflow (Fig. 4) with 1024 cores on the Marconi Tier 0 supercomputer from Cineca, with Intel Xeon 8160 (Skylake) processors, GEM takes all 1024 cores and runs for approximately 27.0 s, while ETS and CHEASE take one core each and run for 0.8 and 1.9 s, respectively. If we were to replace GEM with a more precise but even more costly gyrokinetic turbulence model, then the computing resources consumed by a single iteration would increase and the timescale covered by the model would decrease, so the workflow would require more iterations before reaching saturation. Therefore, developing a method that can detect the appropriate time to terminate the simulation is desirable in order to spare computing resources.

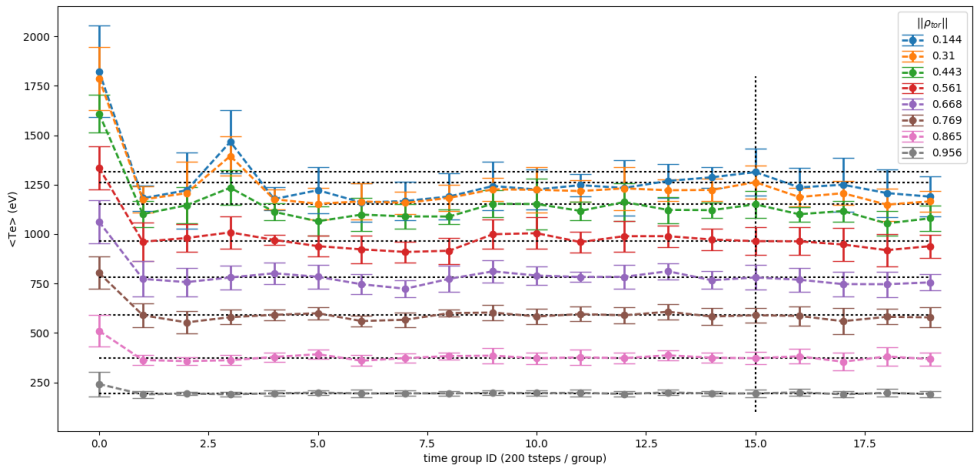

FIG. 8. The time history (time steps divided into groups) of averaged electron temperature $< T_e >$. Horizontal dotted lines mark the $< T_e >$ value of every flux-tube and vertical dotted line marks the time at which quasi-steady state is reached. The colored dash lines represent the $< T_e >$ of every flux-tube, with respect to time.

The T_e for each flux-tube saturates over time as electrons reach quasi-steady state. Hence, the following method is established. We place every N time steps of T_e calculations into a time group. First, we determine for every flux-tube the average of T_e within the n^{th} time group $< T_e >_n$ and its standard deviation σ_n. Then $< T_e >_n$ and σ_n are compared to $< T_e >$ of the previous G time groups, in other words, from $< T_e >_{n-G}$ to $< T_e >_{n-1}$. If those values fall within $< T_e >_n \pm \sigma_n$ for every flux-tube, then quasi-steady state is achieved. The simulation mentioned in section IV A (see Fig. 5) is utilized for the quasi-steady state tests: $N = 10$, 20, 50, 100, and 200 steps/group were tested with G set to 5. While the first four scenarios show that quasi-steady state for all flux-tubes cannot be reached within 4000 time steps, $N = 200$ shows that saturation is achieved, and Fig. 8 shows the $< T_e >$ in time groups for every flux-tube. This time-averaging approach shows that quasi-steady state in the core plasma is reached at $n = 15$, or at $t = 1.235$ s.

To further test the reliability of such approach, the same simulation is extended for another 2000 time steps. Then the time-average of T_e is performed on every 200 steps, as described earlier. There are a total of 14 time groups after the established quasi-steady state (with time group ID of $n = 15$). The $< T_e >_n$ of these later time groups ($16 \leq n \leq 29$)

are being compared to $< T_e >_n \pm\sigma_n$ at $n = 15$. The results show that the outer flux-tubes (flux-tube #3-7, or at normalized toroidal flux coordinate value $||\rho_{tor}|| = 0.561-0.956$) agree with the determined quasi-steady state. Specifically, the $< T_e >_n$ at these outer flux-tubes remain within one standard deviation of the quasi-steady state temperature ($< T_e >_n \pm\sigma_n$ at $n = 15$). However, the values of $< T_e >_n$ (with $16 \leq n \leq 29$) at the inner flux-tubes (flux-tube #0-2, or $||\rho_{tor}|| = 0.144-0.443$) do not always fall within $< T_e >_n \pm\sigma_n$ at $n = 15$. The statistics on the number of occurrences of $< T_e >_n$ (with $(16 \leq n \leq 29)$ at the inner flux-tubes with value that falls within $< T_e >_{15} \pm x\sigma_{15}$ are shown in Table I, with x being a constant above one. Overall, for $< T_e >_n$ with $16 \leq n \leq 29$, any deviation above $\sigma_{n=15}$ happens only at flux-tubes closer to the plasma core. However, these deviations never reach above $1.4\sigma_{n=15}$.

	x				
flux-tube #	$1.01-1.10$	$1.11-1.20$	$1.21-1.30$	$1.31-1.40$	> 1.40
0	1	0	0	0	0
1	1	2	0	1	0
2	2	0	2	1	0

TABLE I. Number of occurrences in which $< T_e >_n$ (with $16 \leq n \leq 29$) at an inner flux-tube falls within $< T_e >_n \pm x\sigma_{n=15}$, where $n = 15$ is the time bin ID # in which quasi-steady state is established.

C. Optimization: improving the level of parallelism

A quick analysis of the workflow presented in Fig. 4 shows that only one component (MUSCLE kernel) can run at a given time while the others would need to wait due to their input data dependencies. Each kernel contains three sequential steps: *blocking receive – run – send*; this means in practice, the time spent in *receive* in each iteration is equal to the sum of *run* time for all other kernels (if we consider high-performance interconnect and the targeted jobs size, communication costs are negligible for such data transfers). Since the TRA, EQU and FDV models are implemented by serial codes, the idle time spent in *receive* for the parallel implementation of the TUR component is $receive(TUR) \approx$

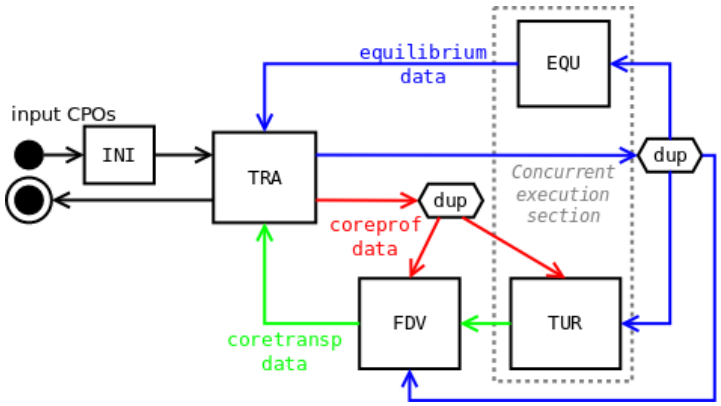

FIG. 9. Parallel (new) workflow: the equilibrium EQU and turbulence TUR codes can run at the same time.

$run(TRA) + run(EQU) + run(FDV)$, which reduces the parallel efficiency of the coupled application.

A closer look into the submodels gives us two important information: the *run* time of EQU is dominant compared to TRA and FDV ($run(EQU) > run(TRA)+run(FDV)$), and EQU has no specified physical time scales. From this observation we decided to modify the logical order of the components of the workflow to obtain the new topology described in Fig. 9. As highlighted in this schema, the TUR and EQU kernels can run at the same time, as soon as they have received output data from TRA. Assuming that $run(EQU) <= run(TUR)$ (which is a valid assumption in most use cases), the waiting time lost in TUR is reduced to $receive(TUR) = run(TRA) + run(FDV)$.

With the same parameters presented in Section IV A, a simulation using the new workflow configuration is conducted with 1024 cores on Marconi supercomputer. The simulation completes 4000 iterations in 31 h, 40 min and 37 s, which is almost two hours less than the simulation using the initial workflow configuration (5.7% faster). As expected, the measurement of *receive* step for TUR shows where most of the improvement lies: from 2.8 s at each iteration in the initial workflow, to 0.9 s with the new workflow. It is also important to note that such performance improvement was obtained without modifying the code base for any of the kernels, but solely through the MUSCLE2 configuration file (thus no re-compilation was necessary). This illustrates how flexible this approach can be for modifying and testing several workflow configurations.

However, this new configuration also means that the heat and particle fluxes calculated

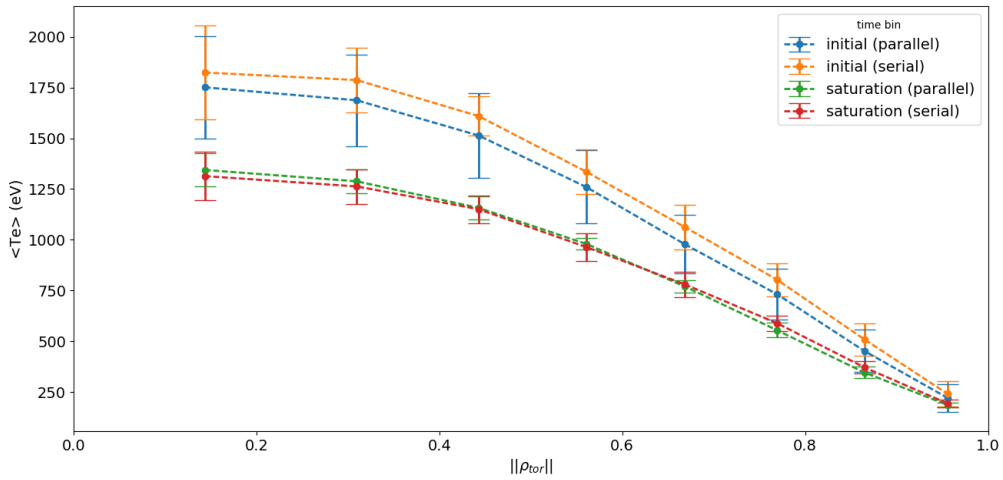

FIG. 10. Comparison between serial and parallel workflows at both initial and saturation time bins. Time-averaged electron temperature $\langle T_e \rangle$ vs. normalized toroidal flux coordinate value $||\rho_{tor}||$

by the turbulence code, in addition to the transport coefficients calculated by the numerical module FDV, would be based on the geometry property computed by the equilibrium code at the previous time step. Since EQU is scaleless in time, such modification is expected to have minimal impact on the simulation results. To validate such statement, we performed the quasi-steady state test (see Section IV B) on the new workflow and then compared the results to the original workflow shown in Fig. 8. Both workflows set N to 200 and G to 5. The quasi-steady state test shows that the initial workflow saturates at group ID #15, or 1.235 s. On the other hand, the new workflow saturates at group ID #16, or 1.201 s. The T_e profiles from the original (serial) and the new (parallel) workflows at initial time group (group ID #0) and saturation time groups are plotted in Fig. 10. At saturation time, as shown in the figure, the T_e profile of the parallel workflow stays within one standard deviation from the serial workflow's T_e profile. This indicates that both workflow configurations yield agreeable results.

V. CONCLUSION AND FUTURE PLANS

To understand the entirety of the problem on how turbulence affects the overall performance of fusion devices such as ITER, a multiscale approach is inevitable. However, building

a multiscale simulation to study fusion plasmas is not so simple because of the disparate time scales involved. The ComPat project offers a framework that helps tackle multiscale problems through a component based approach; this approach connects existing single-scale models into a multiscale workflow. Besides re-using existing single-scale models (CHEASE, ETS, GEM), standardized interfaces (CPOs) and a flexible and extensible coupling library (MUSCLE2), we introduced a time-bridging technique to alleviate the time-scale mismatch between small-scale turbulence and large-scale plasma transport. We defined upper limits on the evolution of electron temperature and radial gradient of electron temperature at each time step. An adaptive time step mechanism was introduced to ensure that such limits can be met. This allows for the short time-scale physics to be correctly resolved in a multiscale simulation involving widely disparate time scales. Determining quasi-steady state of the core plasma for the purpose of stopping simulation at appropriate time is addressed. Specifically, a time-averaging method is developed to find the time-averaged electron temperature of a time group at every flux-tube. The values are then compared to the ones from previous time groups to determine whether quasi-steady state is reached. This heuristic for defining a quasi-steady state has shown promising results, and needs to be confirmed with a more computationally-intensive Gyrokinetic submodel for the turbulence simulation. The method we presented in this paper is one of many ways to find the quasi-steady state quickly. In comparison to one method that takes eddy-turn-over time into consideration[27], our approach aims to ensure the electron temperature at each flux-tube reaches saturation. For example, even though the outer flux-tubes in our simulations reach quasi-steady state before the inner flux-tubes, we chose to terminate the simulation when all flux-tubes reach saturation. Finally, we showed that a workflow's level of parallelism can be improved easily thanks to the effortless topology change in the MUSCLE2 configuration file. We introduced parallelization at the workflow level, with equilibrium and turbulence codes running concurrently. Simulations using both serial and parallel workflows were compared, and the parallel workflow showed an overall gain of 5.7% in runtime, as well as agreeable electron temperature profiles with the serial workflow at quasi-steady state.

In the future, physics challenges mentioned in this paper will be explored. The 3D gyrofluid model (e.g. GEM) will be replaced with more realistic, 5D gyrokinetic codes (e.g. dFEFI and ORB5[28]). Since ORB5 is a global turbulence code, the physics challenge on how to update global profiles and calculate transport coefficients in turbulence code

will naturally be addressed. In addition, parameters implemented in the time-bridging technique will become time-dependent. This would require further understanding on heat flux and temperature, and how they vary based on size of the time step. As for determining quasi steady state, we will test all the mentioned parameters with different values, and try to understand the quasi-steady state for each flux-tube. Other methods on determining quasi-steady state may also be explored in the future. Lastly, the overall performance will be further investigated with respect to available resources allocated to the simulation. Additional techniques such as the use of additional cores on each node may also improve multiscale simulation performance.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme for the ComPat project, under grant agreement No 671564. This project is part of the FET-Future Emerging Technologies funding schema.

REFERENCES

[1] `http://www.iter.org`.

[2] L. LoDestro, B. Cohen, and R. Cohen, "Comparison of simulations and theory of low-frequency plasma turbulence," in *Plasma physics and controlled nuclear fusion research 1990. V. 2* (1991).

[3] A. Shestakov, R. Cohen, J. Crotinger, L. LoDestro, A. Tarditi, and X. Xu, "Self-consistent modeling of turbulence and transport," Journal of Computational Physics **185**, 399–426 (2003).

[4] Y. Nishimura, D. Coster, and B. Scott, "Characterization of electrostatic turbulent fluxes in tokamak edge plasmas," Physics of Plasmas **11**, 115–124 (2004).

[5] D. Reiser and B. Scott, "Electromagnetic fluid drift turbulence in static ergodic magnetic fields," Physics of plasmas **12**, 122308 (2005).

[6] E. Highcock, N. Mandell, M. Barnes, and W. Dorland, "Optimisation of confinement in a fusion reactor using a nonlinear turbulence model," Journal of Plasma Physics **84** (2018).

[7]J. B. Parker, L. L. LoDestro, D. Told, G. Merlo, L. F. Ricketson, A. Campos, F. Jenko, and J. A. Hittinger, "Bringing global gyrokinetic turbulence simulations to the transport timescale using a multiscale approach," Nuclear Fusion **58**, 054004 (2018).

[8]O. Meneghini, S. Smith, L. Lao, O. Izacard, Q. Ren, J. Park, J. Candy, Z. Wang, C. Luna, V. Izzo, *et al.*, "Integrated modeling applications for tokamak experiments with omfit," Nuclear Fusion **55**, 083008 (2015).

[9]M. Barnes, I. Abel, W. Dorland, T. Görler, G. Hammett, and F. Jenko, "Direct multiscale coupling of a transport code to gyrokinetic turbulence codes a," Physics of Plasmas **17**, 056109 (2010).

[10]J. Candy, C. Holland, R. Waltz, M. R. Fahey, and E. Belli, "Tokamak profile prediction using direct gyrokinetic and neoclassical simulation," Physics of Plasmas **16**, 060704 (2009).

[11]A. Becoulet, P. Strand, H. Wilson, M. Romanelli, L.-G. Eriksson, *et al.*, "The way towards thermonuclear fusion simulators," Computer physics communications **177**, 55–59 (2007).

[12]G. L. Falchetto, D. Coster, R. Coelho, B. Scott, L. Figini, D. Kalupin, E. Nardon, S. Nowak, L. L. Alves, J.-F. Artaud, *et al.*, "The european integrated tokamak modelling (itm) effort: achievements and first physics results," Nuclear Fusion **54**, 043018 (2014).

[13]`https://www.euro-fusion.org/`.

[14]F. Imbeaux, S. Pinches, J. Lister, Y. Buravand, T. Casper, B. Duval, B. Guillerminet, M. Hosokawa, W. Houlberg, P. Huynh, S. Kim, G. Manduchi, M. Owsiak, B. Palak, M. Plociennik, G. Rouault, O. Sauter, and P. Strand, "Design and first applications of the iter integrated modelling and analysis suite," Nuclear Fusion **55**, 123006 (2015).

[15]`https://www.mapper-project.eu`.

[16]`https://github.com/psnc-apps/muscle2`.

[17]O. Hoenen, L. Fazendeiro, B. D. Scott, J. Borgdoff, A. G. Hoekstra, P. Strand, and D. P. Coster, "Designing and running turbulence transport simulations using a distributed multiscale computing approach," in *40th EPS Conference on Plasma Physics, EPS 2013; Espoo; Finland; 1 July 2013 through 5 July 2013*, Vol. 2 (2013) pp. 1094–1097.

[18]J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. B. Belgacem, B. Chopard, D. Groen, P. V. Coveney, and A. G. Hoekstra, "Distributed multiscale computing with muscle 2, the multiscale coupling library and environment," Journal of Computational Science **5**, 719–731 (2014).

[19]http://compat-project.eu.

[20]S. Alowayyed, D. Groen, P. V. Coveney, and A. G. Hoekstra, "Multiscale computing in the exascale era," Journal of Computational Science (2017).

[21]D. Groen, S. J. Zasada, and P. V. Coveney, "Survey of multiscale and multiphysics applications and communities," Computing in Science & Engineering **16**, 34–43 (2014).

[22]F. Imbeaux, J. Lister, G. Huysmans, W. Zwingmann, M. Airaj, L. Appel, V. Basiuk, D. Coster, L.-G. Eriksson, B. Guillerminet, *et al.*, "A generic data structure for integrated modelling of tokamak physics and subsystems," Computer Physics Communications **181**, 987–998 (2010).

[23]D. P. Coster, V. Basiuk, G. Pereverzev, D. Kalupin, R. Zagorksi, R. Stankiewicz, P. Huynh, F. Imbeaux, *et al.*, "The european transport solver," IEEE Transactions on Plasma Science **38**, 2085–2092 (2010).

[24]H. Lütjens, A. Bondeson, and O. Sauter, "The chease code for toroidal mhd equilibria," Computer physics communications **97**, 219–260 (1996).

[25]B. D. Scott, "Free-energy conservation in local gyrofluid models," Physics of Plasmas **12**, 102307 (2005).

[26]B. D. Scott, "Dynamical alignment in three species tokamak edge turbulence," Physics of plasmas **12**, 082305 (2005).

[27]C. Holland, L. Schmitz, T. Rhodes, W. Peebles, J. Hillesheim, G. Wang, L. Zeng, E. Doyle, S. Smith, R. Prater, *et al.*, "Advances in validating gyrokinetic turbulence models against l-and h-mode plasmas," Physics of Plasmas **18**, 056113 (2011).

[28]A. Bottino, B. Scott, S. Brunner, B. F. McMillan, T. M. Tran, T. Vernay, L. Villard, S. Jolliet, R. Hatzky, and A. G. Peeters, "Global nonlinear electromagnetic simulations of tokamak turbulence," IEEE Transactions on Plasma Science **38**, 2129–2135 (2010).