Tokamak-agnostic actuator management for multi-task integrated control with application to TCV and ITER

N.M.Trang VU[1]*, T.C. BLANKEN[2], F. FELICI[1], C. GALPERTI[1], M. KONG[1], E. MALJAARS[2], O.SAUTER[1], the TCV team[3] and the EUROfusion MST1 team[4]

[1] *École Polytechnique Fédérale de Lausanne (EPFL), Swiss Plasma Center (SPC), CH-1015 Lausanne, Switzerland*

[2] *Eindhoven University of Technology, Department of Mechanical Engineering, Control Systems Technology Group, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

[3] *See the author list of S. Coda et al., 2017 Nucl. Fusion 57 102011*

[4] *See the author list of H. Meyer et al., 2017 Nucl. Fusion 57 102014*

Abstract

The plasma control system (PCS) of a long-pulse tokamak must be able to handle multiple control tasks simultaneously, and must be capable of robust event handling with a limited set of actuators. For ITER, this is particularly challenging given the large number of actuator-conflicting control requirements. To deal with these issues, this work develops a task-based approach, where a plasma supervisory controller and an actuator manager make high-level decisions on how to handle the considered control tasks, using generic actuator resources and controllers. This simplifies the interface for operators and physicists since the generic control tasks (instead of controllers) can be directly defined from the general physics goals. This approach also allows one to decompose the PCS into a *tokamak-dependent layer* and a *tokamak-agnostic layer.* The developed scheme is first implemented and tested on TCV for simultaneous β control, neoclassical tearing mode (NTM) control, central co-current drive, and H-mode control tasks. It is then applied to an ITER test scenario to prove its flexibility and applicability to systematically handle a large number of tasks and actuators.

Keywords: plasma control system, actuator management, plasma integrated control

1. Introduction

In order to fulfil the global physics goals in complex experiments on long-pulse tokamaks, a plasma control system (PCS) should aim to simultaneously reach multiple control objectives (control tasks) using a limited set of actuators [1]. It must solve the possibly conflicting requests of multiple control tasks and actuator sharing issues during normal operations. At the same time, undesired plasma events or hardware failures can suddenly occur and deteriorate the plasma as well as possibly damage the plasma facing components. For example a neoclassical tearing mode (NTM) that evolves into a locked mode can lead to a disruption, thus the PCS has to predict/detect these events and adapt its control solution to the new situation.

A combination of a supervisory controller and an actuator manager have been proposed as a solution in the PCS design to manage various controllers and actuator sharing based on plasma and plant events [2, 3]. The plasma monitor and supervisory controller takes high-level control decisions on how to continue the discharge based on

*Corresponding author
Email address: `trang.vu@epfl.ch` (N.M.Trang VU[1])

the observed state of the plasma. Then, the actuator manager determines the resource allocation for each controller. Some designs of PCS architecture have been proposed for AUG [4], DIII-D and KSTAR [5], WEST [6] and ITER [7].

The work on AUG [2, 8] provides first examples of an actuator manager, using Electron Cyclotron Resonance Heating (ECRH) system for NTM control. While this shows an excellent first demonstration of actuator management (AM) for this application, most of the decisions for activating and prioritizing various NTM tasks is made by the NTM controller. When multiple control tasks, not only NTM control related, are to be considered, or multiple actuators are to be managed, it would be beneficial to strictly separate between controllers, supervision, and actuator management. On the other hand, [5] focuses on a supervisory logic capable of real-time off-normal event handling tested on DIII-D and KSTAR. Similarly [9] describes a plasma discharge management system, where the supervisory layer makes high-level decisions on event mitigation strategies; this is validated on simulation based on a typical Tore Supra plasma scenario. In these examples, actuator management is done by direct selection of actuators via prioritization. A more optimal solution would require a more complex actuator assignment scheme as is proposed in this paper. In [3], a synthesis of possible architectures for integrated control and actuator sharing is presented, as well as a design of an efficient algorithm. It is argued that the choice of an appropriate PCS architecture depends on the scale and complexity of the tokamak actuator and diagnostics subsystems. The present work proposes a generic PCS architecture which is easily adapted to every tokamak regardless of the complexity of the tokamak subsystems. In particular we demonstrate the first complete implementation of a supervisory controller and an actuator manager as well as experimental results on TCV.

The first contribution of this work is to regroup the components in the PCS into two layers: a *tokamak-dependent layer* and a *tokamak-agnostic layer*. The tokamak-dependent layer provides plasma state and actuator state representation to the tokamak-agnostic layer, then translates generic commands into commands to actuators. The tokamak-agnostic layer is independent of the tokamak subsystems (diagnostics/ actuators), therefore it can be independently developed and maintained, while only the tokamak-dependent layer should be adapted for each tokamak.

The second contribution is to define a *task-based* approach where a supervisory controller and an actuator manager handle *control tasks* (and not controllers) using generic actuator resources and controllers. The task-based approach has several advantages:

- It provides an abstraction layer for operators or physicists as they only need to specify the control tasks from physics goals (or pulse schedule), which are generic and similar among different tokamaks, regardless of the details of the relevant controllers and actuators. This greatly facilitates the interaction between operation and plasma control system software.

- The fact that high-level decisions that depend on the plasma and actuator states are taken away from controllers makes it possible to design controllers in a more generic way, independent of the tokamak or the scenario specific properties.

- The task-based approach provides a natural way to prioritize the task execution order, which leads to clearly distinguished *parallel* and *sequential* tasks. Parallel tasks can be performed without being aware of the resource requests of the other tasks, while sequential tasks depend on the resource requests of the others.

The third contribution of this work is to propose a standardization of interfaces between the components in the tokamak-agnostic layer. These standardized interfaces help to reduce implementation errors and improve maintainability, while granting the flexibility to add or remove controllers. Also, these standardized interfaces allow independent development and testing of each component in this layer.

In Sec. 2, we first introduce the proposed architecture of the PCS for integrated control with the tokamak-dependent layer and the tokamak-agnostic layer. Then the task-based approach and each component in the tokamak-agnostic layer, including the supervisory controller, actuator manager and controllers are discussed in Sec. 2.1 and 2.2. The standardized interfaces between them are also presented in Sec. 2.2, with more details in Appendix A. To clarify the idea of a task-based integrated control scheme, we analyze a concrete example in Sec. 2.3. The proposed solution is validated in simulation and implemented on TCV [10]. First experimental results are shown in Sec. 3, where integrated control of β, NTM and H-mode are demonstrated with central heating and co-current drive. An application to a demonstration ITER scenario, studied in Sec. 4, confirms its applicability and flexibility to systematically handle a large number of tasks and actuators. We conclude in Sec. 5.

2. Generic plasma control system structure

In this section, the proposed architecture of our generic PCS is presented (Fig. 1). The user interface allows the tokamak operators to configure on one hand the pulse schedule of the discharge, and on the other hand the parameters of the components in the PCS (see Sec.3.1 for an example). The tokamak-dependent layer has to be designed for each tokamak but the structure may remain the same as proposed in this work (Fig. 1). In this layer, an *actuator interface* converts the controller commands into the input signals for local actuator controllers and a generic *plasma and actuator state reconstruction* provides real-time information on the plasma state, actuator states and limits, by merging information from multiple plasma diagnostics and local actuator control systems. This state reconstruction may include sophisticated algorithms such as equilibrium reconstruction, profile estimation, ray tracing, etc. The output of the state reconstruction is a representation, in a generic form, of the overall state of the plasma and the key tokamak subsystems. This representation should not depend on the details of the diagnostics used to estimate the plasma state. One example of such a generic state reconstruction code is the RAPTOR-observer [11, 12] which is presently implemented on AUG and TCV.

In this work, we only focus on the development of the tokamak-agnostic layer (detail in 2.2). It is worth noting that these components are tokamak-independent in the sense that their layouts, the communication between them (their connection signals) and their functionalities/ algorithms/ implementations are defined in a general way that should not be aware of the tokamak specificity (hence the name agnostic). Therefore, the particular usage of a specific controller in a specific tokamak for a specific pulse schedule is defined through external parameters and not through a specific implementation of the controllers. For example, a feedforward controller is defined with a general input-output interface like the other controllers (see Appendix A.4), and with a general functionality to provide feedforward commands. The value of the feedforward power is however determined by the user and is indeed different for each tokamak and plasma scenario.

2.1. Task-based vs. controller-based approach

A standard approach to develop an actuator manager is to directly link controllers and actuators together. We refer to this as a controller-based approach, and this is the paradigm followed e.g. in [2, 3]. In this approach, the controllers are responsible for making high-level decisions on what they should do according to the plasma and actuator states. Therefore, they often include detailed knowledge about the specific subsystems of a tokamak. The actuator manager also needs to be aware of details of a particular controller in order to assign relevant actuator resources with respect to the controller requests. Thus, the actuator manager and controllers built this way will be machine dependent. Moreover, it is not straightforward to add or remove a controller, since it may include high level decisions in addition to its specific control task.

The task-based approach we present here is based on the idea that, from the physics goals of each discharge, one can directly distill a set of *control tasks* regardless of the machine specifics, such as β control, NTM control, H-mode entry control, plasma shape control, etc. In this approach, high-level decisions on carrying out control tasks are taken away from controllers and are made in a separate supervisory layer, making them tokamak-independent.

2.2. Details of the tokamak-agnostic layer

Fig. 2 is a zoom of the tokamak-agnostic layer with the details on the components as well as on the signals propagating among them. These components are elaborated hereafter, while their interfaces are studied in detail in Appendix A. This layer, using the task-based approach, specifically deals with the execution of control tasks, without requiring information on features of diagnostics or actuators (kind, functionality, etc.). Thus we also call it the *tokamak-agnostic task layer* or simply *task layer* for short.

In this respect, the information flow between the various components in this layer is also performed as an abstract representation of a task. Although the requirements for actuators acting on the plasma may differ among various tasks, we can generalize these requests into the triplet (*amplitude*, *position*, and *type*). In this way, *amplitude* can represent either a power amplitude, gas flux amplitude, or pellet particle flux (injected particles per second), etc. *Position* is the location of actuation in the plasma for heating and current drive purpose, or the relative position of actuators (gas valves), etc. *Type* is the information about actuator kinds. It may be the direction for heating and current drive; the gas species deuterium, tritium or helium for gas injection, or other information which allows the AM resource allocator to classify the actuators. Consequently, the actuator states and capacities also have to be parametrized in this generalized form, such that: the actuators are enumerated (i), $i = 1, 2, 3, ...$, their capacities are represented as ranges between the minimum and maximum values of (*amplitude*, *position*, *type*) and an active signal represents their state (*on* when it is capable of acting on the plasma, and *off* when it is not). Note that all signals in this layer, except for active and task parameter, are composed of such a triplet (*amplitude*, *position*, *type*). This abstract representation can be extended for more components when we expand the type of control tasks and actuators in future works.

Inside the layer, the hierarchical architecture [3, 13, Sec. 2] is employed for the component connection. Highlighted by the advantages of transparency and ease of im-

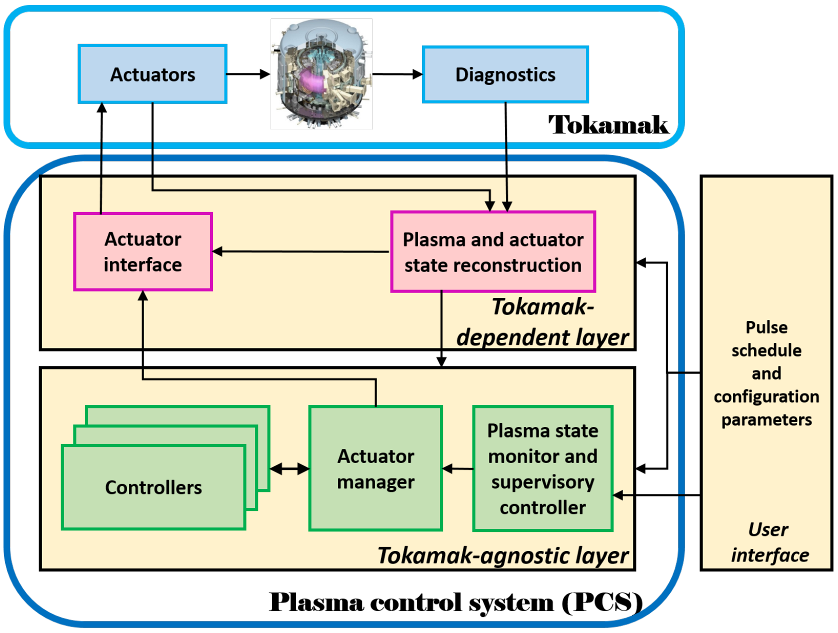
Actuators
Diagnostics
Tokamak
Actuator interface
Plasma and actuator state reconstruction
Tokamak-dependent layer
Controllers
Actuator manager
Plasma state monitor and supervisory controller
Tokamak-agnostic layer
Pulse schedule and configuration parameters
User interface
Plasma control system (PCS)

Figure 1: Overview of our generic tokamak control system architecture: the tokamak system and the plasma control system (PCS). The components in the PCS are split into two layers: the tokamak-dependent layer and the tokamak-agnostic layer. The latter is tokamak-independent and can be applied to different tokamaks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

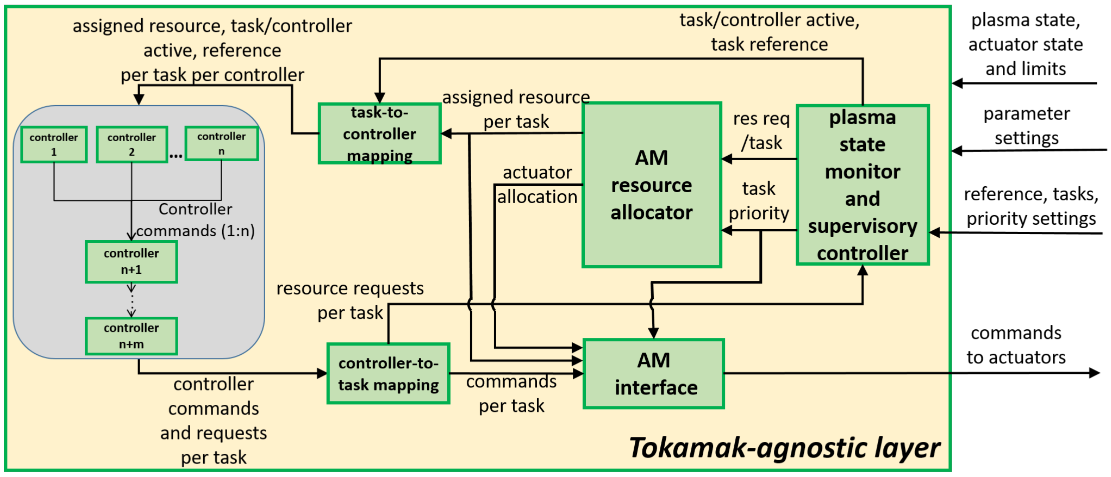
assigned resource, task/controller active, reference per task per controller
task/controller active, task reference
plasma state, actuator state and limits
controller 1
controller 2
controller n
Controller commands (1:n)
controller n+1
controller n+m
task-to-controller mapping
assigned resource per task
actuator allocation
AM resource allocator
res req /task
task priority
plasma state monitor and supervisory controller
parameter settings
reference, tasks, priority settings
resource requests per task
controller-to-task mapping
commands per task
AM interface
commands to actuators
controller commands and requests per task
Tokamak-agnostic layer

Figure 2: Schematic block diagram of the *tokamak-agnostic layer* of the PCS, including supervisory controller, actuator manager and controllers (parallel (1-n) and sequential ((n+1)-(n+m)) controllers)

plementation, this architecture has been elaborated in various applications [7, 4, 6]. The detailed function of each component in the task layer is described next.

- The *plasma state monitor and supervisory controller* computes the task priority and task active based on the plasma state, the actuator state and limits. Depending on the plasma evolution with respect to physics goals, the supervisory controller decides how to continue the discharge by activating relevant tasks and giving them appropriate priorities. It can change the references and task parameters according to the pulse schedule settings. The cross-coupling between control tasks is also taken into account by this supervisory level, e.g. the case when some control tasks cannot be performed simultaneously. Details of the supervisory controller within the scope of the proposed generic PCS architecture are discussed in [14].

- The *AM resource allocator* assigns virtual resources, which is the actuation capability of the allocated actuators (e.g. range of amplitude, position, and type) to each activated task, by solving an optimization problem based on resource limits, task priority and resource requests per task. Details are described in Appendix A.1).

- The *AM interface* distributes the resource commands from tasks to relevant actuators. Furthermore, it also ensures these commands are within the actual actuator limits.

- The *task-to-controller mapping* and *controller-to-task mapping* provide the link between the tasks and the controllers which handle these tasks (see example in Table. 3).

- The *controllers* receive their assigned virtual resources (e.g. ranges of amplitude, position, type) as well as plasma state information, and execute the (feedback) control laws for their tasks. They send out the controller commands per task (commands of amplitude, position, and type for the actuators) and the controller requests per task (a range of amplitude, position, and type for the actuator manager). It is important to distinguish between these two outputs of the controllers: the controller commands (after being limited by the assigned virtual resources) will be distributed to the actuators; while the resource requests, without limits, will be communicated to the AM to be considered for the actuator allocation at the following time step.

The controllers can be executed either in parallel or in sequence. The controllers in the parallel group (controllers $1 \to n$) handle parallel tasks, which are mutually decoupled, meaning that these controllers can execute their control laws independently from the other controllers. On the contrary, the controllers in the sequential group (Fig. 2, controllers $n + 1 \to m$) handle sequential tasks, which need to be aware of the commands of the other tasks. Hence, these controllers need to receive the commands of the other controllers so that they can compute their own outputs. An example thereof is the LH-mode controller, which is discussed in detail in Sec. 4.1. The tasks and the controllers are thus classified into two groups: parallel and sequential groups. Hence task priorities are separated into these two groups as well. This classification and the task priorities are defined in the pulse-schedule via the user interface. For instance, the controller topology presented in Fig. 2 can cover all types of controllers in different control domains (heating and current drive, fuelling, shaping, etc.). In the sequential chain, the execution order is determined by the task priorities of sequential tasks. Therefore the execution order of sequential controllers are not fixed but dynamically depends on their mapped tasks.

It is important to note that the AM resource allocator only assigns actuators for parallel tasks, whereas the AM interface decides relevant actuators for sequential tasks.

Notice also that the controllers interact only with the actuator manager via the *task-to-controller mapping* and the *controller-to-task mapping*. Thus there is no direct connection between controllers and the actuator systems.

The interfaces of various components in this task layer are generally shown in Fig. 2 with main inputs and outputs. These proposed interfaces allow us to easily add or remove tasks and controllers for each discharge without changing the generic interface. For more details of these interfaces, see Appendix A and [14] for the supervisory controller.

To clarify the task and controller concepts that we propose in this work, a concrete example is discussed in Sec. 2.3 below.

2.3. Example of pulse schedule: tasks and controllers in a discharge

To illustrate the task-based approach described in the previous section, we present an example of integrated control featuring several tasks and several controllers. The physics goal, defined by the pulse schedule, is to control the plasma performance (e.g. plasma kinetic pressure β control) while maintaining an H-mode regime. In other words, the physics goal is not only to drive plasma current and β to their references, but also simultaneously to ensure the total heating power is higher than the critical threshold P_{LH} to keep the plasma in H-mode. This threshold power is typically determined via a scaling law [15]. Also, during the discharge, NTMs may appear, so NTM control [16] is required to stabilize (or suppress) NTMs. It has been shown that preventing an NTM by preemption can be more efficient than stabilizing it [17, 18], so our NTM control strategy will use a combination of both. The 2/1 and 3/2 NTMs (NTMs at the q surfaces $q = 2/1$ and $q = 3/2$ respectively) are the most significant ones used in this example. Moreover in the worst case when the plasma is strongly destabilized, leading to a disruption, disruption mitigation [19] must be considered.

According to the pulse schedule described above, several tasks can be defined. These are listed in Table 1, (note that sequential tasks and controllers are indicated by an upper $\star$ before their names, the same indication is also found in Table 2, 5, 7). The controllers capable of executing these tasks are enumerated in Table 2, and the mapping between tasks and controllers are defined in Table 3. In order to avoid controller cross-talk (i.e. when a combination of controllers handles one task), we define the rule that each task is assigned to only one controller, while one controller can handle several tasks (as seen in Table 3).

Remark 1. Note that the previous rule does not limit the kinds of actuators that each task and each controller can use. In fact, a task can require several kinds of actuators using the type request range. For example a "stay in burn mode" task can ask for heating actuators as well as fuelling and radiation actuators. In this case, the controller may have several sub-modules to deal with different actuator kinds.

At each PCS cycle time during the discharge, the supervisory controller will generate active signals and priorities for relevant tasks (Table 1) according to priorities defined through the pulse schedule as well as the actual evolution of the plasma and actuator states. The AM re-

Task 1	β control
Task 2	2/1 NTM stabilization
Task 3	2/1 NTM preemption
Task 4	3/2 NTM stabilization
Task 5	3/2 NTM preemption
Task 6	*be in H-mode
Task 7	Disruption mitigation

Table 1: Example of task list in a discharge, the sequential task (be in H-mode) is indicated by an upper $\star$

Controller 1	performance controller
Controller 2	NTM controller
Controller 3	*LH-mode controller
Controller 4	disruption mitigation valve controller

Table 2: Example of controller list

source allocator will assign the virtual resources to the active parallel tasks, taking into account the relative priority of these tasks. The controllers will execute their control laws and send the commands of each task to the AM interface. The latter first examines the commands from the sequential task, be in H-mode, in order to modify the commands from the parallel tasks so-that the total heating power respects the critical H-mode threshold, then the adjusted commands are sent to the actuators to achieve actual active tasks.

Remark 2. Note here the crucial differences between a controller-based approach and the task-based approach:

- in a controller-based approach, the NTM controller would have to decide if preemption or stabilization should be performed, while with the task-based approach, such a high-level decision is carried out by the supervisory controller instead.
- the task-based approach allows to distinguish the priorities of the tasks executed by one controller (e.g. NTM stabilization and NTM preemption tasks by the NTM controller), hence the actuator allocation for each task is clearly separated. While there is only one priority for each controller in a controller-based approach. For example, if the NTM controller has higher priority than the β controller, the NTM preemption task will always be considered more important than the β control task, which is not necessarily the case.

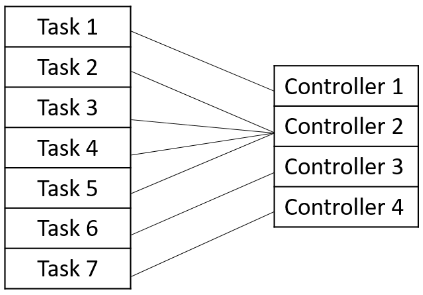

Table 3: Task-controller mapping, linking the tasks in Table 1 and the controllers in Table 2. The task order (2 to 5) assigned to the controller 2 is used to compute the inverse mapping from controllers to tasks.

Remark 3. Note as well that despite this being a relatively simple example of β control in H-mode, seven tasks and four controllers are needed, not to mention the actuators, but the generic nature of the interfaces helps to maintain this manageable even with more complicated cases.

Note also that separating "β control" and "be in H-mode" tasks, for example, is the key to simplify the tasks design, avoiding task cross-talk, while ensuring all control tasks to be effective.

In the following, the first results of TCV experiments and the results of an ITER test on integrated control are presented. Exactly the same tokamak-agnostic layer is used in both cases. For the sake of simplicity, only heating and current drive actuators are considered in these tests. The proposed PCS structure remains the same for future extension to actuators for fuelling, shaping and other purposes.

3. First results on TCV

The proposed integrated control scheme has been tested and implemented on TCV. For this purpose, it was implemented in Matlab/Simulink [20], from which C code was generated and included in the TCV digital real-time control system [21], [22].

Two test scenarios are described in this section. In the first test, the physics goal is to control β using Electron Cyclotron (EC) power, while switching to the control of a 2/1 NTM when the mode appears during the discharge. In this test, only the power and injection angle of the EC H&CD system are controlled while other actuators such as the ohmic coils and gas valves are not considered. In the second test, β control in L- and H-mode is demonstrated by varying the powers of the EC and the Neutral Beam Injection (NBI).

Remark 4. Due to the limit of H&CD actuator systems in TCV, the following experiments may appear simple with very small numbers of considered actuators and tasks. The main aim here is to demonstrate the first complete implementation and real-time test of the proposed PCS architecture. Note that the task layer is identical to the one used in more complex scenarios with a large number of actuators and tasks, as in Sec.4, demonstrating its generic use and possibility to be tested in different tokamaks.

3.1. First test: β control with NTM stabilization

In this test, the main goal of the discharge is to achieve β control, while NTM stabilization is also considered in order to keep the discharge with good performance. A feedforward central co-CD heating task (central co-CD) at the beginning of the discharge is used to establish the plasma in an operational equilibrium, before switching on β control task (β control) and NTM stabilization task (NTM2/1 stab). For NTM2/1 stab, the NTM controller will move the assigned actuator(s) to the target surface (i.e. surface $q = 2$) to stabilize the mode with full allocated power. During the time when the EC deposition is changing from the present position to the target, the NTM controller requests a lower power in order to limit perturbing the plasma.

This pulse schedule is configured in the user interface where the considered tasks are chosen, accompanying with their default priorities, their active time, as well as the controllers that handle them. This is summarized in Table. 4. Here three tasks are mapped to three controllers: feedforward, NTM and performance [23] controllers. Note that NTM2/1 stab is active only if a 2/1 mode is detected by the plasma state monitor.

The user interface also allows one to regulate the configuration parameters of each component in the PCS. In each controller, the tunable parameters (such as feedback gain, feedforward wave form, and some specific parameters of the controllers) are defined for each task. These tunable parameters are scenario- and device-specific. Other configuration parameters for the supervisory controller, AM resource allocator and AM interface are provided via the user interface as well.

Two EC launchers (indicated as $L4$ and $L6$) are available for the considered tasks. The power limits, actual power and position of these actuators are given in real-time by the actuator subsystem. Due to technical issues, the lower power limit is set to 0.2 MW to prevent EC trip during the discharge. Note that $L4$ and $L6$ have the same EC power supply, thus they always have (approximately) the same power, and can not be controlled with different power commands. However their deposition location can be different and changed in real-time. This means that

task	priority	active time (s)	active condition	controller
central co-CD	0.2	[0.4, 0.55]		feedforward
NTM2/1 stab	1	[0.5, 2.5]	2/1 NTM detection	NTM
β control	0.8	[0.5, 2.5]		performance

Table 4: TCV first test: table of tasks, controllers and the mapping between them

the higher priority task receiving one (or both) of these actuators controls the powers of both of them, whereas the other task, if it receives the remaining actuator, can only control the position of that actuator. As a result, the power range of the assigned virtual resource for the second task is fixed at the actual power of the assigned actuators (i.e. $P_{min\,assigned} = P_{max\,assigned} = P_{actual}$) by the AM resource allocator. The AM interface will also take this fact into account when distributing power to each actuator.

Based on other experimental results on TCV, it is known that 2/1 NTMs can be triggered with central co-CD at the relatively low plasma current [16, 18, 24], this is why NTM2/1 stab has been included. The β control request is fixed at $[0,\ 1MW]$ corresponding to the maximum amplitude range for the EC power, thus the task can receive both $L4$ and $L6$ (with maximum $0.5MW$ each) when available. These task active and priorities are managed in real-time by the supervisory controller.

In Fig. 3, the results of shot 58821 are presented. Initially, in the interval ①-②, central co-CD is the only activated task, hence it has the highest priority and is assigned both $L4$ and $L6$ (see Fig. 3a and b), after that it receives the lowest priority due to the activation of NTM2/1 stab and β control. At ② and ⑥, a 2/1 NTM is detected, the supervisory controller sets the highest priority to NTM2/1 stab, and the second-highest priority to β control. Accordingly, the AM resource allocator assigns $L4$ to NTM2/1 stab and $L6$ to β control, with nothing left for central co-

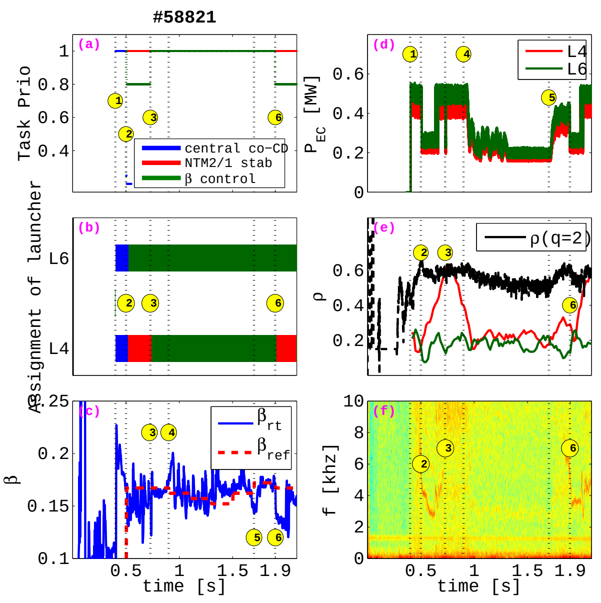

Figure 3: TCV shot 58821: result of integrated control experiment with three control tasks and two EC actuators in TCV, showing a sequence of events: ① central co-CD is activated; ② An NTM is coincidentally detected at the same time as activation of β control. L4 is assigned to stabilize the mode and moved to the $q = 2$ surface. ③ The NTM is stabilized and β control has the highest priority. It asks L4 to return to the plasma center. ④ β_{ref} is decreased, the power command reaches the actuator lower limits and β oscillates. ⑤ β_{ref} is increased, β recovers and triggers an NTM at ⑥. (a) Task priorities decided by the supervisory controller. (b) Launcher allocation from the AM resource allocator, same legend as (a). (c) β_{ref} vs real-time β_{rt} estimated by LIUQE code [25]. (d) EC powers of $L4$ and $L6$. (e) $L4$ and $L6$ positions vs NTM position, corresponding to the radial location of the $\rho(q = 2)$ surface, estimated by TORBEAM code [26], same legend as (d). (f) NTM freq spectrum shows the presence of the mode.

CD. During the phase without NTM, starting at ③, β control always gets the highest priority and thus is assigned both $L4$ and $L6$, given the high power request from this task ($[0,\ 1MW]$).

Regarding the EC powers and launcher positions controlled by the two considered controllers (Fig. 3d and e), in the interval ①-②, maximum power of $L4$ and $L6$ as well as the central position are required by central co-CD.

During NTM2/1 stab, $L4$ power is adapted by the NTM controller according to the distance between it and the target ($\rho\,(q=2)$). During the β control-only period, the central position and the power of both launchers are decided by the performance controller. One can notice that the performance controller is assigned both $L4$ and $L6$ in the interval ③-⑥ since it sends a high power request to the AM resource allocator in order to get a large virtual resource. However, it only sends a lower power as commands to these launchers in the interval ④-⑤, according to the actual β. It is to show that the two controller outputs, the controller commands and the controller requests, can be totally independent.

It should be noted that in the presence of the NTM, β cannot reach the reference since its available power is insufficient. Without NTM on the whole interval ③-⑥, the real-time β matches quite well the reference only in the intervals ③-④ and ⑤-⑥. During ④-⑤, the real-time β cannot track the reference which is so low that the lower limits of the EC power are reached. After ⑤, the β reference increases again, so that successful control can again be achieved. The higher β however triggers an NTM at ⑥, which is again detected and handled by the system. The end of the discharge is reached before the mode could be stabilized, but the NTM2/1 stab task was successfully fulfilled at ③.

3.2. Second test: β control in L and H-mode

The main objective of this test is to make the plasma β to track a reference trajectory, both in L and H-mode regimes. To implement this, both parallel tasks (feedforward FF control and β control) and sequential tasks (the LH-mode tasks: be in L-mode and be in H-mode) are required. The β control task is accomplished by feedback control of the power of some actuators based on the tracking error. The be in L-mode task and the be in H-mode task (see Appendix B for details) are alternatively active during the discharge. These LH-mode tasks are accomplished by constraining the total input power to the plasma to a maximum/minimum in order to keep it below/above the LH transition threshold respectively, with some margin. If the total power required by the other control tasks is already within the specified limits by the LH-mode tasks, then no change to the power is requested.

This example serves to illustrate clearly the difference between parallel and sequential tasks (see Sec. 2.2 for details). The LH-mode tasks are sequential since their commands depend on the commands of all the other controllers, hence they are executed (sequentially) after the others. The other tasks are parallel, since they do not depend on each other and can be executed independently.

The considered tasks and controllers in this test are listed in Table 5. In this discharge, three EC launchers $L7$, $L8$ and $L9$ and one NBI are available with different power limits described in Table 6.

task	priority	active time(s)	controller
FF control	0.45	[0.6, 0.7]	feedforward
β control	0.6	[0.7, 1.8]	performance
*be in L-mode	0.8	[0, 0.9] & [1.3, 1.8]	*LH-mode
*be in H-mode	1	[0.9, 1.3]	

Table 5: TCV second test: table of tasks, controllers and the mapping between them

actuator	name	power limits [MW]
EC	$L7$	[0.1, 0.5]
	$L8$	[0.1, 0.2]
	$L9$	[0.1, 0.5]
NBI	NBI	[0.05, 1.05]

Table 6: TCV second test: available actuators and their power limits

Fig. 4 shows the results of shot 62441. The be in H-mode task is active in the interval ③-⑥, and the be in L-mode task is active during the rest of the discharge. At

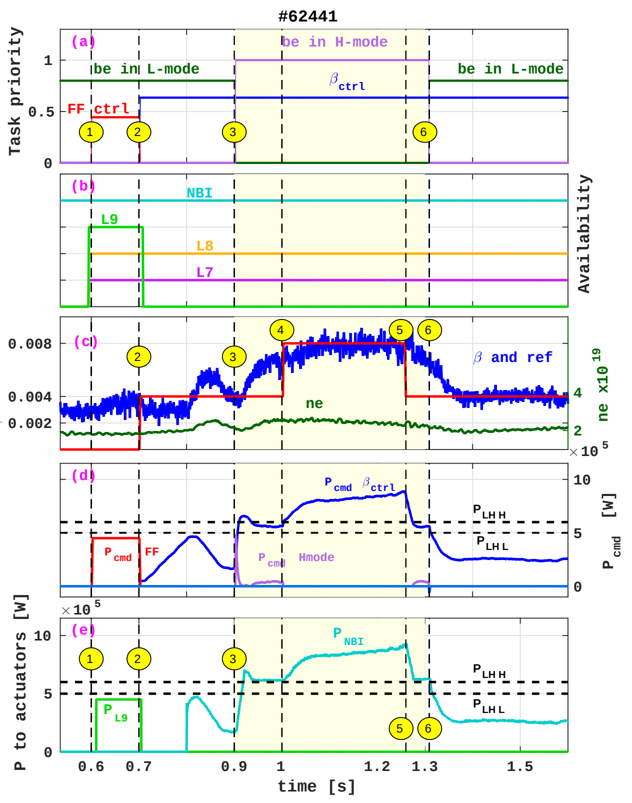

Figure 4: TCV shot 62441: result of an integrated control experiment with four control tasks and four actuators in TCV, showing a sequence of events: ① FF control is activated and is assigned $L9$; ② β control is on and FF control is off. β control is assigned NBI. NBI switches on some time later and the reference β is reached after an overshoot. ③ be in H-mode is active. LH-mode controller commands more power to NBI to go to and maintain H-mode. ④ β_{ref} is increased, the power command for NBI increases accordingly and β reaches the reference in H-mode. ⑤ β_{ref} is decreased, but since the plasma should still stay in H-mode phase, NBI power is decreased but still higher than the limit $P_{LHHmode}$, β decreases but can not reach the reference. ⑥ the be in L-mode is active, the NBI power can decrease and β reaches the reference in L-mode. (a) Task priorities decided by the supervisory controller. (b) Actuator availability. (c) Estimated real-time β vs β_{ref} and plasma density n_e showing the L and H-mode transitions. (d) Power commands from each task. (e) Readback power from actuators.

first, FF control (feedforward) is activated (①-②) and is assigned $L9$. Then from ② until the end of the discharge, β control is switched on and FF control is switched off. The NBI is assigned to β control thanks to its high reliability, but it only fires 0.1s later for some technical reasons that will be discussed later. Since this is not known to the controller, the integrator term in the β controller continues integrating the error, eventually causing an overshoot of β when the NBI fires (Fig. 4c). The control of β is then regained and β is decreased to reach the reference. The power of $L9$, when not in use, is asked to decrease to its minimum, but then the corresponding gyrotron trips and is unavailable for the rest of the discharge. When the be in H-mode task is active at ③, the LH-mode controller compensates the missing power in order to bring the plasma to H-mode and to remain in H-mode during ③-④. Consequently in this interval, β can not reach the low reference which is unachievable in H-mode. During ④-⑤, the target beta is increased and β matches very well the reference in H-mode. In the interval ⑤-⑥, the β reference is decreased (on purpose to a value corresponding to an L-mode plasma) and the β controller commands a low power. The be in H-mode task compensates the heating power to remain above the threshold as seen by $P_{NBI} >= P_{LHH}$ in Fig. 4e.

At ⑥, the be in L-mode task is activated again, the β controller can freely decrease the NBI power and β tracks very well the reference in L-mode.

We have not aligned the time evolution of the β references with the L- and H-mode phases on purpose, to test the supervisory controller, the actuator manager and the sequential tasks. Therefore when a low β is requested but the be in H-mode task is still active with a higher priority, the system is forced to provide the lowest power while staying in H-mode, as expected.

Via these examples, we have shown that the supervisory controller and actuator manager can adequately manage tasks and allocate actuators in a proper way. Note in particular the successful test of generic sequential tasks, which is a first demonstration to our knowledge. All scenarios were executed using the same architecture (Fig.1) but with different configuration parameters of the pulse schedule (number of tasks and controllers). We stress that the time-behavior of the controllers was not programmed. The time-evolution in the experiments is solely a result of the high-level specification of the objectives of the discharge, and the automated actions of the control system. This also means that all necessary knowledge of the plasma state and actuator states need to be available in RT. For example in the second test, the starting time for the NBI was set before the shot by the NBI operator to $0.8s$, but allowed to be controlled by the RT-system over the whole shot. Therefore the actuator manager received the information that NBI was available before it was actually the case. This is why NBI started only at $0.8s$ although it was requested before. This will be corrected for future experiments, but illustrates well the degree of integration of all the systems that need to be obtained and included in the overall RT control strategy.

In the following section, the scheme will be applied to the more complex actuator set of ITER.

4. Task layer test with ITER scenario

In order to further illustrate the capabilities of the task layer proposed in this paper, it is applied to a test with a mimicked ITER pulse-schedule and ITER control system. This system is chosen due to the complexity of its actuator systems [27] and the necessity to handle a multitude of tasks simultaneously. In this section, we describe the ITER test scenario and make some assumptions on the actuator system for the sake of simplicity.

Note that this test does not include any model of the plasma, instead, the task active from the supervisory controller will be based on the active window manually specified, mimicking a likely evolution of the plasma. This aims to demonstrate that the AM resource allocator can appropriately handle more tasks and more actuators in such a case. This also means that the performance of the considered control tasks on ITER plasma is out of the scope of this test.

4.1. Description of the test scenario

In this test scenario for ITER, we aim at simultaneous control of β and q-profile in an H-mode regime with NTM stabilization at different q surfaces ($q = 2/1$ and $q = 3/2$).

In addition, a central heating control can also be included for several purposes, such as burn control, impurity or temperature control, etc. Also a sawtooth control (at $q = 1$ surface) is considered since long period sawtooth can readily destabilize NTMs [28, 29]. In this example, a fixed power onto the target surface is desired for sawtooth control; thus this task is considered functionally equivalent to NTM preemption at the $q = 1$ surface and is handled by the same controller, now referred to as MHD controller.

A sequential task be in H-mode (or H-mode, see Appendix B) is also active during the discharge. This task ensures that the total power injected in the plasma is larger than a threshold power P_{LHH} (fixed at $57\,MW$ for this example) which is needed to bring and maintain the plasma in H-mode.

All considered tasks and their active times are listed in Table 7. To fulfil these six tasks, three controllers are employed: MHD, performance, and LH-mode controllers. The corresponding task-controller mapping is also shown in Table 7.

For simplicity, we assume constant resource requests from each task (specified in Table 8). Their resource requests, which are sent to the AM resource allocator, consist of a range of requested power P_{req}, normalized position ρ_{req} and type icd_{req}. The icd quantity allows us to classify the current drive effect of each actuator, as follows:

Task index	task name	priority	active time (s)	controller
1	NTM2/1 stab	1	[50, 70] and [100, 140]	MHD
2	NTM3/2 stab	0.8	[40, 80] and [120, 150]	
3	sawtooth control	0.2	[30, 170]	
4	$\beta + q$-profile control	0.6	[24, 160]	performance
5	central heating	0.5	[20, 166]	
6	*H-mode	0.15	[1, 200]	*LH-mode

Table 7: Task and controller lists of the ITER test, including priorities and active windows provided by the pulse schedule

- $icd \in [0.2, 1]$: Co-current
- $icd \in [-1, -0.2]$: Counter-current
- $icd \in (-0.2, 0.2)$: Heating

Remark 5. The type of each actuator is usually fixed before the discharge, i.e. the toroidal angles of EC, IC and NBI are determined in the user interface, which helps to classify them as heating, co or counter current drive actuators. It is also important to note that the type in the recent application is tailored to H&CD actuators, thus its value is in the range of $[-1, 1]$. Other actuator types, i.e. gas types, will be defined in other value ranges.

In this respect, the request of icd_{req} range of a task determines the type of actuators which will be assigned to that task. For example, since the icd_{req} of NTM2/1 stab is $[0.5, 1]$, only co-current actuators will be assigned to that tasks if needed. While the icd_{req} of H-mode is $[-1, 1]$, this task can be allocated any H&CD actuators.

Also, all controllers (except for the H-mode controller) give a command equal to the maximum power assigned from the AM resource allocator. Then, the H-mode task computes the required additional power (which may be zero if the other controllers already request sufficient power to sustain H-mode), see Appendix B.

4.2. ITER actuators and assumptions

Of the approximately 20 actuators that will eventually be connected to the ITER PCS, we consider here only the heating and current drive systems of Electron Cyclotron (EC), Ion Cyclotron (IC) and Neutral Beam Injection (NBI). We propose the following simplifying assumptions on the actuator systems for the purpose of this illustration (though the proposed solution is equally applicable to the full system):

- The EC system has 24 gyrotrons (power supply) of about $0.83MW$ each (effective power delivered to the plasma), thus $20MW$ in total, and 11 steerable mirrors (detail in [30]).
  - 1 Equatorial Launcher (EC_EL) with 3 mirrors, which we assume to split into 2 groups: the first group has 1 mirror for counter-CD (CD in the opposite direction from the plasma current), and the second group has 2 mirrors for co-CD.
  - 4 Upper Launchers (EC_UL) with 2 mirrors of each launcher for off-axis co-CD
  - In this example, we reserve a group of 6 gyrotrons for counter-CD, that we name actuator -1EC (-1 stands for counter-CD). The other 18 gyrotrons are named actuators 1EC (1 stands for co-CD).
- The IC system, for central heating purpose, has 2 antennas which deliver $20MW$ into the plasma, thus $10MW$ max per antenna. They are named 0IC (0 stands for heating).

Task index	Request index	$P_{req}\,(MW)$	ρ_{req}	icd_{req}
1	1	[6, 10]	[0.5, 0.6]	[0.5, 1]
2	2	[3, 5]	[0.3, 0.4]	[0.5, 1]
3	3	[1, 5]	[0.1, 0.2]	[0.5, 1]
4	4	[1, 4]	[0, 0.2]	$[-1, -0.5]$
	5	[5, 8]	[0, 0.2]	[0.5, 1]
5	6	[10, 15]	[0, 0.2]	[0, 1]
6	7	[0, 0]	[0, 1]	$[-1, 1]$

Table 8: Task (fixed) requests of the ITER test

- The NBI system has 2 sources of $16.5MW$ each, named 1NBI (1 stands for co-CD), which inject current in the co-current direction as well as heating the plasma. The NBI power can be only *on* or *off* ($P_{NBI} \in \{0,\ 16.5MW\}$) and the deposition location, which is assumed to be near the center in this scenario, cannot be changed.

Notice that all actuators can in principle be assigned to any considered tasks. On one hand, NTM stabilization is proved more efficient using co-ECCD [31][18],. Consequently, only 1EC actuators will be used for the NTM tasks performed by the MHD controller and all others are marked as *excluded* actuators. On the other hand, in order to illustrate the actuator switching between tasks, EC actuators are also preferred actuators for all other tasks, except for central heating which prefers IC actuators. This is determined in the pulse schedule, and independent of the actual controllers used.

4.3. ITER test results

In this test scenario we consider only a $200s$ period during the H-mode flat-top for simplicity. First, we show a result for the normal case in Fig. 5 when all actuators work perfectly. In the second example in Fig. 6, we demonstrate the ability of the actuator resource allocation to compensate for the loss of three EC, one IC and one NBI actuators during the discharge.

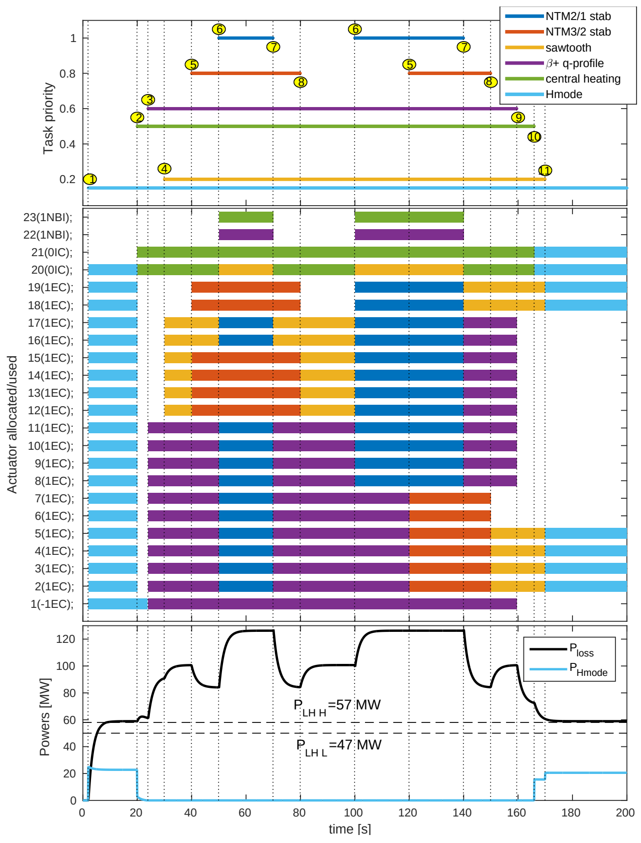

Figure 5: ITER test scenario 1: (top) time-varying task priorities, (middle) corresponding actuator allocation computed by the actuator management for each task, (bottom) power across the separatrix P_{loss} and H-mode task power command. ① H-mode is activated, power is increased to the level required for H-mode. ② central heating is activated; ③ $\beta + q$-profile control is activated; ④ sawtooth control is activated; ⑤ a 3/2 NTM is detected, various actuators are assigned to suppressing it; ⑥ a 2/1 NTM is detected, actuators are re-prioritized according to the tasks priorities; ⑦ 2/1 NTM is stabilized; ⑧ 3/2 NTM is stabilized; ⑨ $\beta + q$-profile control is turned off; ⑩ central heating is turned off; ⑪ sawtooth control is turned off, H-mode task is the only active task and it ensures the total injected power remains above the threshold.

Fig. 5 presents the result for the first test including: the prescribed time-varying task priorities (top panel), the computed actuator allocation for each task (middle panel) and the command power of H-mode P_{Hmode} versus the plasma total heating power P_{loss} (bottom panel). The latter includes total auxiliary power from the external heating source (EC, IC and NBI in this test), the self-heating power P_α in case of fusion reaction, and the radiation term P_{Brem} (see Appendix B for more detail). As can be seen in the top panel, H-mode is enabled for the whole discharge (from ①). This task requests sufficient heating power to stay above the power threshold , but this is necessary only at the very beginning and the end of the discharge when it is the only active task. After a small amount of time, central heating and $\beta + q$-profile control are also activated (② and ③) and are assigned their actuators. In particular $\beta + q$-profile control, which requires both co-CD and counter-CD, is assigned -1EC and 1EC actuators, which have opposite current drive capability. central heating is given two 0IC actuators which fully satisfies its request. Three MHD tasks NTM2/1 stab, NTM3/2 stab and sawtooth control share 1EC actuators $2-19$. NTM2/1 stab, which has the highest priority when a 2/1 NTM is detected (⑥), gets 12× 1EC actuators; while NTM3/2 stab and sawtooth control receive 6× 1EC actuator each (⑤ and ④). When the 2/1 and 3/2 NTMs happen at the same time, sawtooth control gets an 0IC , since it has a lower priority and no 1EC actuators are left available. Similarly, each of central heating and $\beta+q$-profile control receives one 1NBI to be fulfilled.

The second test, Fig. 6, shows the case when some actuators trip during the discharge. Immediately, a new actuator allocation is calculated without these tripped actuators by the AM resource allocator. More actuator switching is required to fulfil the high priority tasks, for example when 14(1EC) trips, 4(1EC) is switched from $\beta + q$-profile control to 2/1 NTM stab. Due to the limit number of available actuators in this test, some tasks cannot be fully satisfied, such as NTM3/2 stab or central heating. Particularly, sawtooth control receives nothing during $t = [100s, 160s]$ because it has the lowest priority and all available actuators are assigned to the other tasks.

Figs. 5 and 6 illustrate well that an efficient actuator allocation can only be decided in real-time and needs to be flexible, since the number of concurrent tasks and available actuators can vary significantly during a long pulse. It should be emphasized that the same task layer has been used for the TCV experiments and for the ITER tests. Only the actuator setup and interfaces, as well as the pulse schedule are different. This confirms the generic nature of our proposed PCS architecture.

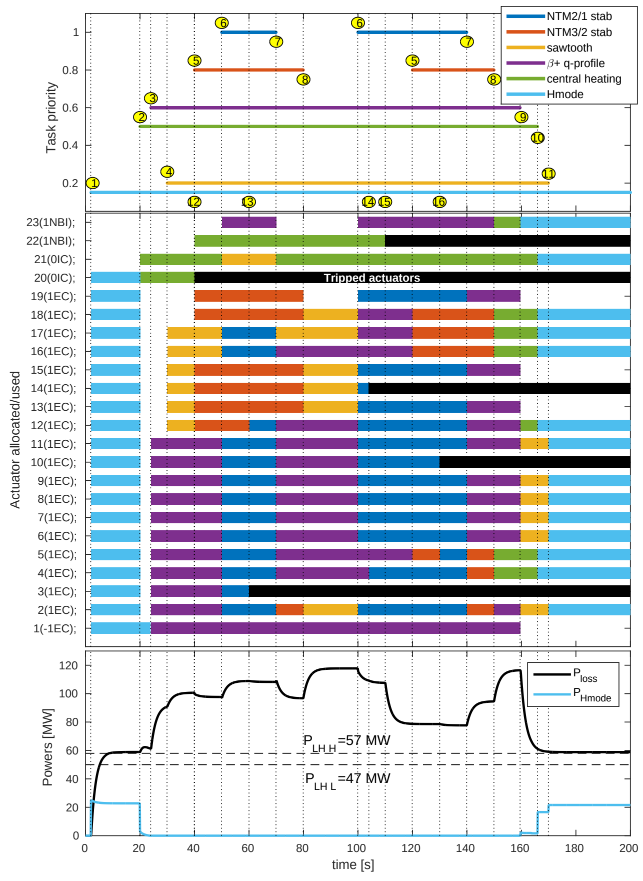

Figure 6: ITER test with tripped actuators: Task priority and actuator allocation for each task with its corresponding color ①-⑪ same as Fig. 5, ⑫ one 0IC trips at $t = 40s$; ⑬,⑭,⑯ three 1EC trip at $t = 60s, 105s, 130s$; ⑮ one 1NBI trips at $t = 110s$.

5. Conclusion

A generic architecture for task-based integrated plasma control is developed in this work. Using a task-based approach for integrated control, we have built a task layer composed of a supervisory controller, an actuator manager and controllers, which is completely tokamak-agnostic. As a consequence, the proposed task layer can be widely and easily applied to different tokamaks. The developed supervisory controller and actuator manager can deal with the actuator sharing issue for multiple plasma control tasks in the same discharge. This work has shown the design, implementation and experimental demonstration of actuator management for integrated control. Generic, standardized interfaces between the various components were developed. The scheme has been implemented in TCV and successfully used in experiments. The tests for ITER confirm the ability to systematically and dynamically allocate shared actuators to simultaneously active tasks of the proposed work also for large amounts of actuators and tasks.

In the future, we aim to extend the developed scheme to other tokamaks, in particular those where plasma monitoring, actuator sharing and integrated control are crucial to fulfil operational requirements.

Appendix A. Task layer components and interfaces

In this section the components of the *task layer* are described in more detail, also clarifying the interface of each component in Fig. 2. We present the various interfaces composed of AM resource allocator, AM interface, controllers, task-to-controller mapping, and controller-to-task mapping components, while the plasma state monitor and supervisory controller interface is elaborated in [14]. Each

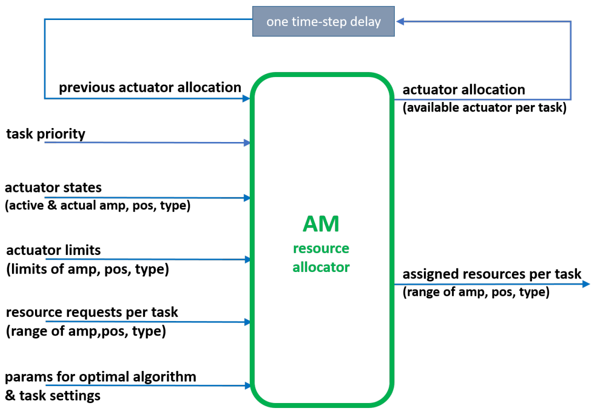
one time-step delay
previous actuator allocation
task priority
actuator states
(active & actual amp, pos, type)
actuator limits
(limits of amp, pos, type)
resource requests per task
(range of amp,pos, type)
params for optimal algorithm
& task settings
AM
resource
allocator
actuator allocation
(available actuator per task)
assigned resources per task
(range of amp, pos, type)

Figure A.7: AM resource allocator inputs and outputs

interface is presented with its inputs and outputs and their meaning.

Appendix A.1. Actuator Manager resource allocator

In this subsection, we describe in detail the AM resource allocator algorithm and its interfaces Fig. A.7.

Inputs

- *previous actuator allocation*: the result of actuator allocation (refer to AM resource allocator outputs) at the previous time step. It allows the resource allocation algorithm to skip new computation if the previous solution still fully satisfies the requirement at the actual time step.

- *task priority*, defined by the supervisory controller, represents the normalized priority (values between [0, 1]) of each considered task. Disabled tasks have priority 0. To avoid ambiguity in allocating actuators to tasks, the supervisory controller has to ensure that active tasks in each group of parallel or sequential tasks never have the same priority values. It is worth noting that the task priority represents the importance of a task to be executed, however, it does not always indicate the execution order. In fact, the tasks in the parallel controller group are executed in parallel, their priorities help the AM resource allocator to decide how to assign resources to these tasks. Whereas, in the sequential group, the task priorities are used to determine their execution order, from the lowest to the highest priority. Thus the most important sequential task is the last to be executed.

- *actuator states* represent the present state of each actuator including the active signal and the triplet of (actual amplitude, position and type). Specifically, the amplitude gives absolute values in the appropriate unit; the radial position is normalized values (e.g.$\in$ [0, 1]), and if we consider the heating and current drive actuators, the type is the direction of current drive (in the range [-1, 1], negative value is for counter-current, positive value for co-current and value near 0 for heating). These states come from the local actuator control system and diagnostics via the plasma and actuator state reconstruction in the tokamak-dependent layer.

- *actuator limits* give the operational limits (min-max) per actuator, for each state of (amplitude, position and type). They may be ranges of power, position or direction for H&CD actuators, or the range of amplitude for gas valves, the type of gas, etc. Other important information such as the maximum rates of changing an actuator output, delays and bandwidth of the actuator can also be considered. These limits are given by the local actuator control system.

- *resource requests per task* are the requests for actuation resources (again specified by the range of amplitude, position and type) ideally needed to execute each task (as requested by the controllers). In case where the range is not critical for some requests, a full range is given; for example, the β control task is insensitive to whether a H&CD actuator also performs co- or counter-current drive, so the requested type range is $[-1, 1]$. The resource requests per task form the main requirement that the AM resource allocator has to attempt to satisfy within the limits of actuators and the constraints of the global physics goals.

- *parameters for optimal algorithm and task settings* include pre-defined settings for the resource allocation optimization algorithm (such as the cost function weights), as well as some specific settings for tasks (such as preferred or excluded actuators). These parameters come from the pulse schedule via the user interface.

Outputs

- *actuator allocation* represents the optimal choice of assigned virtual resources per task, as determined by the allocation algorithm. The signal specifies which actuators are assigned to each task, and which are not (*true* or *false* respectively). An actuator is only assigned to one task. Some active tasks can receive nothing if there are not enough actuators.

- *virtual resources per task* give the limits of resources allocated to each task (range of power, position and type per task). The virtual resource limits are computed from the actuator allocation solution and the actual limits of the actuators; for example the assigned lower power limit is the minimum of all lower limit powers of assigned actuators to a task, and the assigned upper power limit is the sum of their maximum powers; while the limits of position and type are the largest common range of all assigned actuator limits. The assigned resources must either fully (i.e. the assigned resources covers the maximum of the request) or partially (i.e. the assigned resources only covers the minimum but not the maximum of the request) satisfy the resource requests per task, otherwise, nothing is given.

Algorithm for actuator allocation

The actuator allocation algorithm is the core of the AM resource allocator block. It determines the optimal actuator allocation for all the control tasks while considering the actuator availability and limits. The underlying principles of the algorithm, which is inspired by earlier works [2, 3], are described in Fig. A.8.

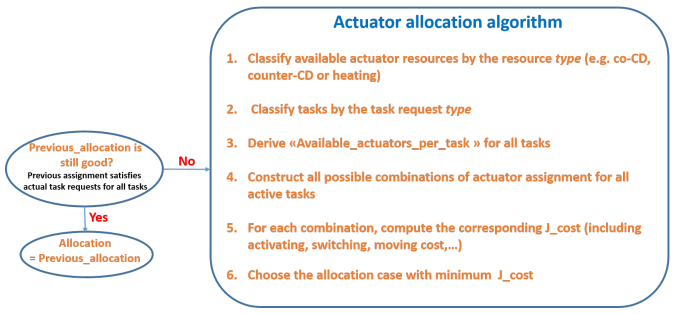
Actuator allocation algorithm
1. Classify available actuator resources by the resource type (e.g. co-CD, counter-CD or heating)
2. Classify tasks by the task request type
3. Derive «Available_actuators_per_task » for all tasks
4. Construct all possible combinations of actuator assignment for all active tasks
5. For each combination, compute the corresponding J_cost (including activating, switching, moving cost,…)
6. Choose the allocation case with minimum J_cost
Previous_allocation is still good?
Previous assignment satisfies actual task requests for all tasks
No
Yes
Allocation = Previous_allocation

Figure A.8: Algorithm for actuator allocation in the AM resource allocator

In the present implementation, a quasi brute-force method is used, whereby the cost function is evaluated for all possible allocation options that partially or fully satisfy the resource requests. After all active tasks have been considered, the best allocation is chosen as the one with the lowest cost. The cost function includes the task priorities, a penalty cost for activating actuators, a penalty cost for switching an actuator from one task to another, and a penalty cost for moving actuators to new positions. Furthermore, a term is added to promote a solution satisfying as many tasks as possible as well as minimizing the number of allocated actuators without deteriorating the quality of control.

The algorithm allows additional requirements for actuators to be specified, such as *preferred actuators* and *excluded actuators* for specific tasks (due to technical constraints or practical purposes). The algorithm removes excluded actuators from the available actuator lists of these relevant tasks and tries to allocate first the preferred actuators if they are still available. These requirements are pre-set by the user. The present algorithm was sufficient for the first tests described in this paper, but we highlight that it can be replaced by e.g. a Mixed Integer Programming algorithm as in [3], which may be more efficient for large numbers of tasks, actuators and constraints.

Appendix A.2. AM interface

The AM interface (Fig. A.9) merges the commands per task into commands to the actuators. This component receives the task priority from the supervisory controller, assigned virtual resources from the AM resource allocator and commands per task from the controllers. It uses this information to map the controller commands from each task (specified as amplitude, position and type) into the commands for each actuator, and send them to the actuator interface in the tokamak-dependent layer. In the previous work [3], this component was referred to as the Low-level Actuator Manager but we have changed this name for increased clarity.

Inputs

Some inputs are the same as the AM resource allocator inputs and outputs (see Appendix A.1), the others are specified hereafter

- *commands per task* contains commands from the controllers, specified in terms of the triplet (amplitude,

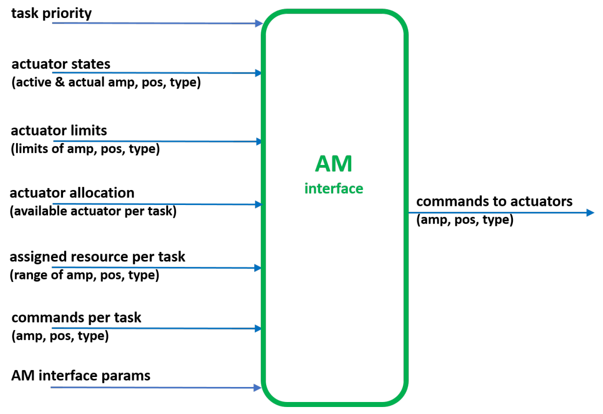

Figure A.9: AM interface inputs and outputs

range, type).

- *AM interface params* include pre-set parameters for AM interface from the pulse schedule.

Output

- *commands to actuators* are the effective amplitude, position and type for each individual actuator. This signal is sent to the actuator interface in order to be converted into the inputs of the actuator systems.

Appendix A.3. Task-to-controller interface

The inputs and outputs of the task-to-controller mapping and controller-to-task mapping components are shown in Fig. 2. These components act as an interface between the controllers and the *task layer*. The task-to-controller mapping functions as a filter to distribute tasks per controller, following the mapping similarly defined in Table 3. On the contrary, an inverse mapping of Table 3 allows the controller-to-task mapping to regroup all controller outputs into task-dependent signals in the correct order of tasks. For example, the outputs of controller 2 are arranged to correspond to the tasks 2 to 5 in Table 3, respectively.

Appendix A.4. Controller interface

The generic interface of the controllers (Fig. A.10) is elaborated in this subsection. This interface allows to integrate various controllers into the proposed scheme in a standardized way and facilitates the addition of new controllers or replacement of existing ones.

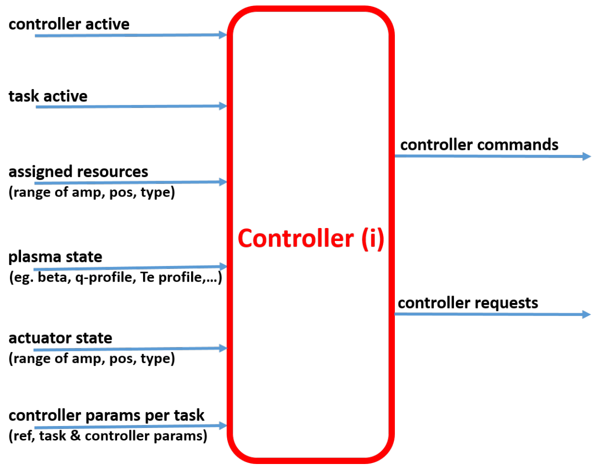
controller active
task active
assigned resources
(range of amp, pos, type)
plasma state
(eg. beta, q-profile, Te profile,...)
actuator state
(range of amp, pos, type)
controller params per task
(ref, task & controller params)
Controller (i)
controller commands
controller requests

Figure A.10: Generic controller inputs and outputs

Inputs

- *controller active* is the active signal for this controller. Note that some controllers may need to run in the background to track some plasma states or actuator states before their tasks are active to avoid a jump of the state errors from zero.

- *task active* is the active signal for the various tasks associated with this controller.

- *virtual resources* are the resources assigned to each task that is handled by this controller. This signal comes from the AM resource allocator via the task-to-controller mapping (see Appendix A.1).

- *plasma state* is the full representation of the plasma state (such as q-profile, β, T_e profile, shape, etc.). This input is necessary to execute the (feedback) control laws. To be generic, all controllers receive the same plasma state information and they can each extract the information they require.

- *actuator state* is the representation of the combination states (e.g. range of amplitude, position and type) of assigned actuators in virtual resources.

- *control params per task* are all parameters of the concerned tasks, including references, and task parameters (such as controller gains per task, etc.). They are defined in the pulse schedule and can be modified in real-time by the supervisory controller based on rules provided by the pulse schedule when necessary.

Outputs

- *controller commands* are the commands of each controller for each task in terms of the resources, within the limit of the assigned virtual resources, represented (per task) by the triplet signals of amplitude, position and type. This is sent to the actuators via the controller-to-task mapping and AM interface.

- *controller requests* are the requests (e.g. range of amplitude, position and type) of each controller for the given tasks, sent to the AM resource allocator (see Appendix A.1) via the controller-to-task mapping for the next time step.

The controller interface in Fig.A.10 is applied to the *parallel* controllers (see controller structure in Sec. 2.2, controller bullet). For the *sequential* controllers, the *sequential* tasks do not need virtual resources specifically dedicated to them, because their commands, which aim to compensate the commands of the other controllers so-that the sequential tasks are achieved, should not be limited by the virtual resources and can also be negative. As a result, the input *assigned resources* is replaced by the *other controller commands* on one hand, and the output *controller requests* is not necessary on the other hand.

Note also that since the task priority defines the execution order of the sequential tasks, the execution order of the controllers is dynamically based on their mapped tasks. The result of the full chain of the sequential tasks (determined by the AM interface after the controller-to-task mapping) is the modification requests of the commands to actuators obtained from the parallel tasks. The AM interface will distribute these modifications (more or less power, gas, etc.) to the actuators of its choice.

Appendix B. Details of the be in H-mode task

It is necessary to discuss further details about the be in H-mode task as an example of sequential tasks since its commands depend on the power commanded of the other controllers. Its mission is to ensure that the loss power P_{Loss} (or the total plasma heating power) is higher than a critical threshold P_{LH}. The threshold power of H-mode depends on the plasma state (e.g. plasma density, plasma size, magnetic field, etc.) and on whether the plasma is currently in H-mode or L-mode. Hence, this value is given to the LH-mode controller as a time varying task parameter by the supervisory controller. In [15] some threshold power estimations for ITER are proposed, however in the demonstrative ITER example in Sec. 4, it is fixed at $P_{LH} = 52MW$. The loss power P_{Loss} has to include the total auxiliary power P_{aux} ($= P_{NBI} + P_{IC} + P_{EC}$), the self-heating power P_α in case of fusion reaction, and the radiation term P_{Brem}. The latter is defined in [32], however it is considered constant in this work. The power P_α can be related to the total auxiliary power P_{aux} using the factor Q [32], such that:

$$\begin{aligned} P_{Loss} &= P_{aux} + P_{\alpha} - P_{Brem} \\ &= \left(1 + \frac{Q}{5}\right) P_{aux} - P_{Brem}. \end{aligned}$$

For the sake of simplicity, we propose a linearized relation between Q and $P_{EC}\,[MW]$ based on different values of HH, the factor represents any confinement improvement or degradation, in Fig. 1 in [32] (three lines corresponding to $HH = 0.75$, 0.85 and 1) for three cases in the Table B.9, such that:

$$Q = \zeta\,(HH)\,P_{EC} + \xi\,(HH)$$

HH	ζ	ξ	
1	-0.14	10	without NTM
0.85	-0.10	7	with 3/2 NTM
0.75	-0.07	4.7	with 2/1 NTM

Table B.9: Parameters for linearized Q factor

In case of two or more NTM appearances at the same time, the lowest Q value is taken into account.

Finally a low-pass filter is applied to P_{Loss} in order to simulate the effect of the energy confinement time on the rate of change of P_{Loss} whenever P_{aux} changes or an NTM appears. The time constant of the filter is considered equal to the expected thermal energy confinement time on ITER ($t_{confinement} = 3.5s$). This has been used to determine the effective power required to remain in H-mode in the ITER tests with and without NTMs (Sec. 4), since the fusion power decreases with the presence of NTMs in particular [32].

Acknowledgment

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

This work was supported in part by the Swiss National Science Foundation. The authors thank W. Treutterer and O.Kudlacek (IPP Garching) for fruitful discussions on this subject.

References

References

[1] D. Humphreys and et al., "Novel aspects of plasma control in ITER," *Physics of Plasmas (1994-present)*, vol. 22 021806, 2015.

[2] C. Rapson and et al., "Actuator management for ECRH at ASDEX Upgrade," *Fusion Engineering and Design*, vol. 96, pp. 694–697, 2015.

[3] E. Maljaars and F. Felici, "Actuator allocation for integrated control in tokamaks: architectural design and a mixed-integer programming algorithm," *Fusion Engineering and Design*, vol. 122, pp. 94–112, 2017.

[4] W. Treutterer and et al., "ASDEX Upgrade discharge control systems real-time plasma control framework," *Fusion Engineering and Design*, vol. 89 (3), pp. 146–154, 2014.

[5] N. Eidietis and et al., "Implementing a finite-state off-normal and fault response system for disruption avoidance in tokamaks," *Nucl. Fusion*, vol. 58 056023, 2018.

[6] N. Ravenel and et al., "Architecture of WEST plasma control system," *Fusion Engineering and Design*, vol. 89 (5), pp. 548–552, 2014.

[7] W. Treutterer and et al., "Towards a preliminary design of the ITER plasma control system architecture," *Fusion Engineering and Design*, vol. 115, pp. 33–38, 2017.

[8] C. Rapson and et al., "Experiments on actuator management and integrated control at ASDEX Upgrade," *Fusion Engineering and Design*, vol. 123, pp. 603–606, 2016.

[9] R. Nouailletas and et al., "Plasma discharge management for long-pulse tokamak operation," *Fusion Science and Technology*, vol. 64(1), pp. 13–28, 2013.

[10] F. Hofmann and et al., “Creation and control of variably shaped plasmas in TCV,” *Plasma Phys. Control. Fusion*, vol. 36 B277, 1994.

[11] F. Felici and et al., “Real-time physics-model-based simulation of the current density profile in tokamak plasmas,” *Nucl. Fusion*, vol. 51, 2011.

[12] ——, “Real-time model-based plasma state estimation, monitoring and integrated control in TCV, ASDEX Upgrade and ITER,” *26th IAEA Fusion Energy Conference, Kyoto, Japan*, 2016.

[13] E. Maljaars and et al., “Simultaneous control of plasma profiles and neoclassical tearing modes with actuator management in tokamaks,” *42nd European Physical Society Conference on Plasma Physics, EPS*, 2015.

[14] T. Blanken and et al., “Real-time plasma state monitoring and supervisory control on TCV,” *Nucl. Fusion*, vol. 59 (2), 2019.

[15] Y. Martin and et al., “Power requirement for accessing the H-mode in ITER,” *J. Phys.: Conf. Ser. 123 012033*, 2008.

[16] M. Kong and et al., “Real-time control of neoclassical tearing modes and its integration with multiple controllers in the TCV tokamak,” *44 th EPS Conference on Plasma Physics, Belfast, Northern Ireland*, 2017.

[17] K. Nagasaki and et al., “Stabilization of neoclassical tearing mode by eccd and its evolution simulation on JT-60U tokamak,” *Nucl. Fusion*, vol. 45 1608, 2005.

[18] M. Kong and et al., “Control of NTMs and integrated multi-actuator plasma control on TCV,” *Nucl. Fusion*, vol. 59 076035, 2019.

[19] P. Martin and et al., “Physics, control and mitigation of disruptions and runaway electrons in the EUROfusion Medium Size Tokamaks science programme,” *26th IAEA Fusion Energy Conference Kyoto, Japan*, 2016.

[20] MATLAB and I. N. M. U. S. Statistics Toolbox Release 2012b, The MathWorks.

[21] C. Galperti and et al., “Integration of a real-time node for magnetic perturbations signal analysis in the Distributed Digital Control System of the TCV Tokamak,” *20th Real Time Conference, Padova, Italy*, 2016.

[22] F. Felici and et al., “Distributed digital real-time control system for TCV tokamak,” *Fusion Engineering and Design*, vol. 89, pp. 155–164, 2014.

[23] D. Kim and et al., “An active feedback plasma profile control approach applied to tcv plasmas & perspectives toward iter,” *43nd European Physical Society Conference on Plasma Physics, EPS*, 2016.

[24] H. Reimerdes and et al., “From current-driven to neoclassically driven tearing modes,” *Phys. Rev. Lett.*, vol. 88 105005, 2002.

[25] J.-M. Moret and et al., “Tokamak equilibrium reconstruction code liuqe and its real time implementation,” *Fusion Engineering and Design*, vol. 91, pp. 1–15, 2015.

[26] E. Poli, A. Peeters, and G. Pereverzev, “Torbeam, a beam tracing code for electron-cyclotron waves in tokamak plasmas,” *Computer Physics Communications*, vol. 136, pp. 90–104, 2001.

[27] M. Singh and et al., “Status of heating and current drive systems planned for ITER,” *IEEE Transactions on Plasma Science*, vol. 44 (9), p. 1514–1524, 2016.

[28] O. Sauter and et al., “Control of neoclassical tearing modes by sawtooth control,” *Physical review letters*, vol. 88 105001, no. 10, 2002.

[29] I. Chapman and et al., “Empirical scaling of sawtooth period for onset of neoclassical tearing modes,” *Nucl. Fusion*, vol. 50 102001, 2010.

[30] M. Henderson and et al., “The targeted heating and current drive applications for the ITER electron cyclotron system,” *Physics of Plasmas*, vol. 22 021808, 2015.

[31] D. Humphreys and et al., “Active control for stabilization of neoclassical tearing modes,” *Physics of Plasma*, vol. 13 056113, 2006.

[32] O. Sauter, M. Henderson, G. Ramponi, and C. Zucca, “On the requirements to control neoclassical tearing modes in burning plasmas,” *Plasma Phys. Control. Fusion*, vol. 52 02502, 2010.