Fusion Engineering and Design 168 (2021) 112412

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

PETRA: A generalised real-time event detection platform at JET for disruption prediction, avoidance and mitigation

C.I. Stuart [a,*], G. Artaserse [b], P. Card [a], I.S. Carvalho [c], R. Felton [a], S.N. Gerasimov [a], A. Goodyear [a], R.B. Henriques [c], D. Karkinsky [a,d], P.J. Lomas [a], P. McCullen [a], F. Rimini [a], A. V. Stephen [a], D.F. Valcárcel [a], J. Waterhouse [a], M. Wheatley [a]

[a] UK Atomic Energy Authority, Culham Science Centre, Abingdon OX14 3DB, UK
[b] ENEA, Via E. Fermi 45, 00044 Frascati, Roma, Italy
[c] Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, P-1049-001 Lisboa, Portugal
[d] ITER Organization, Route de Vinon-sur-Verdon, CS 90 046, 13067 St Paul Lez Durance Cedex, France

ARTICLE INFO

Keywords:
Exception handling
Event detection
Real-time software
Fusion plasmas
MARTe
JET

ABSTRACT

Plasma Event TRiggering for Alarms (PETRA) is a flexible new system at JET for detecting plasma events to trigger exception handling responses. It provides a platform for all existing and new techniques which involve processing JET real-time data to identify when to change the trajectory of a plasma shot, and reduces the technical cost and risk in demonstrating new techniques. At the same time as providing a protection function to the machine, it still supports additions and modifications to the event detector library during the course of an experimental campaign. It has successfully demonstrated novel techniques, and has computational capacity to continue supporting research into further event detection triggers.

1. Introduction

Experimentation on a tokamak device naturally involves exploring the bounds of its operational region, and uncertainty in those bounds means that it is often necessary to adjust the trajectory of a plasma shot away from the pre-programmed plan in real-time. This is known as exception handling (EH). EH is required for: the triggering of protection systems for investment protection; reducing scarce resource consumption in failing shots; correcting unhealthy plasma state to maintain performance; handling hardware failures during operations; accounting for unpredictable machine response in the presence of disturbances such as impurity influxes or actuator failures.

To handle exceptions, first we need to have means of detecting them. There are also benefits to experiment design in detecting certain non-exceptional plasma conditions, for example to trigger the start of a termination phase based on plasma radiative properties, rather than at a pre-programmed time.

Exception handling is often part of a scheme of protecting the machine. Fusion devices tend to apply many layers of protective systems, with differing aims ranging from the safety of personnel down to the longevity of individual components.

The requirements for these protective systems vary, often with a distinction between those which *aim* to operate the machine in a safe way, and those which *guarantee* an aspect of operational safety. For example at JET, the Plasma Position and Current Controller (PPCC) [1] is set up to hold the plasma at a safe distance from vessel walls, but does not guarantee this. Meanwhile the WAll Load Limitation System (WALLS) [2] and Vessel Thermal Map system (VTM) [3] ensure that temperature and energy limits on plasma facing components (PFCs) are maintained by raising alarms that terminate the shot when they observe these limits have been or will be surpassed.

Considering EH specifically in respect of plasma conditions, the avoidance and mitigation of disruptions has long been a requirement in tokamaks running larger energy plasmas [4,and references therein], to avoid high disruption forces and rapid deposition of large thermal energies onto PFCs and the vessel.

Beyond disruption mitigation, recent efforts in the field of disruption prediction [5–11] aim to produce earlier and more reliable means of identifying conditions that lead to disruptions in real-time. These are then used to trigger disruption avoidance responses, either attempting to land a plasma early without the disruption occurring or predicting the disruption sufficiently far in advance to adjust trajectory and maintain a

* Corresponding author.
E-mail address: Chris.Stuart@ukaea.uk (C.I. Stuart).

https://doi.org/10.1016/j.fusengdes.2021.112412
Received 27 November 2020; Received in revised form 1 February 2021; Accepted 24 February 2021
Available online 2 March 2021

healthy plasma.

On JET up to 2018, real-time implementations for *disruption mitigation* event detection would be found in VME systems running either compiled schemes, or programmable schemes in the Real Time Central Controller (RTCC) [12]. Indicators for *disruption avoidance* may also have been programmed with RTCC, or for more demanding machine learning-based schemes a dedicated Linux PC system would be introduced, with considerable software and hardware integration costs.

The problems solved by these systems can be generalised into the concept of a plasma event detector. Introducing a single common technology for plasma event detectors has operational benefits in providing a consistent approach to configuring exception handling regimes. Meanwhile, it can bring down the costs of introducing new plasma event detectors, both in the initial integration and the ongoing maintenance of these schemes. This reduced barrier to entry and cost of failure allows us to implement more experimental schemes.

The Plasma Event TRiggering and Alarms system (PETRA) introduced in 2018 is a platform for new and existing plasma event detectors. It provides a framework for the software design of event detectors, access to a wide range of data from the JET real-time data network (RTDN) [13] and the ability to connect event detector outputs to exception handling responses in the Real Time Protection Sequencer (RTPS) [14] and RTCC. It provides a protection function in triggering the mitigation of damage from disruptions, while at the same time enabling the use and development of certain non-protection event detection functions throughout experimental campaigns.

2. Requirements

PETRA was designed to meet the following requirements.

- Maintain a processing cycle with a hard real-time deadline of 2 ms, to match the performance of current disruption mitigation triggering systems and maintain access to the highest resolution of data available in real-time.
- Process input data to detect a variety of plasma conditions classified as significant events.
- Transmit alarms to RTPS and RTCC to enable the trajectory of a pulse to be modified in response to detected events.
- Make use of a pre-existing PC setup identical to WALLS, powered by an AMD Phenom II X6 1090T CPU and running a Linux distribution with the Linux 2.6 PREEMPT kernel.
- Enable the safe handling by RTPS and RTCC of missing data inputs to PETRA.
- Accommodate present and future data processing techniques to enable continued development of additional event detectors.
- Enable regular minor updates and additions to the library of event detectors.
- Operate in the context of the typical JET operations schedule, with long experimental campaigns separated by shutdown phases of varying lengths and short restart phases.
- Store all data required for the purposes of determining how PETRA behaved during a pulse and for later analysis of the behaviour and performance of event detectors.

3. Design

PETRA re-purposes parts of an existing system at JET. Discussion of the hardware, operating system and integration into the JET control and data acquisition environment will not be repeated in this paper, as these inherited parts of the system are based on the design of WALLS, which is discussed in detail by Valcárcel et al. [2].

As with many JET real-time systems [2,3,14,15], PETRA makes use of the MARTe framework [16]. Generic and specialised modules for taking in data inputs and providing data outputs are written in C++, inheriting from MARTe classes. Using MARTe configuration language,

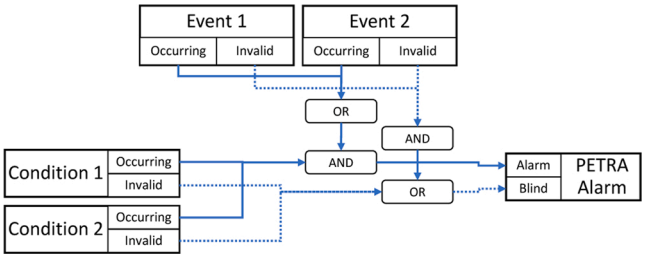

Fig. 1. Each PETRA alarm and blind is a logical combination of the connected events and conditions.

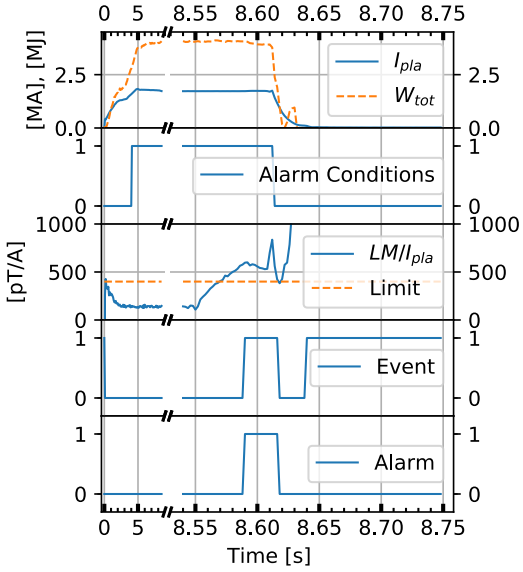

Fig. 2. In this JET restart pulse #93912, a locked mode detector is connected to one PETRA alarm. The conditioning inputs to the alarm were 1 = "time > 0 s" and 2 = "plasma current > 1.6 MA OR plasma energy > 5 MJ", which are both met from 4.058 s. The locked mode event detector observes the locked mode amplitude normalised by plasma current exceeding a threshold at 4.570 s. After a 20 ms assertion time, it outputs that an event is occurring. Since the conditions are met and an event is occurring, the alarm is raised, and PETRA thereby triggers a stop and MGI. The pulse rapidly concludes, during which the normalised locked mode amplitude briefly dips below the threshold but then, as plasma current reduces so the normalised locked mode measure increases towards infinity. After another 20 ms assertion time the detector once more outputs an event occurring. However, the alarm is not raised again because the conditioning inputs are no longer both met.

instances of these modules are then configured and connected together to build an application. A new event detector could be built using existing modules configured and connected in a new way in some cases, or by writing and compiling new modules and configuration in others.

Data is received over the RTDN in data packets from numerous diagnostics and control systems, and transmitted back by the same means.

Data feeds into specialised modules for different event detectors, but these all produce a standardised event detector two-part output: is this event occurring, and has this event received enough data to be valid? This output is structured to fit into a single word in the application data buffer. Since received data are usually only useful if they are recent, PETRA checks that these packets are received at the expected rate, and the event detector output indicates this.

Fig. 1 indicates how the outputs of event detectors are fed into PETRA alarms. Grouping events together into alarms is necessary because, while PETRA must be able to process a number of event detectors which varies on a shot-by-shot basis, the systems it communicates with are not expected to take in a varying number of PETRA

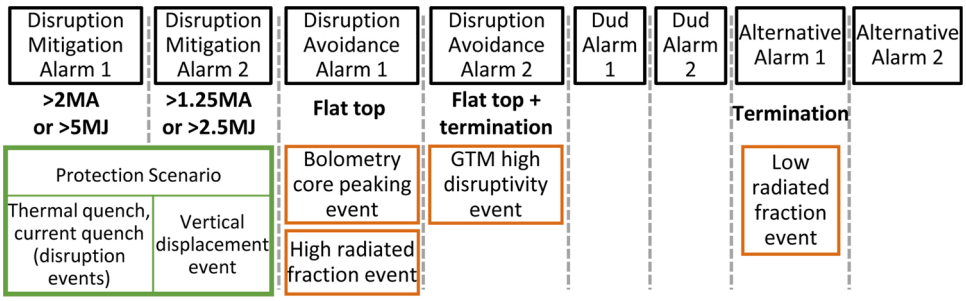

Fig. 3. A typical pulse allocates event detectors to the various PETRA alarms to target different exceptions in different phases of the plasma.

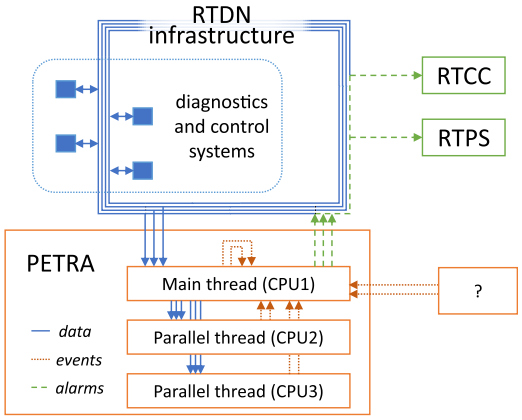

Fig. 4. All data input to PETRA comes from the RTDN and goes through a single main thread, which synchronises and shares input data with two parallel threads. These threads send event outputs back to the main thread, which combines these with its own events and produces the outputs. It sends outputs back over the RTDN. With minimal additional work, events generated externally from PETRA could be sent to the main thread over the RTDN and treated in the same manner as those generated on the parallel threads.

outputs. The event detectors therefore feed into a set number of PETRA alarms, each with an accompanying blind alarm. The alarms also take in 'conditioning' inputs. When the conditions are not all met, an alarm will ignore its 'event occurring' inputs, so that specific alarms can be applied only at certain times or above certain plasma currents or energies. These conditioning inputs have been implemented as event detectors.

A practical example is provided in Fig. 2, showing the trajectory of a plasma pulse which was stopped by PETRA due to a locked mode.

There are 8 such PETRA alarms in the present configuration, leading to typical usage as in Fig. 3. The link between these 8 alarms and exception handling responses is handled within the receiving systems, RTPS and RTCC.

Data collection in PETRA requires an additional JET service on top of those used by WALLS, as a result of the ability of PETRA to run any number of plasma event detectors, each of which may define its own additional dataset. The General Acquisition Program (GAP) is not designed to handle any arbitrary dataset, so another JET module known as the Real Time Logging library was adapted for MARTe applications. This module, as used by RTCC, stores data in a file system with a single file per signal and an index file for signal discovery. The directory containing these files is transmitted as a whole to the JET data warehouse for storage after each pulse, and the common JET software libraries for data retrieval can be used to later access this data.

PETRA runs multiple synchronised threads to allow parallel calculations on the input data (Fig. 4). One main thread is responsible for synchronisation with other JET systems through a timing packet received on the RTDN. This thread handles all input and output from PETRA. It shares the input data with the parallel threads and triggers their execution. A standardised data format is used to move event detector outputs generated in the other threads back to the main thread, where all event detector outputs are then used as inputs to the PETRA alarms.

The same standardised data format could be used to send event detector outputs to PETRA over the RTDN. This would allow a future additional system to run the PETRA framework and feed into the existing infrastructure over RTDN packets alone, in order to enable event detectors which require greater CPU power or potentially GPU processing.

3.1. *Designing for both protection and experimentation*

PETRA is a protection system which makes guarantees about a subset of its event detectors. These will be used to trigger massive gas injection (MGI), which must be achieved on known timescales to effectively mitigate damage caused during certain plasma events. However, it also needs to accommodate new or updated event detectors to facilitate experimentation. It is a requirement that new or updated event detectors will not change PETRA's behaviour (what decisions it makes) or performance (its hard real-time performance) with regard to those guarantees. Any changes to PETRA that may interfere with this requirement invalidate the use of PETRA to provide disruption mitigation until a commissioning procedure has been completed which once again proves its behaviour and performance.

This full commissioning procedure requires a number of shots on JET, and observation of plasma conditions which trigger its disruption event detectors and its vertical displacement event detector. Typically this commissioning would be carried out parasitically during a restart phase of operations, where PETRA's guarantees are not yet needed because of the low plasma currents and energies of the shots being run. Outside of a restart phase, repeating this commissioning only for PETRA is generally prohibited, as it would take time away from the scientific programme of the machine.

Prohibiting all changes to PETRA's event detectors was not an option though, as this would hinder progress in the development of new, experimental schemes through PETRA.

PETRA is inherently reconfigurable, as it is built in the data-driven MARTe framework. From a configuration stream sent to PETRA, it will instantiate a set of Generic Application Modules (GAMs), which read in data signals, execute some algorithm on that data and write out data signals. GAMs are the application building blocks in the MARTe framework. They each have a section of configuration parameters fed to them, and their input and output signals are connected together into a network of data flows which carries out the application's purpose. Many small changes in application behaviour can be achieved by changing the parameters of GAMs or by modifying the network connections, while still only using the same proven pieces of application code.

However, there will still be times when newly compiled code is needed to update an event detector. For this reason, all compiled code which enables the protection elements of PETRA sits in two distinct software products (one for the event detectors and one for the PETRA

framework that they sit inside), while all compiled code for the other event detectors sits in a third. Furthermore, the PETRA application separates out its protection role to run on one main real-time thread (see Fig. 4), with other event detectors running inside either one or two other real-time threads. All of these threads run on different isolated CPU cores. If the non-protection threads fail to meet their real-time deadline, this will not prevent the protection thread from meeting its own deadline.

With these design choices, it becomes safe to make software modifications to this third software product without recommissioning the entire application. Instead, a single shot recommissioning procedure is used, in which all the new code is exercised in the non-protection threads and the impact (or lack thereof) on the performance of the protection thread is monitored.

3.2. Shot-by-shot configuration

A new configuration stream is sent to PETRA before every JET shot. The majority of this configuration is static, but the choice and configuration parameters of event detectors running on each thread will vary, as will the connections from event detectors to alarms, and the alarm conditioning parameters.

The JET user chooses a single Protection Scenario for a shot, which implements a number of event detectors attaching them to the two PETRA alarms reserved for triggering MGI. The parameters of these detectors are not adjustable, as they correspond to limits specified in the operating instructions of the JET device. Protection Scenarios are curated by the responsible officer for PETRA, who could produce a Protection Scenario that goes against these limits where such exceptions have been approved for a specific experiment. The full protection scenario runs on the main PETRA thread.

Next, the JET user has access to a number of Experiment Event Scenarios, each with configurable parameters and a configurable connection to any of the other six PETRA alarms. These will run on the other real-time threads.

Finally, the JET user can configure the conditioning on the eight PETRA alarms, within bounds chosen to guarantee its protection functions.

4. PETRA usage

At time of writing, PETRA has been used as the sole disruption mitigation trigger in all 2155 JET shots for which it was fully commissioned. It is the only system configured to trigger the MGI system at JET in most shots; all the pre-2018 plasma event detectors for disruption mitigation have been reimplemented and are found in the default Protection Scenario.

PETRA has also introduced a new detector for triggering MGI at vertical displacement events (VDEs), which prior to 2018 were not protected against. Furthermore, PETRA now hosts many well-established event detectors which were previously implemented in RTCC, freeing up this programmable, hardware-limited system for other purposes.

PETRA has supported the implementation and development of novel detectors based on various works [17–19], the details of which are not covered in this paper. Mostly these are now available for regular use, but novel detectors are not uniformly successful, as is expected when supporting active research. In particular, PETRA demonstrated the use of generative topographic mappings (GTM) in real time on JET [11]. While the technique has been shown to be effective in offline data, the differing quality of the input data available in real time versus that used to build the model offline was such that the real-time output was unreliable. Now that the plasma shots planned in upcoming experimental campaigns have already been designed without GTM for exception handling, further work on improving this technique at JET is presently less of a priority.

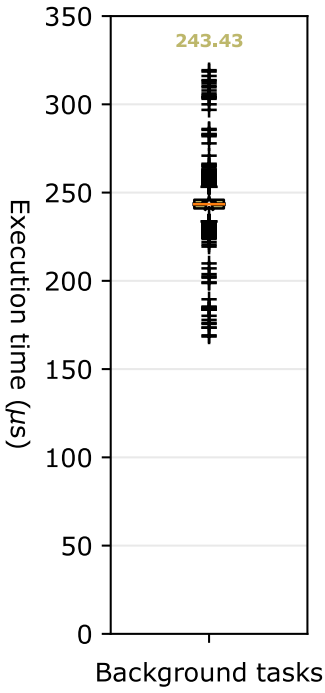

Fig. 5. Background activities in the real-time threads usually use around 250 μs in each cycle.

Had GTM been implemented on its own dedicated system, there would now be redundant hardware and software remaining, which would not be used and would soon become unsupportable. However, because it was implemented in PETRA, there is no redundant hardware, and the software has been developed in a standardised way with the PETRA framework. That framework is in continued use in the other event detectors, thus a future update of GTM in PETRA will continue to be technically cheap to implement, whether that happens in the short term or in the longer term.

With the currently available event detectors, PETRA is not close to using its full processing capacity, and therefore not close to missing its real-time deadlines. Fig. 5 shows that in the most recent recommissioning pulse, the tasks other than event detectors in each real-time thread require up to only ~320 μs in each cycle. Meanwhile, Fig. 6 shows the distribution of execution times of various pieces of PETRA event detectors during the same pulse. Some of these pieces are data preprocessing related to specific event detectors, while others are the event detectors themselves. The GTM event detector is the most demanding task PETRA currently completes, and has been plotted on a separate scale for this reason. However, even this task leaves plenty of spare processing capacity within the 2 ms cycle time, even without making use of multiple threads.

5. Conclusions and future work

Introducing a dedicated plasma event detection system at JET has consolidated existing triggers used in exception handling and facilitated cheaper development of new triggers. The use of PETRA as both a protection and an experimentation system has been a successful strategy, but must be supported by careful control of its configuration and sufficient separation in the software design between these two functions.

At present, PETRA still has significant capacity to run more instances of the existing event detectors, or more computationally demanding event detectors. Future work involving computational complexity orders of magnitude greater could be accommodated by reducing the cycle time of the third real-time thread, or if a faster deadline is required then an implementation on separate hardware making use of recent CPUs, GPUs

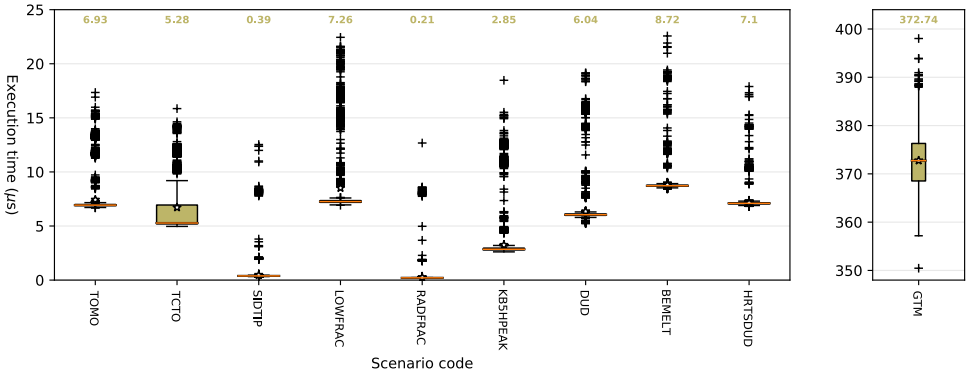

Fig. 6. The mean time taken to execute various tasks in a recent commissioning pulse is shown above a plot of the distribution. The specifics of each task are not the focus of this paper; for details see cited works where available. Key: TOMO = tomography reconstruction [18]; TCTO = tomography core-to-outboard ratio event detector; SIDTIP = centroid method event detector [19], LOWFRAC = low radiated fraction event detector; RADFRAC = high radiated fraction event detector; KB5HPEAK = bolometery peaking event detector; DUD = dud detector [17]; BEMELT = beryllium melting event detector; HRTSDUD = Thomson scattering failure detector; GTM = GTM processing [11]gr6

or even FPGAs could feed into the main PETRA infrastructure via the RTDN.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgements

This work was funded by the RCUK Energy Programme [Grant number EP/P012450/1]. Contract for the operation of the JET facilities co-funded by Euratom. This work has been carried out within the framework of the Contract for the Operation of the JET Facilities and has received funding from the European Union's Horizon 2020 research and innovation programme. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] A. Cenedese, F. Sartori, Plasma position and current control management at JET, Proceedings of the IEEE Conference on Decision and Control, vol. 5 (2003) 4628–4633, https://doi.org/10.1109/cdc.2003.1272293.

[2] D.F. Valcárcel, D. Alves, P. Card, B.B. Carvalho, S. Devaux, R. Felton, A. Goodyear, P.J. Lomas, F. Maviglia, P. McCullen, C. Reux, F. Rimini, A. Stephen, L. Zabeo, K. D. Zastrow, JET EFDA Contributors, The JET real-time plasma-wall load monitoring system, Fusion Eng. Des. (2014), https://doi.org/10.1016/j.fusengdes.2013.10.010.

[3] D. Alves, R. Felton, S. Jachmich, P. Lomas, P. McCullen, A. Neto, D.F. Valcárcel, G. Arnoux, P. Card, S. Devaux, A. Goodyear, D. Kinna, A. Stephen, K.D. Zastrow, JET Contributors, Vessel thermal map real-time system for the JET tokamak, Phys. Rev. Spec. Top. Acceler. Beams (2012), https://doi.org/10.1103/PhysRevSTAB.15.054701.

[4] S.N. Gerasimov, P. Abreu, G. Artaserse, M. Baruzzo, P. Buratti, I.S. Carvalho, I. H. Coffey, E. De La Luna, T.C. Hender, R.B. Henriques, R. Felton, S. Jachmich, U. Kruezi, P.J. Lomas, P. McCullen, M. Maslov, E. Matveeva, S. Moradi, L. Piron, F. G. Rimini, W. Schippers, C. Stuart, G. Szepesi, M. Tsalas, D. Valcarcel, L. E. Zakharov, Overview of disruptions with JET-ILW, Nucl. Fusion (2020), https://doi.org/10.1088/1741-4326/ab87b0.

[5] C.G. Windsor, G. Pautasso, C. Tichmann, R.J. Buttery, T.C. Hender, JET EFDA Contributors, the ASDEX Upgrade Team, A cross-tokamak neural network disruption predictor for the JET and ASDEX Upgrade tokamaks, Nucl. Fusion (2005), https://doi.org/10.1088/0029-5515/45/5/004.

[6] G.A. Rattá, J. Vega, A. Murari, G. Vagliasindi, M.F. Johnson, P.C. De Vries, JET EFDA Contributors, An advanced disruption predictor for JET tested in a simulated real-time environment, Nucl. Fusion (2010), https://doi.org/10.1088/0029-5515/50/2/025005.

[7] R. Aledda, B. Cannas, A. Fanni, A. Pau, G. Sias, the ASDEX Upgrade Team, Improvements in disruption prediction at ASDEX Upgrade, Fusion Eng. Des. (2015), https://doi.org/10.1016/j.fusengdes.2015.03.045.

[8] R. Moreno, J. Vega, S. Dormido-Canto, A. Pereira, A. Murari, JET Contributors, Disruption prediction on JET during the ILW experimental campaigns, Fusion Sci. Technol. (2016), https://doi.org/10.13182/FST15-167.

[9] C. Rea, R.S. Granetz, Exploratory machine learning studies for disruption prediction using large databases on DIII-D, Fusion Sci. Technol. (2018), https://doi.org/10.1080/15361055.2017.1407206.

[10] J. Kates-Harbeck, A. Svyatkovskiy, W. Tang, Predicting disruptive instabilities in controlled fusion plasmas through deep learning, Nature (2019), https://doi.org/10.1038/s41586-019-1116-4.

[11] A. Pau, A. Fanni, S. Carcangiu, B. Cannas, G. Sias, A. Murari, F. Rimini, the JET Contributors, A machine learning approach based on generative topographic mapping for disruption prevention and avoidance at JET, Nucl. Fusion (2019), https://doi.org/10.1088/1741-4326/ab2ea9.

[12] R. Felton, E. Joffrin, A. Murari, L. Zabeo, F. Sartori, F. Piccolo, J. Farthing, T. Budd, S. Dorling, P. McCullen, J. Harling, S. Dalley, A. Goodyear, A. Stephen, P. Card, M. Bright, R. Lucock, E. Jones, S. Griph, C. Hogben, M. Beldishevski, M. Buckley, J. Davis, I. Young, O. Hemming, M. Wheatley, P. Heesterman, G. Lloyd, M. Walters, R. Bridge, H. Leggate, D. Howell, K.D. Zastrow, C. Giroud, I. Coffey, N. Hawkes, M. Stamp, R. Barnsley, T. Edlington, K. Guenther, C. Gowers, S. Popovichef, A. Huber, C. Ingesson, D. Mazon, D. Moreau, D. Alves, J. Sousa, M. Riva, O. Barana, T. Bolzonella, M. Valisa, P. Innocente, M. Zerbini, K. Bosak, J. Blum, E. Vitale, F. Crisanti, E. De La Luna, J. Sanchez, EFDA-JET Contributors, Real-time measurement and control at JET experiment control, Fusion Eng. Des. (2005), https://doi.org/10.1016/j.fusengdes.2005.06.286.

[13] R. Felton, K. Blackler, S. Dorling, A. Goodyear, O. Hemming, P. Knight, M. Lennholm, F. Milani, F. Sartori, I. Young, Real-Time plasma control at JET using an ATM network, SANTA FE 1999-11th IEEE NPSS Real Time Conference, Conference Record, RT 1999 (1999), https://doi.org/10.1109/RTCON.1999.842597.

[14] J.S. Edwards, I.S. Carvalho, R. Felton, C. Hogben, D. Karkinsky, P.J. Lomas, P. A. McCullen, F.G. Rimini, A.V. Stephen, Robust configuration of the JET Real-Time Protection Sequencer, Fusion Eng. Des. 146 (2019) 277–280, https://doi.org/10.1016/j.fusengdes.2018.12.045.

[15] F.G. Rimini, F. Crisanti, R. Albanese, G. Ambrosino, M. Ariola, G. Artaserse, T. Bellizio, V. Coccorese, G. De Tommasi, P. De Vries, P.J. Lomas, F. Maviglia, A. Neto, I. Nunes, A. Pironti, G. Ramogida, F. Sartori, S.R. Shaw, M. Tsalas, R. Vitelli, L. Zabeo, JET EFDA Contributors, First plasma operation of the enhanced JET vertical stabilisation system, Fusion Eng. Des. 86 (2011) 539–543, https://doi.org/10.1016/j.fusengdes.2011.03.122.

[16] A.C. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, A. Barbalace, H. Fernandes, D.F. Valcárcel, A.J. Batista, MARTe: a multiplatform real-time framework, IEEE Trans. Nucl. Sci. 57 (2010) 479–486, https://doi.org/10.1109/TNS.2009.2037815.

[17] L. Piron, C. Challis, R. Felton, D. King, M. Lennholm, P. Lomas, C. Piron, F. Rimini, D. Valcarcel, JET Contributors, The dud detector: An empirically-based real-time algorithm to save neutron and T budgets during JET DT operation, Fusion Eng. Des. (2019), https://doi.org/10.1016/j.fusengdes.2019.02.077.

[18] D.R. Ferreira, P.J. Carvalho, H. Fernandes, Deep learning for plasma tomography and disruption prediction from bolometer data, IEEE Trans. Plasma Sci. (2020), https://doi.org/10.1109/TPS.2019.2947304.

[19] J. Vega, A. Murari, S. Dormido-Canto, F. Hernández, T. Cruz, D. Gadariya, G. A. Rattá, JET Contributors, A linear equation based on signal increments to predict disruptive behaviours and the time to disruption on jet, Nucl. Fusion 60 (2020), https://doi.org/10.1088/1741-4326/ab5880.