NENE
International Conference
Nuclear Energy for New Europe

Implementation and Application of a GitLab Continuous Integration Pipeline for AC² Validation

Vincenzo Anthony Di Nora
Gesellschaft für Anlagen- und Reaktorsicherheit (GRS)
Forschungszentrum Boltzmannstraße 14
85748 Garching, Germany
vincenzo.dinora@grs.de

Andreas Wielenberg, Thorsten Hollands
Gesellschaft für Anlagen- und Reaktorsicherheit(GRS)
Forschungszentrum Boltzmannstraße 14
85748 Garching, Germany
andreas.wielenberg@grs.de, thorsten.hollands@grs.de

ABSTRACT

System codes are applied to demonstrate nuclear facility safety and reliability using best-estimate models. They require proper verification and validation in line with current nuclear regulations. The AC² system code package is developed by the Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) for analysing nuclear facility behaviour from operating conditions up to severe accidents. A systematic process for verification and validation during code development is implemented. This includes an automated continuous integration process using a Jenkins® server. To strengthen its verification and validation activities, GRS will increasingly use the continuous integration and deployment features of Gitlab®. A well-structured continuous integration and deployment pipeline configuration was developed for the multi-project AC² package. It is capable of performing automated testing and generating graphical comparisons to support the assessment of new AC² code development branches or beta versions. The working principle and use cases of this pipeline are presented in this work. Its application to the assessment of a beta version of AC²-2023 illustrates its flexibility and efficiency, as well as its utility for the verification and validation process.

1 INTRODUCTION

System codes play a crucial role in ensuring the safety and reliability of nuclear facilities through the use of best-estimate models for analyses and safety demonstrations. To meet regulatory requirements, such codes must undergo proper qualification, including a systematic verification and validation process during their development.

The Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) has developed the AC² code package to analyse operational conditions, design basis, design extension conditions, and severe accidents in nuclear facilities. The quality assurance process for the AC² codes, see e.g. [1], implements regulatory expectations as found, e.g., in IAEA SSG-2 [2]. The AC² development process is supported and documented using a GitLab® instance running on a server operated by GRS. Verification and validation tests are still conducted on a separate Jenkins® server and then assessed by GRS experts. Systematic validation is performed against both existing and new validation cases by a dedicated validation team outside of Jenkins® and GitLab®. Suitable sets of test cases are used to support development in order to

confirm non-regression and specified functionality of new features and different sets are used for the actual validation of pre-release and release versions.

The use of GitLab allows to exploit its embedded automated testing and continuous integration/continuous deployment (CI/CD) features, enabling basic unit tests for the codes and, in principle, detailed verification calculations. As AC^2 consists of multiple repositories, navigating consistent multi-project builds and thus supporting multi-project development adds to the challenge. By leveraging GitLab CI/CD capabilities, the overall process can be supported for several tasks, and importantly, by implementing structured CI/CD test trains, so-called pipelines, capable of partially automatizing the verification and validation process while leaving final assessments to the experts.

Starting from simplified existing GitLab pipeline configurations, and a post-processing plotting tool developed by GRS called "batchplot", this work introduces the work on setting up a GitLab pipeline tailored to supporting the verification and validation of AC² codes. The implemented pipeline was designed to:

- Either retrieve pre-built AC² executables or build AC² executable from specific user-defined (development) branches in the respective separate code and input deck repositories.
- Utilize selected executables to perform calculations against user-selected validation matrices of samples or non-regression test cases.
- Automatically compare the solutions obtained by the different code versions: For instance, by generating sets of several selected figures of merit, optionally including measurement data of test cases.
- Automatically generate a report file with previously generated plots of figures of merit for the purpose of assessing assessment AC² code versions.

The implementation of such pipelines makes the verification and validation efforts both more effective and allows for a more systematic consideration of numerous test cases, thus ensuring further predictiveness and reliability of the new AC² versions for their intended applications.

2 BACKGROUND

For the sake of completeness, the subsequent sections provide an overview of GitLab along with its CI/CD capabilities, as well as a description of the GRS in-house post-processing plotting tool batchplot, and its functionalities.

2.1 Gitlab

GitLab is a web-based Git repository manager that provides a comprehensive DevOps platform for software development teams [3]. GitLab is a powerful tool that offers a wide range of features to help development teams efficiently manage their software development lifecycle. GitLab is built on top of the Git version control system, which allows developers to track changes in their codebase, collaborate with other developer teams, and manage multiple versions of their projects.

At its core, GitLab functions as a Git repository hosting service, providing users with a centralized platform to store and manage their Git repositories. Developers can clone, push, and pull code, or generic files, such as e.g., input decks, from these repositories, making it easy to collaborate on projects and keep track of changes made by different team members. GitLab allows users to create and manage multiple Git repositories within a single instance.

GitLab also offers dedicated features for continuous integration (CI) and continuous deployment (CD) that are essential for modern software development practices, particularly in the context of DevOps. CI is the practice of frequently integrating code changes from multiple developers into a shared repository or multiple repositories. Code changes are automatically checked by running a series of automated tests. The main goal of CI is to detect integration issues early in the development process, ensuring that the code base remains in a consistent and functioning state. CD is an extension of CI. It involves automatically deploying code changes that pass the CI tests. With CD, the entire process of building, testing, and deploying software is automated, eliminating the need for manual interventions.

CI/CD pipelines are defined in a configuration file, i.e., the ".gitlab-ci.yml" file, and consist of different stages and jobs. Each job represents a specific task, such as building an executable, running tests, or deploying to a server. Stages serve to group related jobs together, representing the steps in the pipeline. For each CI/CD pipeline, GitLab can retain build artifacts, which are the outputs generated during the process. These artifacts include compiled binaries, test reports, and other relevant files. Storing artifacts as specified in the pipeline allows easy access to the job outcomes and helps in maintaining and auditing the codebase. Using CI/CD pipelines come with several benefits. For instance:

- Early detection of issues: CI/CD pipelines detect integration problems and bugs early in the development process, particularly for multi-project developments, reducing the likelihood of major issues in production.
- Faster feedback loops: Automated testing and deployment allow developers to receive immediate feedback on their code changes, enabling faster iteration and development cycles.
- Increased reliability: Automated testing and deployment reduce the chance of human error, leading to more reliable and stable software releases.

2.2 Batchplot

Batchplot is a versatile post-processing tool developed by GRS and distributed together with the AC² package. Its primary objective is to streamline the analysis through graphical visualization of data trends produced by AC². It provides a standardized and platform-independent solution for post-processing AC² data in the Hierarchical Data Format (HDF5) format, as well as Comma-Separated Values (CSV) format. Batchplot is a Python-based tool relying on two essential Python libraries: matplotlib for high-end plotting capabilities and PyTables for efficient access to HDF5 data files. By utilizing Python, batchplot achieves platform independence.

The central focus of batchplot is to offer a user-friendly experience, catering to both basic and advanced users for efficient data processing. This is achieved through the utilization of user-defined input files, referred to as batchdef.py files. These files automate post-processing tasks and enable advanced data analysis. With ATLASneo, AC²-2023 will provide an advanced GUI that is based on the same paradigm as batchplot.

3 IMPLEMENTATION

In agreement with the specifications and goals outlined in section 1, a sophisticated, and adaptable GitLab CI/CD pipeline has been designed and implemented to facilitate the verification and validation. The specific realization we discuss in the following focusses on the needs and task of the validation team; the pipeline can be easily adapted to other needs.

The new pipeline was structured into four main stages, designed as "prepare", "build", "run", and "report". Stages, related jobs, and their working principles are presented in the next

dedicated sections and finally summarized in Table 1. Further refinements will occur based on user feedback.

3.1 Stage prepare

"Prepare" is the initial stage of the pipeline and encompasses a set of crucial tasks focused on preparation, calculation, and data retrieval. The following jobs are involved:

- getUtility: This job is responsible for fetching essential utility scripts specifically developed to enhance test batchdef.py files and report templates. The artifacts obtained through getUtility are made available to all subsequent stages.
- getBatch: Within this task, the python script of batchplot is retrieved from the designated project. This ensures that the most updated version of the script and its new features are available for use. Alternatively, the batchplot script can be integrated into the project where the pipeline is run.
- getCmake: In this job, user-selected repository versions of the AC² code packages, along with the necessary building files, are downloaded and stored as artifacts. This facilitates the comparison of different versions of the AC² package, and specifically developer versions.
- getSamples: This task is responsible for retrieving samples from a user-specified repository, providing enhanced flexibility. Similar to getCmake, it allows the retrieval of different versions of the sample repository, which are suitable for the selected code versions.

3.2 Stage build

This stage is executed optionally, primarily when testing or comparing specific development branches where no existing deployments are available on the server. It includes jobs "buildLinux<n>" where n can currently assume values from 1 to 4. These jobs build the executables from user-defines development branches of AC². It requires artifacts from getCmake jobs. This type of job is only executed when branches are specified by the user to be built.

3.3 Stage run

The running stage is responsible for performing calculations on the matrix of samples. It includes jobs "runLinux<n>" where n can currently assume values from 1 to 4. These jobs retrieve pre-built executables of the package, retrieving them from their deployment location. Alternatively, it retrieves AC² executables from artifacts of buildLinux jobs. Then it performs calculations of the samples specified by the user, which requires input decks via the getSamples job, and, optionally, executables from buildLinux.

3.4 Stage report

This stage is dedicated to generating plot data and text files for reporting purposes. It comprises the following essential jobs:

- plotData: This task involves accessing hdf5-files for each analysed test and executing the batchplot python script with predefined batchdef.py files. The outcome is a series of comparison plots showcasing transient trends of user-defined figures of merit. To accomplish this, the job requires artifacts (hdf5-files) obtained from runLinux and getSamples.

- writeTex: For each test analysed, this task generates well-structured tex-files that present the graphical comparisons previously created by plotData. It extends the process using utility script templates downloaded through getUtility. This step necessitates plots from plotData and the script and template from getUtility.
- buildPdf: The final task involves compiling all the tex-files generated in the previous step into a comprehensive report pdf-file. This compilation ensures that all the analysis and comparisons carried out in the earlier stages are consolidated into a single, cohesive document.

Table 1: Stages and jobs summary

Stage	Jobs	Scope	Dependecies
Prepare	getRegre	Retrieves pipeline utility scripts	-
	getBatch	Retrieving batchplot script	-
	getTest<n>	Retrieving samples	-
	getSrc<n>	Retrieving executable source	-
Build	linBuild<n>	Building executable from source files	getSrc<n>
Run	linRun<n>	Running samples	getTest<n>, linBuild pdf
Report	linCmp	Preparing plots	getRegre, getBatch, linRun pdf
	linRep	Preparing tex-files	getRegre, linCmp
	linPdf	Compiling report pdf -file	linRep

With <n> currently ranging from 1 to 4

3.5 Samples repositories

As the validation process is based on validation matrices, we provide a brief description of the sample repositories, employed for verification/validation purposes.

Validation matrices consist of sets of input-decks, i.e. samples, that have been selected for verification or validation purposes. While implementing the current pipelines, suitable sets of samples were identified and collected in dedicated repository for each purpose, hereafter referred to as sample repositories, so to facilitate the access of each pipeline to required data. A sample repository can be selected according to user needs. For instance, a sample repository can include extensive sample matrices for validation of a new AC² release, or simple sample matrices, even a single sample, for unit tests verification.

The current configuration of sample repositories provides for in the repository the presence of a .matrix.yml which serves as an extension of the pipeline configuration .gitlab-ci.yml file. This file includes a list of samples to be run via the pipeline, for which reports are produced, which can then be assessed. For each sample in that list, the repository should also include dedicated sample folders, containing:

- The sample to be considered.
- A batchdef.py, providing the instruction for generating the sample's figures of merit.
- Optional experimental data or reference values to compare with in the plots of figures of merit against AC² evaluated trends.

4 APPLICATION

The upcoming release AC²-2023 includes several improvements requiring verification and validation such as: upgraded fuel rod models, Reynolds number dependent form loss coefficients, new working fluids and gas components, nitride formation models in melting process, overall program enhancements, and various bug fixes.

For the AC²-2023 pre-release validation, the implemented CI/CD pipeline has been successfully applied to the first internal beta versions release. In the following we illustrate this for two distinct simulated validation tests for ATHLET and ATHLET-CD: The test case for ATHLET is "Run 916" (performed at the "ROSA-III" facility), and ATHLET-CD case involving core degradation phenomena is the "QUENCH-18" test. Considering the changes in modelling basis, changes between the new and old code versions are expected to be minor. The results as shown have been extracted from the report file automatically generated by the pipeline.

4.1 ROSA-III: Run 916

The Rig of Safety Assessment (ROSA)-III is a program, initiated in 1976, aimed to study a Boiling Water Reactor's behaviour during a simulated Loss of Coolant Accident (LOCA) with Emergency Core Cooling System (ECCS) activation. This was done using a scaled-down facility resembling a 3500 MW BWR/6-251 reactor. The program's objectives were to collect data for improving predictive computer codes, assessing ECCS and safety feature performance, and understanding unexpected behaviour in reactor and safety systems [4]. The nuclear facility of the program consists of four subsystems: the pressure vessel, steam line and feedwater line, recirculation loops, and the ECCS. The core contains model fuel assemblies with electrically heated rods, and several instruments to measure various thermal-hydraulic parameters during tests.

The test Run 916 carried out at the facility simulated a 50% break at the recirculation pump suction in one of the recirculation lines. A sharp-edged orifice was used as break plane, and blowdown was initiated by opening a blowdown valve. The initial conditions and specifications of the test were provided in [4], including core power, steam flow, and ECCS actuation timings. The test was terminated after the entire core was quenched, which occurred 255 seconds after the break initiation.

The ROSA-III facility and the Run 916 test were modelled by applying the AC² code ATHLET. Details on modelling including system network, nodalization, boundary conditions, etc. are reported with a high degree of detail in [1]. Calculations of the test were already part of the validation for the release AC²-2021. Therefore "Run 916" test have been recalculated with a recent beta version of AC²-using the CI/CD pipeline.

A subset of such figures of merit was selected, and is shown in Fig. 1, and briefly discussed in the following. For more detail on the significance of the trends' behaviour please refer [1].

Figures of merit for the Run 916 test selected for discussion were:

- Coolant temperature at core outlet, and in lower plenum, see Fig. 1a, and b: The graphs compare the liquid and vapor temperatures calculated by AC²/ATHLET at the core outlet and in the lower plenum, and fluid temperatures measurements. For both liquid and vapor trends, the AC² releases provide almost identical results. Compared to the previous release, the small additional underestimation of liquid temperature between 250 to 300 s is not significant and acceptable considering that phase temperatures are compared to one measured temperature value.
- Total break mass flow, and steam line mass flow, see respectively Fig. 1c, and d: The left figure shows the total break mass flow as a sum of the contribution of leak mass

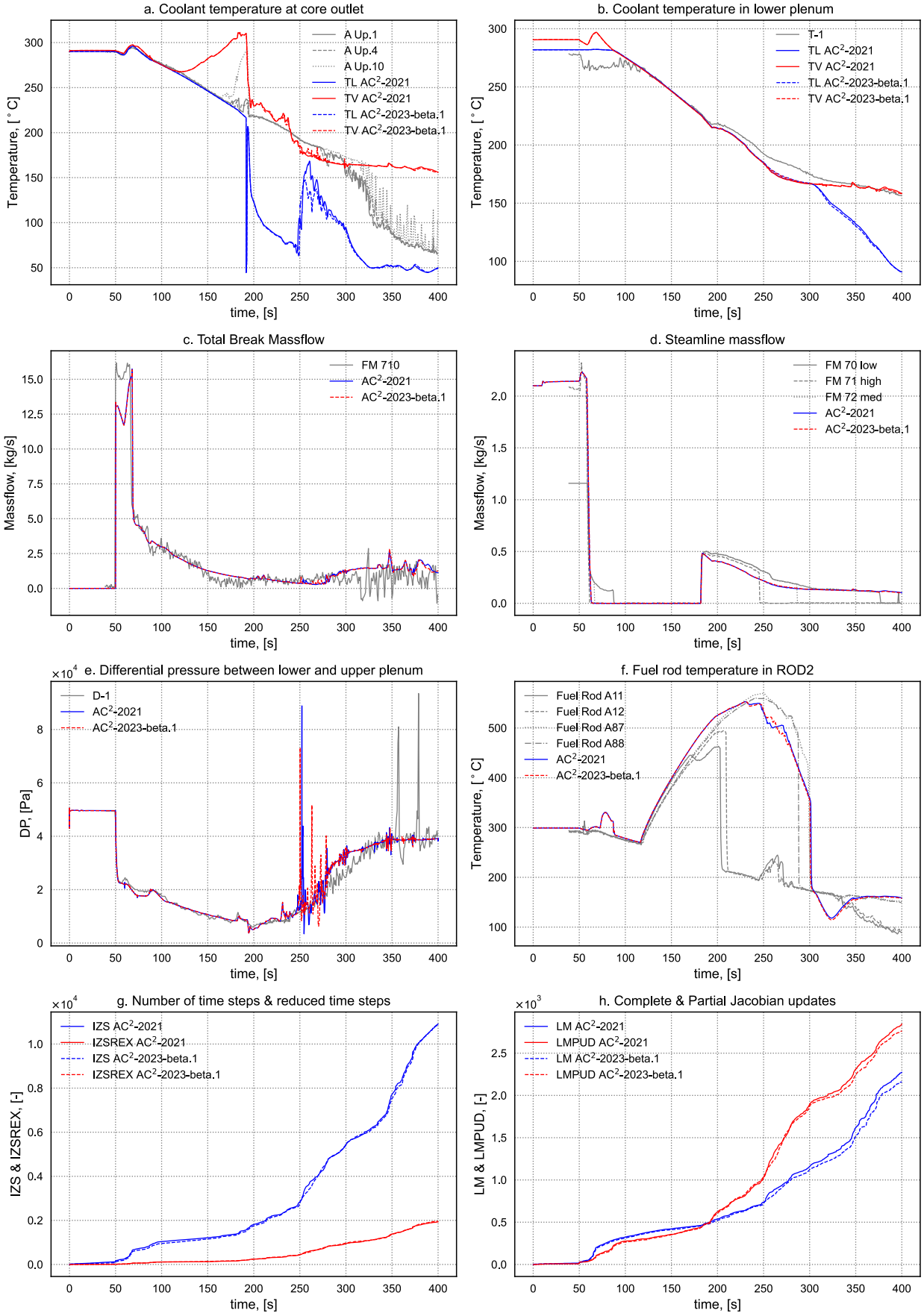

Figure 1: ROSA-III test: Figures of merit

flows on both the reactor vessel and main recirculation pump sides of the simulated break. The trends calculated by the two code versions are practically the same and both underestimate the actual break mass flow trend between 50 and 75 s, in line with expectations given well-known limitations of the break flow modelling during that phase. The right figure presents the steam line mass flow. A good agreement was generally found between the two evaluated trends and the experimental data.

- Differential pressure between lower and upper plenum, see Fig. 1e: Despite the unstable pressure behaviour expected from 200 and 400 s during the test, the pressure difference.

between lower and upper plenum appears to be consistently predicted by both AC² releases. Qualitatively good agreement of the trends is also found with respect to measured data.

- Fuel rod temperature of the hottest fuel bundle, see Fig. 1f: The figure presents the fuel rod temperature of the hottest fuel rod in proximity to the core top, as well as the dry out of the top core region. Both code versions present similar prediction capabilities that overestimate in both cases the temperature in the dry out phase and the time to successful reflooding.
- Total number of evaluated times steps and reduced time step number, respectively designated as IZS, and IZSREX, as well as complete and partial updates of the Jacobian of the system model, respectively designated as LM, and LMPUD, see Fig. 1g, and h: These figures of merit were chosen as representative indicators as they effectively summarize the numerical performance of the AC²/ATHLET simulation. Both versions exhibit remarkable consistency. This shows that the new version has no additional performance issues.

4.2 QUENCH-18

The QUENCH program at Karlsruhe Institute of Technology aims to study core thermal response, cladding oxidation with hydrogen release, and water injection cooling under accident conditions in nuclear reactors [5]. The program began in 1996 and continues to this day, involving nineteen high-temperature bundle tests. One such test, QUENCH-18, was conducted in 2017 as part of the ALISA program. This experiment focused on air ingress and aerosol release, investigating the oxidation of M5® claddings in a mixed air/steam atmosphere after pre-oxidation in steam. The experiment aimed to understand interactions with nitrogen and the effects of control rods and pressurized rod simulators on bundle degradation. This research is relevant for studying severe accidents in nuclear power plants and spent fuel pools, with other institutions also conducting similar studies. The QUENCH test facility comprises a test section with a bundle of up to 22 fuel rod simulators. This setup includes a steam generator and superheater, which supply superheated steam and argon as a carrier gas into the bottom of the test bundle. Water injection for reflooding is also possible through separate inlets at the bottom. The bundle consists of heated and unheated fuel rod simulators, resembling those in nuclear reactors, with the heated ones utilizing Zircaloy-4 cladding and tungsten heating elements. The test section is equipped with various thermocouples for temperature measurements at different points, including the rods, shroud, and cooling jackets. A Zircaloy shroud with insulation and a stainless-steel double-walled cooling jacket surrounds the bundle. Hydrogen production is monitored using mass spectrometers and a hydrogen analyser.

As described in [5], in the test the rod bundle is heated with varying power and at varying gas flows. Initially, the bundle is pre-heated to 900 K using 4.1 kW power, while simultaneously injecting argon and steam. The power is then raised to 9.1 kW for pre-oxidation. Rods #9 and #15 fail at 1035 K and 1045 K, respectively. After 4000 s, the temperature hits about 1400 K at 950 mm height, generating 11.5 g of hydrogen. Cooling starts by reducing power to 3.8 kW at 6309 s, lowering the maximum rod temperature to 1080 K. Air is introduced at 7537 s to prevent excessive oxidation, leading to increased oxygen consumption. The first AIC-rod failure of occurs at 10530 s, releasing aerosols with a cladding temperature at around 1350 K.

Steam consumption rises afterward, accompanied by hydrogen release. Accelerated heating occurs around 10660 s, followed by nitrogen consumption. Major cladding tube failure leads to absorber rod melt relocation. By 11000 s, steam is completely consumed. Around 11253 s, argon ingress begins due to shroud failure, releasing 45 g of hydrogen while consuming oxygen and nitrogen. The quenching phase starting at 12329 s involves injecting 50 g/s of water at 3.8 kW power, with air and steam off. Temperatures peak at 2450 K in the middle and upper zones. Quenching is quicker in the middle, fully quenching the bundle in about 800 s. This flooding phase produces around 238 g of hydrogen and releases over 54 g of nitrogen due to nitride re-oxidation.

The QUENCH-18 test have been simulated using the AC² package, with detailed modelling information available in [6]. Previous validation was already conducted for AC²-2021. For pre-release validation of AC²/ATHLET-CD, the "QUENCH-18" test part of a CI/CD pipeline.

As done for the ROSA-III Run 916, experts identified several figures of merits to the assess solution quality and code performance of the new version. Trends for the different code versions are compared using batchplot features in the pipeline. In the following, we show a subset of these figures of merit as extracted from the automatically generated report of the CI/CD pipeline, see Fig. 2, and briefly the results. For more detail on the significance of the trends' behaviour please refer to [6].

The selected figures of merit for the QUENCH-18 test selected are:

- Cladding temperatures at elevation 550 mm, see Fig. 2a: Both release versions slightly overestimate temperatures and exhibit similar results up to flooding time. Quenching temperature excursions are well captured, but both versions also show variations, and predict delaying cooling by about 500 s compared to the experiment. In general, the evaluated temperature trends overlap during roughly the first 12000 s, and temperatures during oxidation are systematically overestimated. During the late oxidation and then quenching phase, both versions show some deviations, however the newer version actually quenches a bit earlier.
- Integral hydrogen production, see Fig. 2b: The figure illustrates the integral mass of hydrogen generated, both with and without auxiliary heat structures, labelled as "H2.MASS1" and "ACCH2", respectively. Both releases predict similar total integral hydrogen masses, which are systematically underestimated. However, they show markedly divergent trends in the hydrogen mass generated by bundle rods. This disparity suggests the possible existence of modelling issues in the new version and requires further detailed analysis.
- Bundle outlet mass flowrates of steam, hydrogen oxygen, and nitrogen, see Fig. 2c, d, e, and f: Fig. 2c, and d show that AC²-2021 and AC²-2023-beta.1 predict similar trends of steam and hydrogen mass flowrates. For the latter, realistic behaviour of the hydrogen mass flow rate could not be simulated by both releases. As evident from Fig. 2e, for nitrogen, and Fig. 2f, for oxygen, there is a good agreement between simulations and experiments for outlet mass flows during the whole air ingress phase (first 7800 seconds). However, the abrupt increase of oxygen and nitrogen mass flow rates after the beginning of quenching could not be captured by both release versions. This is probably due to the fact that nitride reoxidation is not considered in the simulation. The evaluated trends of nitrogen, and oxygen mass flow rate are however in good agreement with each other, with an exception made for a brief timeframe around 12000 s.
- Total number of evaluated times steps and reduced time step number, respectively designated as IZS, and IZSREX, as well as, complete and partial Jacobian updates, respectively designated as LM, and LMPUD, see Fig. 2g, and h: The new release requires both a higher number of time steps and complete Jacobian updates to

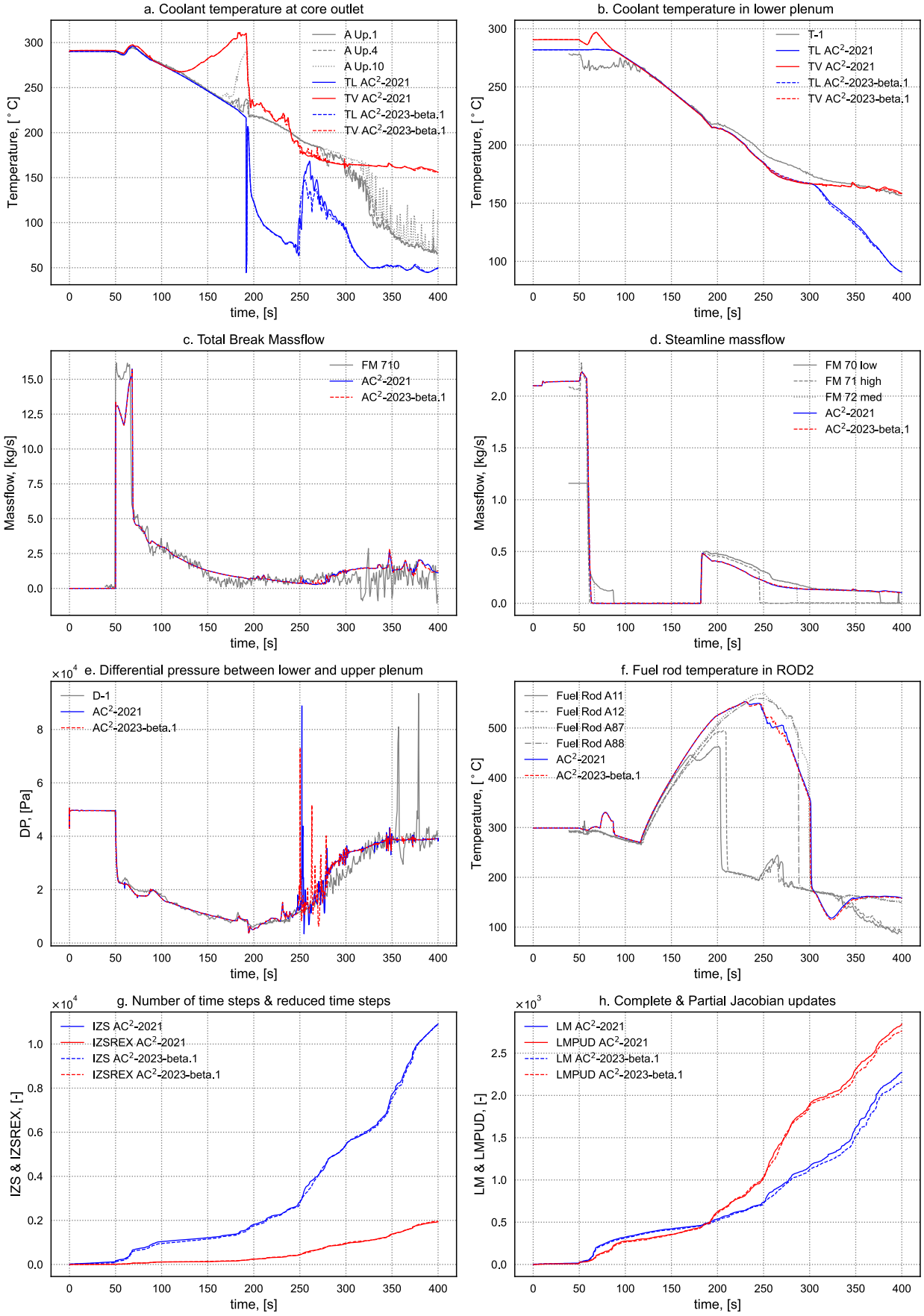

Figure 2: ROSA-III test: Figures of merit

accomplish the simulation, hinting at the deterioration of the new release's computational performance with respect to the last release. While this may well be spurious is requires careful follow-up in further test cases.

Overall, the figures of merits automatically compared via the pipeline offer a solid supportive basis for the assessment of the new release candidate by GRS experts, helping to detect unexpected behaviour and potential bugs early in the process. Based on this, further improvements and bugfixes were introduced in the current version, which will be verified and validated further for its release.

5 CONCLUSION

5.1 Summary

The present work focused on the development of a GitLab CI/CD pipeline that efficiently supports the verification and validation activities for AC², both from the perspective of testing new potential AC² candidates for future official releases and for testing new development branches of models and packages. The implementation strategy was discussed in detail, including descriptions of essential steps and work flow. In particular, the new pipeline is capable of:

- Flexibly retrieve pre-built executables of the package or build executables for specific development branches as specified by the pipeline user.
- Utilize the selected executables to perform calculations on a verification or validation matrix, configurable by the user.
- Automatically compare solutions for different code versions, generating a set of figures of merit.
- Automatically generate a well-structured report file for the user, facilitating the assessment of code versions for verification and validation.

The pipeline was successfully tested by the AC² validation team. Finally, the pipeline was applied for actual validation purposes for AC²-2023-beta.1. It was extensively applied to validation matrices carefully selected by developers. These matrices included scenarios that either anticipated or didn't anticipate core degradation phenomena. We illustrated this via applications to the ROSA-III Run 916 test and the QUENCH-18 assessment. While, no apparent inconsistencies were detected in the validation exercise conducted on the Run 916 test, the validation exercise against the QUENCH-18 revealed significant potential issues in hydrogen generation modelling.

Overall, the application allowed the validation team to early and easily detect inconsistencies providing important hints for eventual bug fixes to incorporate in the upcoming release. Overall that the pipeline achieved its objective of supporting the validation team and increased its efficiency.

It's important to highlight that while the pipeline configuration was initially designed and implemented for the verification and validation of AC², its underlying operational approach holds broader potential applicability. The pipeline working principle could be extended to encompass a wider range of system codes addressing nuclear safety concerns.

5.2 Near-term further developments

Already in its current state, the pipeline has proven to be highly useful to the validation teams. Nevertheless, there is still potential for enhancing this tool further. For instance, while it allows the comparison of trends by Linux OS-based AC² executables, AC² executables built

for Windows® OS should be covered. The same applies to parallelized versions of AC² codes including those that use NuT. Furthermore, the pipeline currently builds AC² executables only in the "release" mode of the CMake package (saving considerable run time). Extending to "debug" mode will be particularly relevant for verification purposes.

Another area for improvement pertains to graphical comparisons. Presently, the pipeline exclusively enables graphical comparison for time-dependent trends. However, for validation purposes, providing also spatial plots could offer significant benefits. For instance, generating graphical comparisons at specific time points for generic axial fuel rod and coolant channel temperature profiles, or even spatial plots of system variables within regions of a simulated nuclear facility (e.g., pressure or temperature in cold and hot legs of LWRs, or in the downcomer), could enhance the pipeline's utility.

A critical verification step involves consistency checks of key representative system variables. Maintaining such consistency for restart cases, e.g., is essential. Traditionally, the validation team such checks manually. However, a more systematic approach could be established by integrating these checks into the pipeline's configuration options.

Finally, extending the pipeline to more tests, verification and validation cases, and setting up suitably configured pipelines for the tasks in the overall AC² quality assurance process is a continual work.

ACKNOWLEDGMENTS

The research at hand was funded by the Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz (BMUV), in accordance with the resolutions of the German Bundestag, as part of the projects RS1600 and UMRS1606.

REFERENCES

[1] T. Hollands, H. Austregesilo, S. Buchholz, N. Dünne, M. Junk, J. Lee, G. Lerchl, P. Schöffel, D. von der Cron and A. Wielenberg, "ATHLET 3.3 - Validation Manual," 2021a.

[2] International Atomic Energy Agency (IAEA), Deterministic Safety Analysis for Nuclear Power Plants, Specific Safety Guide. IAEA Safety Standards Series, SSG-2 (Rev. 1), Vienna, 2019..

[3] GitLab B.V., gitlab, 2023. [Online]. Available: https://docs.gitlab.com/.

[4] T. Yonomoto, K. Tasaka, Y. Koizumi, Y. Anoda, H. Kumamaru, H. Nakamura, M. Suzuki and H. Murata, "ROSA-III 50 % break integral test RUN 916 (JAERI-M--85-109)," Japan, 1985.

[5] J. Stuckert, J. Kalilainen, T. Lind, U. Stegmaier, M. Steinbrück and Y. Zhang, "Results of the QUENCH-18 Bundle Experiment on Air Ingress and AgInCd absorber behavior," Karlsruher Institut für Technologie, Karlsruher, 2020.

[6] T. Hollands, H. Austregesilo, C. D'Alessandro, L. Lovasz, P. Pandazis, L. Tiborcz and A. Wielenberg, "ATHLET-CD 3.3 - Validation Manual," 2021b.

[7] Hollands, T., Austregesilo, H., Buchholz, S., Di Nora, V. A., Dünne, N., Eckert, D., Junk, M., Schöffel, P. J., Wack, J., Wielenberg, A., "ATHLET 3.3.1 Validation. GRS-P1/Vol.3 Rev. 7," Cologne, November 2022.