Contents lists available at ScienceDirect

Fusion Engineering and Design

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

Reducing the user burden when running MELCOR for accident analysis for a tokamak

G. Karajgikar, J.J. Nebrensky [*]

United Kingdom Atomic Energy Authority, Culham Science Centre, Abingdon-On-Thames, OX14 3DB, UK

ARTICLE INFO

Keywords:
Fusion
Accident
Modelling
MELCOR
Safety
Tritium

ABSTRACT

MELCOR is a software tool for accident analysis with a long history in the fission sector. A model has been developed for use in MELCOR, using proposed layouts of EU DEMO and assumptions based on current design choices, to facilitate safety design and development for fusion power plants.

There seems however to be a misconception that MELCOR is difficult to run and that the tasks must be executed sequentially. Accident analysis also requires reruns of multiple slightly different scripts. We have streamlined the running of MELCOR tasks to allow simultaneous execution on a computing cluster. The input decks are version controlled within UKAEA's internal GitLab repository, and shell scripts wrap the separate MELCOR invocations, embed specific job parameters, and keep separate the inputs and outputs of multiple separate instances.

We illustrate the streamlined running using two examples: a matrix representing potential responses to an incident by different valves, and a parameter sweep across a range of response times for a single valve. The new technique is shown to be a much faster method of carrying out sensitivity studies and allowing further investigations into specific areas of concern in more detail.

1. Introduction

Fusion power is a critical part of the future energy landscape yet is all too often depicted with the focus on the fusion reaction itself and little attention on supporting systems. In practice of course a real power plant will have the tokamak surrounded by a plethora of ancillaries: the fuel cycle, cooling systems, diagnostics, radiation management, controls, and so on. Understanding the consequences and mitigation for possible accidents is crucial for public acceptance but requires navigating this complex web of support systems.

MELCOR [1,2] is a software tool for accident analysis with a long history in the fission sector. It is a two-step process: first MELGEN reads and parses the text-format input deck, creates the required data structures and checks for consistency with the active packages, and writes out a representation of the model in its initial state. MELCOR itself then carries out the actual simulation, propagating the initial state forward through time and dumping the current state to a "plotfile" at in-simulation intervals defined in the input deck.

A model has been developed for use with MELCOR 1.8.5 and 1.8.6 For Fusion, using proposed layouts of EU DEMOnstration Powerplant (DEMO) and assumptions based on current design choices, to facilitate safety design and development.

Among the issues we encountered whilst using MELCOR were, firstly, that the input deck mark-up format — as can be seen in [2] — is not intuitive to read and check which makes it difficult to spot (possibly unintended) changes to the input parameters in the code. Secondly, within the MELCOR user community there is an acknowledgement that MELCOR can be difficult to run and that the tasks should preferably be executed sequentially.

Our interest was in studying tritium release following a break in the pumping line of a torus exhaust circulation loop, including a study of the effect of responses by safety relevant valves at various locations around the system. Comparison of actuating a valve in one or another of two locations requires multiple changes to the input deck: first the valve in the required location must be activated and the previous valve disabled, and then to prevent various output and log files from being overwritten either eight separate output filenames re-defined, or the user must manage the separation/collation of the different tasks in different sub-directories.

To expedite running the models we have implemented a simple

* Corresponding author.
E-mail address: Henry.Nebrensky@physics.org (J.J. Nebrensky).

https://doi.org/10.1016/j.fusengdes.2024.114399
Received 13 October 2023; Received in revised form 15 March 2024; Accepted 25 March 2024
Available online 5 April 2024

framework atop the Sun Grid Engine (SGE) batch system [3] to allow simultaneous execution of MELCOR tasks on a computing cluster. The input decks are version controlled within UKAEA's internal GitLab repository, and shell scripts wrap the separate MELGEN and MELCOR invocations, embed specific job parameters, and keep separate the inputs and outputs of multiple individual instances. Parameter sweeps are enabled by the Grid Engine batch system's Array Job feature.

We aim to resolve the issues mentioned by implementing our shell scripts; and illustrate the improved workflow with two examples: the aforementioned study on the effect of various valve locations, and a 22-point parameter sweep looking at the effect of varying delays in valve closure times on the amount of tritium released from primary pipework. Section 2 below describes the accident scenario and representation in MELCOR; Section 3 our workflow and Section 4 the example outcomes.

At the time of writing our MELCOR model was still in development, and we stress that the results are presented here to demonstrate the workflow, and not intended to be representative of accident consequences for DEMO.

2. Example accident scenario

A fusion fuel cycle is typically a tritium-deuterium mixture fed into the tokamak, followed by purification and subsequent re-injection of the exhaust gas stream. This cycle forms a continuous loop driven by a tritium plant installed in an adjacent building.

Our intention was to investigate the consequences of a Postulated Initiating Event (PIE) of a break in the torus pumping line that takes the tokamak exhaust to the tritium plant for processing. This particular PIE is derived from Scenario INTL-1, PIE TP-2 in Shaw et al. [4] where it is denoted a Design Basis Accident. Fig. 1 is a concept illustration of the DEMO tokamak building. The location of the break is the lower pipe chase (labelled B3).

The PIE is a break in the pumping line between the torus (labelled A in Fig. 2) and tritium processing plant (F). As the pumping line is below atmospheric pressure the air from the lower pipe chase (B) flows into the primary pipework filling the line, up until the next closed valve, until it too reaches atmospheric pressure. Once the pipeline reaches equilibrium with respect to the lower pipe chase, tritium begins to escape into the room. The tritium can then leak through the internal wall into the lower feeder and tank level, then through the external wall into the tritium building (F), and finally out of the tritium building into the environment. This overall accident progression is summarised in Fig. 3.

Some of the possible questions include the best locations for safety relevant valves and how quickly they should respond; how pumps should react, and then the interdependencies between these factors and so on. These require re-running similar cases, with slight changes in multiple locations in the input deck that can readily be checked for correctness.

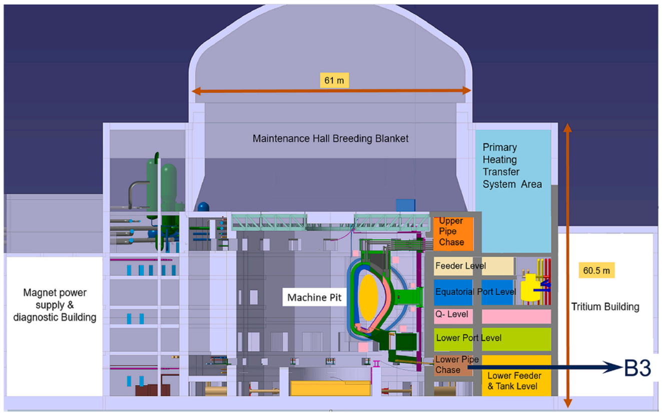
61 m
Maintenance Hall Breeding Blanket
Primary Heating Transfer System Area
Upper Pipe Chase
Feeder Level
Equatorial Port Level
Q- Level
Lower Port Level
Lower Pipe Chase
Lower Feeder & Tank Level
60.5 m
Tritium Building
Magnet power supply & diagnostic Building
Machine Pit
B3

Fig. 1. Concept design of the DEMO tokamak building from Gliss et al. (2022) [5]. The break in the pumping line occurs in the lower pipe chase at B3.

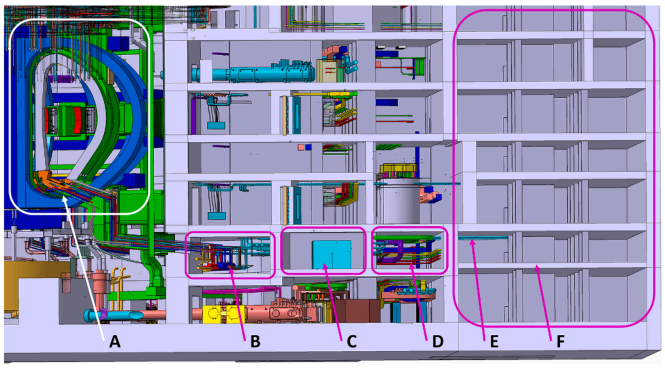

Fig. 2. Cross section through the tokamak building, containing the torus (A), and tritium building. The break in the pumping line occurs in the lower pipe chase indicated by the box labelled B. C indicates the vacuum pumping room, D the gallery of the tokamak building, E the pipeline running from the tokamak building to the tritium building and F the tritium building itself.

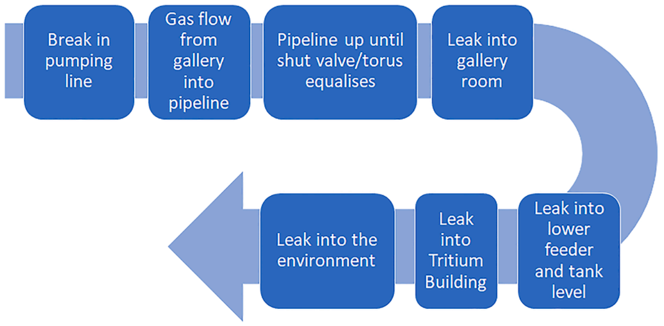
Break in pumping line
Gas flow from gallery into pipeline
Pipeline up until shut valve/torus equalises
Leak into gallery room
Leak into lower feeder and tank level
Leak into Tritium Building
Leak into the environment

Fig. 3. Summary of accident progression.

We have begun with a simplified, linear model. Fig. 4 represents our nodalisation of the accident in MELCOR. CVH refers to the control volumes modelled in MELCOR which mainly consist of primary pipework except CVH-005 which represents a wider Gallery room containing air. Each pump and pipework volume represents a metal foil pump

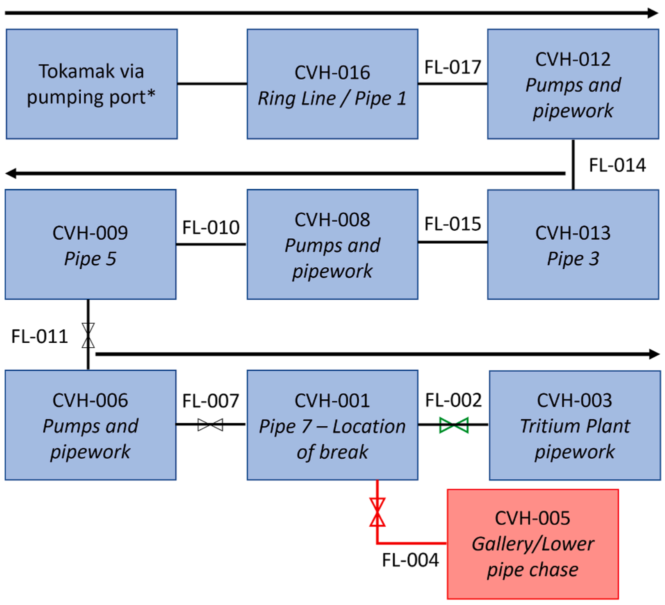
Tokamak via pumping port*
CVH-016 Ring Line / Pipe 1
FL-017
CVH-012 Pumps and pipework
FL-014
CVH-009 Pipe 5
FL-010
CVH-008 Pumps and pipework
FL-015
CVH-013 Pipe 3
FL-011
CVH-006 Pumps and pipework
FL-007
CVH-001 Pipe 7 – Location of break
FL-002
CVH-003 Tritium Plant pipework
FL-004
CVH-005 Gallery/Lower pipe chase

Fig. 4. MELCOR nodalisation of the accident scenario. The black arrows denote the direction of flow. Nodes in red are activated at the moment of the accident.

(MFP), linear diffusion pump (LDP) or a liquid ring pump (LRP) similar to those shown in figure 1 of Giegerich et al. [6]. The pipes are the main connections between the rooms containing the pumps. FL represents the flow paths of the gases (mainly tritium and deuterium) contained in the primary pipework. CVH-001 represents the pipe, located within the Gallery, that fractures during the guillotine break. FL-002 represents the flow of the torus exhaust to the tritium plant during normal operations. Note that in our nodalisation there is no return of the gas through the torus.

The example used here begins with a guillotine break in the pumping line, implemented as the opening of a notional valve in FL-004 (Fig. 4). The two safety relevant valves that could close in reaction to this accident scenario are positioned in FL-007 and FL-011. Once the pipeline reaches equilibrium with respect to the lower pipe chase, tritium begins to escape into the building (referred to as the Gallery, CVH-005).

We have chosen to investigate two aspects; the effect of activating safety relevant valves at different locations in reaction to the accident, and the effect of delaying the reaction time of a safety relevant valve on the quantity of tritium released into the environment.

The apparent simplicity of the nodalisation in Fig. 4 masks the underlying complexity of real-world infrastructure (Figs. 1 and 2): even our simplistic model already generates 855 process variables that may need to be analysed and understood by the user. Similarly, our input deck is already 591 lines long, compared to the second example in [2] – “Adiabatic Expansion of Hydrogen” – at just 153, increasing the difficulty of maintaining code correctness.

3. Code execution

The user’s goal is to understand the consequences of some hypothesised event: e.g. how much tritium is released from a broken pipe? The user merely wishes to define the input to the model and request the batch system to carry out the job.

Executing MELCOR tasks sequentially and the need to individually respecify the names of multiple output and log files cause excess user burden. We have resolved this with simple shell scripts that can be downloaded from [7] that isolate and repackage the output data from separate invocations of MELCOR and pass the overhead of finding idle compute nodes and monitoring job progress over to the cluster’s batch system (here, SGE [3]).

Our workflow is illustrated in Fig. 5. We prepare the description of the plant and accident scenario as a MELCOR input file on a Microsoft Windows laptop using Notepad, and upload to an institutional Gitlab instance using TortoiseGit [8].

The base input file contains all the relevant data that is constant between runs. Any quantity that can vary is replaced with a placeholder that can later be substituted by some wrapper script.

Once we have defined the input to the model we request the SGE batch system to carry out the computing job. SGE locates suitable available compute nodes and runs the individual tasks.

We have written one shell script– *wrap.sh* – that:

- creates a unique temporary directory on the node’s local storage
- ensures all necessary input files are present: e.g. input deck and physical properties
- runs MELGEN and then MELCOR, checking for reported errors
- and consolidates the output on central shared storage.

The script parses the MELGEN and MELCOR output (log) files knowing what the output from successful completion looks like and will alert the user and return an error code (to the batch system) when a problem has occurred. It does not check for infinite execution, but this could be enforced by the batch system.

It also makes simple checks in advance for common environment

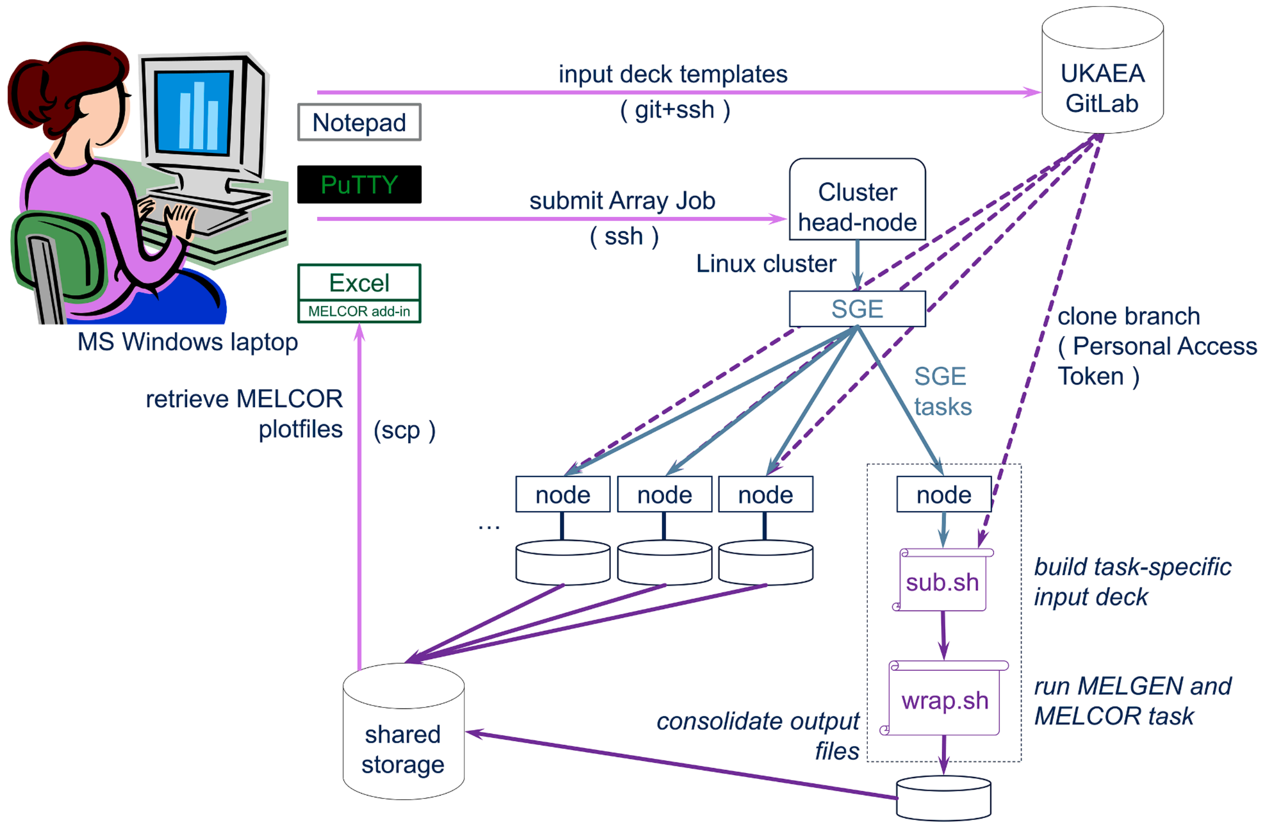

Fig. 5. Improved workflow: tasks are submitted from a standard Windows laptop to nodes on a compute farm running Linux. The batch system (here, SGE) identifies idle machines and distributes the tasks.

issues, such as a lack of free disk space.

We collate the actual input deck, log files, and output plotfile into a single zip archive for reference. This allows for systematic storage and retrieval of previous runs and consequent cross-checking. For example, if we have a parameter sweep of delays of a single valve, and a matrix of valve actuations at a single delay, we can readily check that the input decks for the matching case across both studies are indeed identical. *wrap.sh* can be run directly by the user, but running tasks via the batch system also reduces the need to manually monitor long-running jobs.

In effect, *wrap.sh* acts as an abstraction layer enclosing MELCOR specifics; thus, the few differences in output files and error reporting between MELCOR versions can be handled within this script. We have thus been able to use this framework to successfully run jobs using both MELCOR versions 1.8.5 and 1.8.6 for Fusion.

The production version of a second script, *sub.sh*:

- substitutes values for placeholders such as the run name, and fills out the sweep parameters provided by SGE
- calls *wrap.sh* to run the task

sub.sh allows us to systematically generate a series of input decks – representing the specific cases within a larger study – from a single template, thus saving both the overhead of manually repeating the changes across many files, and also reducing the risk of introducing accidental changes elsewhere in the file. Generating the changes from within the batch system means that SGE takes on the effort of finding idle nodes and running and monitoring the multiple tasks. As illustrated in Fig. 5, we have also successfully deployed a variant of the *wrap.sh* and *sub.sh* scripts that fetches the template input deck from a version-control system (here, GitLab) rather than reading from disk. Unfortunately, we have not yet had the opportunity to refactor the code such that we can readily work with input decks on disk and in Gitlab.

Finally, we fetch the outputs from the cluster shared storage for further investigation (in our case, via the MELCOR add-in to Microsoft Excel), and hopefully gain new insights and ideas for mitigation.

The benefit of the *wrap.sh* and *sub.sh* scripts is that the user does not require more computing knowledge to get the best out of MELCOR modelling. All that is required is knowledge of writing MELCOR inputs and the ability to submit the wrapper scripts to the batch system.

4. Results

We present here summary results from our two example cases, run with MELCOR 1.8.5. Losses will be given as a fraction of the total mobilisable inventory at the instant of the accident. We only consider transport within the tokamak building.

For the activation matrix, enabling and disabling different valves requires multiple changes to the input deck and hence we manually created four successive revisions of the input deck within GitLab, submitting to the batch system after each. We found that the input decks were difficult to check as the mark-up format, as seen in [2], is not intuitive to read making it difficult to spot (possibly unintended) changes to the input parameters in the code. Using a GitLab repository has ensured that all input decks produced in MELCOR are version controlled, making it easy to visualise the actual changes made by using the diff feature. Version control allows the easy creation of an auditable trail of parameter sets to incorporate into accident analysis reports and subsequent safety case analyses. It also enables repeat runs on outlier cases. GitLab readily allows multiple people to work on the code collaboratively.

Fig. 6 shows the hypothetical amount of tritium escaping into the Gallery (CVH-005) following the guillotine break of the pipework at 500 s into the simulation. The figure shows that the position of the safety relevant valve is important, as loss of tritium can be reduced significantly the nearer the safety relevant valve is to the fracture point. This is expected, as a valve closing closer to the fracture point reduces the

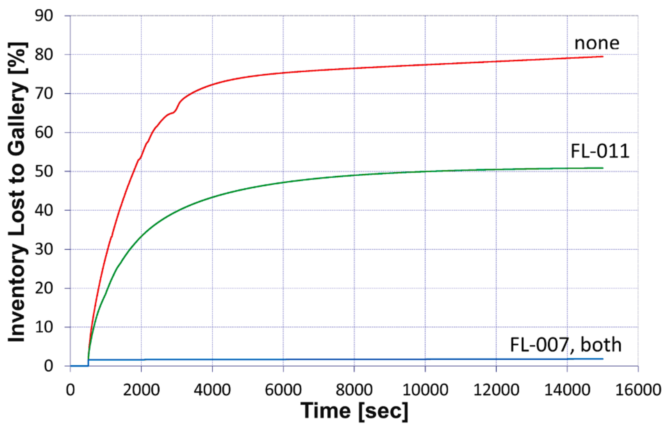

Fig. 6. The cumulative fraction of available tritium lost into the Gallery, CVH-005, following the pipe breaking. Each set of results represents a different combination of the valves in FL-007 and FL-011 responding (closing) or staying open.

volume of tritium-containing pipework exposed to the Gallery room. This suggests that valves should be placed at locations close to areas of concern, i.e. weaknesses in pipework such as welds. In the simulation it makes no difference whether FL-007 is closed on its own or if FL-011 and FL-007 close together, which is expected.

For the delay sweep we generate the only changing variable - the delay - in SGE automatically by using the Array Job feature, substituting it into the input deck within the *sub.sh* script when the task is run. The batch system can then execute potentially all of the cases simultaneously (assuming sufficient computing nodes are free) and thus all 22 tasks can complete in the same time as just one. Automated parameter sweeps enable outlier cases to be followed up with more detail as there is nearly no effort required to increase the resolution of the sweep in regions of interest.

Results of the sensitivity analysis for valve closure delay indicate counter-intuitively that as the delay initially increases up to 120 s, the total amount of tritium released decreases (Fig. 7); with response delays of 180 s and longer the tritium release increases as expected. This may be due to the time it takes for the pipework to reach equilibrium with the Gallery. Initially, the air, at atmospheric pressure, rushes into the primary pipework, at 95 kPa, pushing back the gas contained within and so reducing the amount of tritium lost to the room. After 120 s the primary pipework may have taken in as much of the air from the room as possible and therefore the amount of tritium released increases. This parameter sweep outcome demonstrates the advantage of readily running MELCOR tasks in bulk, and thus of being able to run a wide-ranging parameter sweep, in improving safety design as it alerts the user to unexpected

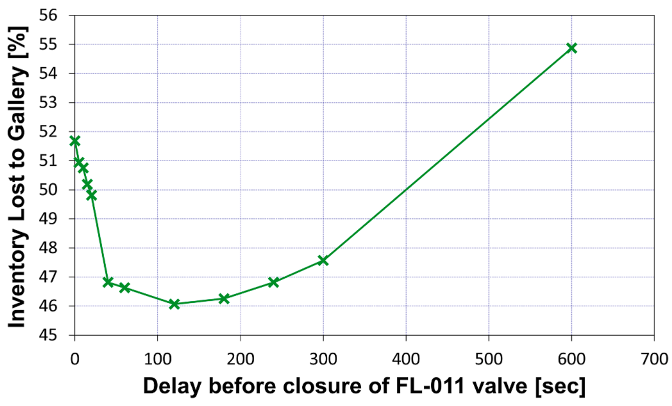

Fig. 7. Overall tritium release into Gallery as FL-011 valve closure delay increases.

changes in behaviour of quantities of interest. In this case, delaying the response of the FL-011 valve unexpectedly reduces the amount of tritium-containing gas released to the environment.

The ability to automate the running of parameter sweeps is extremely beneficial as accident scenario modelling becomes more detailed. Our models are relatively simple so our tasks only take 45 min each to run, however, this rise in complexity will lead to an inevitable rise in run time of the code. Automating running these parameter sweeps in parallel not only reduces the time taken to run all values in a parameter sweep but also means that there is no need for the user to keep checking on the running of the code until all the outputs are returned to them.

5. Conclusions

Efficient parallel running of related cases readily provides new insights into accident consequences and mitigation; for example, counter-intuitively delaying the response of the FL-011 valve can reduce the amount of tritium-containing gas released to the environment. Our models are relatively simple so our tasks only take 45 min each to run, though as the complete data set for the delay sweep illustrated has 22 tasks we already see a significant benefit in completion time. Comparison of Figs. 1 and 2 versus Fig. 4 shows the extent to which our nodalisation is a simplification of reality: detailed models can require run times approaching a day for each task making manual sequential execution impractical.

We still find the final data exploration step a bottleneck: importing the plotfile into Microsoft Excel and generating the relevant charts manually can take the best part of an hour for each task. Automation of the data extraction and reduction is thus a critical counterpart to our improvements to code execution.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion) and from the EPSRC [grant number EP/W006839/1]. To obtain further information on the data and models underlying this paper please contact PublicationsManager@ukaea.uk. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Fig. 1 herein is an annotated version of figure 4 of C. Gliss et al. [5] as distributed under CC-BY-4.0. Visualisation of DEMO in Fig. 2 herein has been extracted by Curt Gliss from a DEMO CAD model by the EUROfusion Consortium, Garching, Germany. In our Fig. 5, “The User” is clip-art originally included with Microsoft Office 2000.

References

[1] B.J. Merrill, P.W. Humrickhouse, M. Shimada, Recent development and application of a new safety analysis code for fusion reactors, Fusion Eng. Des. 109–111 (2016) 970–974, https://doi.org/10.1016/j.fusengdes.2016.01.041.
[2] L.N. Kmetyk, MELCOR Assessment: Gedanken Problems, Volume 1, SAND92-0762, NPRW-SA92-22, Sandia National Laboratories, Albuquerque, NM, 1993. https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/DE93008698.xhtml.
[3] W. Gentzsch, Sun Grid Engine: towards creating a compute power grid, in: Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, Brisbane, Queensland, Australia, 2001, pp. 35–36, https://doi.org/10.1109/CCGRID.2001.923173.
[4] R. Shaw, B. Butler, Initial accident scenario analysis in support of a preliminary DEMO tritium plant design, Fusion Eng. Des. 189 (2023) 113482, https://doi.org/10.1016/j.fusengdes.2023.113482.
[5] C. Gliss, et al., Integrated design of tokamak building concepts including ex-vessel maintenance, Fusion Eng. Des. 177 (2022) 113068, https://doi.org/10.1016/j.fusengdes.2022.113068.
[6] T. Giegerich, et al., Preliminary configuration of the torus vacuum pumping system installed in the DEMO lower port, Fusion Eng. Des. 146 (2019) 2180–2183, https://doi.org/10.1016/j.fusengdes.2019.03.147.
[7] Nebrensky, J.J., Karajgikar, G. Shell scripts to support running MELCOR within batch systems (2024), doi:10.14468/82gv-2676.
[8] TortoiseGit. https://tortoisegit.org/, 2023 (accessed 11 October 2023).