Proceedings of International Conference on Nuclear Engineering
ICONE 2018
July 22-26, 2018, London, UK

ICONE26-81574

CREATION OF AN OPENFOAM FUEL PERFORMANCE CLASS BASED ON FRED AND INTEGRATION INTO THE GEN-FOAM MULTI-PHYSICS CODE

Carlo Fiorina*
Andreas Pautz
Laboratory for Reactor Physics and System Analysis
École polytechnique fédérale de Lausanne
Lausanne 1015
Switzerland
carlo.fiorina@epfl.ch
andreas.pautz@epfl.ch

Konstantin Mikityuk
Laboratory for Scientific Computing and Modelling
Paul Scherrer Institut
Villigen 5232
Switzerland
konstantin.mikityuk@psi.ch

ABSTRACT

The FRED code is an in-house tool developed at the Paul Scherrer Institut for the so-called 1.5-D nuclear fuel performance analysis. In order to extend its field of application, this code has been re-implemented as a class of the OpenFOAM numerical library. A first objective of this re-implementation is to provide this tool with the parallel scalability necessary for full-core analyses. In addition, the use of OpenFOAM as base library allows for a straightforward interface with the standard OpenFOAM CFD solvers, as well as with the several OpenFOAM-based applications developed by the nuclear engineering community. In this paper, the newly developed FRED-based OpenFOAM class has been integrated in the GeN-Foam multi-physics code mainly developed at the École polytechnique fédérale de Lausanne and at the Paul Scherrer Institut. The paper presents the details of both the re-implementation of the FRED code and of its integration in GeN-Foam. The performances and parallel scalability of the tool are preliminary investigated and an example of application is provided by performing a full-core multi-physics analysis of the European Sodium Fast Reactor.

INTRODUCTION

In recent years there have been rapid advances in the field of multi-physics numerical simulation for nuclear reactors. In particular a trend is observed in the creation of new solvers and in the re-implementation of legacy solvers using modern numerical libraries for the general solution of partial differential equations (PDEs) on general geometries and unstructured meshes. Example of these activities are the Moose project [1] and the several activities dedicated to the use and development of OpenFOAM [2] based solvers [3].

An important example of legacy codes that would benefit from a re-implementation based on modern libraries is that of fuel performance codes. These codes are normally based on specific methodologies that make difficult their parallelization and coupling with other codes. In addition, most of these codes were not conceived for parallel calculation. Re-implementing these tools in a modern library would address these problems by providing proper parallelization and streamlined coupling with other "physics". This re-implementation is however non-trivial due to the peculiar 1.5-D representation of fuel rods, which challenges the application of standard 3-D PDE solvers.

The laboratory for reactor physics and system behavior at the EPFL and PSI has been developing since a few years a multi-physics solver named GeN-Foam [4–7] and based on OpenFOAM. OpenFOAM is distributed as a CFD toolbox but it is built as a general object-oriented library for the finite-volume discretization and parallel solution of PDEs. In this work, the in-house PSI fuel performance code FRED [8] has been re-implemented as an OpenFOAM class and coupled to the GeN-

*Address all correspondence to this author.

Foam multi-physics solver. The paper describes the details of this re-implementation, especially for what concerns the strategy for maintaining a 1.5-D description of single rods, as well as for properly employing the OpenFOAM general parallelization framework for this specific type of methodology.

The work is performed in the framework of the ESFR-SMART Euratom project and aims at supporting its modeling activities by providing a novel tool based on modern computational methodologies and that would supplement available legacy codes in the modeling of complex phenomena that challenge their applicability.

BACKGROUND

This work is based on two main research activities carried out at the EPFL and at the PSI, namely: the development of a simple fuel performance code named FRED; and the implementation of a multi-physics solver named GeN-Foam. The following two subsections briefly describe these tools.

The GeN-Foam multi-physics solver

The GeN-Foam multi-physics solver is an OpenFOAM-based solver for reactor transient analysis that couples [5]: a multi-group neutron diffusion / SP3 sub-solver; a fine- and coarse-mesh thermal-hydraulics sub-solver; a thermal-mechanics sub-solver; and a simple sub-scale fuel model.

The neutron transport sub-solver [6, 7] is based on standard models for neutron diffusion and SP3, including an arbitrary group structure and the use of isotropic discontinuity factors. Different from most legacy solvers however, these models are implemented based on finite volume techniques for general unstructured meshes. This has the drawback of a notably increased computational burden. On the other hand it allows handling arbitrary geometries and it permits the use of standard finite volume routines e.g. for projecting fields from and to different meshes used for other "physics", and for mesh deformation. This has been employed in GeN-Foam to couple the neutronic sub-solver to the other sub-solvers, as well as to directly evaluate the neutronic feedbacks from thermal expansions by deforming the neutronic mesh according to the calculated displacements.

The thermal-mechanics sub-solver is employed to evaluate the thermal deformations of core and structures that contribute to the reactivity feedbacks. It combines a standard linear-elastic displacement-based solver for structures and a simple 1-D evaluation of fuel expansion.

The thermal-hydraulics sub-solver is based on the standard k-ε turbulence model for compressible or incompressible flows, but extended to coarse-mesh applications through the use of a porous medium approach (Vafai, 2005) for user-selected regions inside the mesh. This allows to simulate a full primary circuit using standard CFD techniques for open spaces such as the plena in LWRs or the pools in SFRs, while employing a coarse mesh with dedicated sub-scale models in complex components like core and heat exchangers. Currently, only single-phase fluid flow is allowed.

Since the core is typically modeled with a coarse mesh, a sub-scale fuel model is employed to evaluate the local temperature profile in fuel and cladding, starting from the information about the coolant temperature (from the thermal-hydraulic sub-solver) and the volumetric power (from the neutronic sub-solver). A 1-D radial representation of rods or plates is employed. This simplified fuel model has represented the starting point for implementing the FRED-based sub-solver, and remains as a user-selectable sub-solver that could be used for transient analysis.

A first order implicit Euler scheme is used for time integration. The coupling between equations is semi-implicit using Picard iteration. Three different meshes are used for thermal-hydraulics, thermal-mechanics and neutron diffusion, while the sub-scale fuel model is solved in each mesh cell within the fuel zones of the thermal-hydraulics mesh. This allows for different refinements of the meshes and reduces computational requirements.

The FRED fuel performance code

The FRED code [8] has been developed for the simulation of FR and LWR fuel behavior under base-irradiation and accident conditions. The current version of the code calculates temperature distribution in fuel rods, fuel-cladding gap conductance, fission gas release, fuel rod inner pressure, and, stress-strain condition of fuel and cladding. The code was evaluated in OECD MOX fuel performance benchmarks and against the data of the IFA-503.2 Halden tests with LWR fuel [8].

IMPLEMENTATION

The work here performed has followed two main steps, namely: the re-implementation of the FRED code as an OpenFOAM class and according to modern techniques for mesh and data handling; and the integration in the GeN-Foam multi-physics solver.

Re-implementation of FRED

The great majority of fuel performance codes (including FRED) employ a so-called 1.5-D treatment of the fuel rod. A pure 1-D treatment consists in axially discretizing each rod into several slices and solving a 1-D radial problem for each of these slices. Although this treatment is often sufficient for reactor transients, it fails in properly evaluating a longer term evolution of the fuel. The reason for this is the impossibility to predict the variation of gas pressure in the rod. To solve this issue, the 1.5-D treatment is introduced in order to axially couple the different slices by gathering the information of each slice concerning its

temperature, gap size, central hole size, and fission gas release. These information is combined with the size and temperatures of gas plena to predict the evolution of gas pressure during irradiation. In particular, the pressure in a rod is calculated as:

$$p_{rod} = \frac{R(\mu_0 + \mu_{FGR})}{\sum_{slice} \frac{V_{slice}}{T_{slice}} + \sum_{plenum} \frac{V_{plenum}}{T_{plenum}}} \quad (1)$$

where R is the universal gas constant, μ_0 the moles of gas initially the rod, μ_{FGR} the moles of released fission gases, V the gas-filled volume associated to a slice or a plenum, and T the temperature associated to this gas.

A 1-D treatment can be implemented in a PDE library like OpenFOAM by simply solving a 1-D problem for each coarse-mesh cell that belongs to the core region. The 1-D sub-scale sub-solver will take from the other sub-solvers the information about coolant temperature and pressure, fuel-clad heat transfer coefficient, fuel volumetric power and fast neutron flux, and will use this information to solve a 1-D problem for each cell. A main challenge for re-implementing FRED (or other fuel performance codes) originates instead from properly defining a rod in a general unstructured (and possibly decomposed) mesh, and from the axial coupling of the different slices of a rod.

Re-implementation of the 1D solver The re-implementation of the FRED code follows the same logic as the original implementation [8]. In particular, the equations for burnup, temperature, swelling, creep, strains and stresses are solved in a fully coupled way using a Newton method. In order to accelerate the solution and to avoid initial stability problems, a predictor has been included for the temperature field. In particular, the fuel and clad temperatures are calculated at the first iteration by employing a simple finite difference method like the one employed in the basic fuel temperature model of GeN-Foam [5]. The matrix obtained at each Newton iteration is solved using a Gaussian elimination method with pivoting.

Fuel rod representation OpenFOAM is a general library for the solution of PDEs and it employs standard algorithms for the handling of unstructured meshes. In this framework, a logical mesh for fuel performance analysis is an axially extruded mesh that is discretized in the radial core direction (X-Y plane) into a number of cells equivalent to the number of rods that one wishes to simulate. In this way, the solver will gather from other solvers the average value of coolant pressure, coolant temperatures, etc., in each cell and simulate the behavior of a slice of an average rod with those data as input. Fig. 1 shows an example of a possible discretization strategy to simulate 6 rods per assembly for a core with hexagonal assemblies.

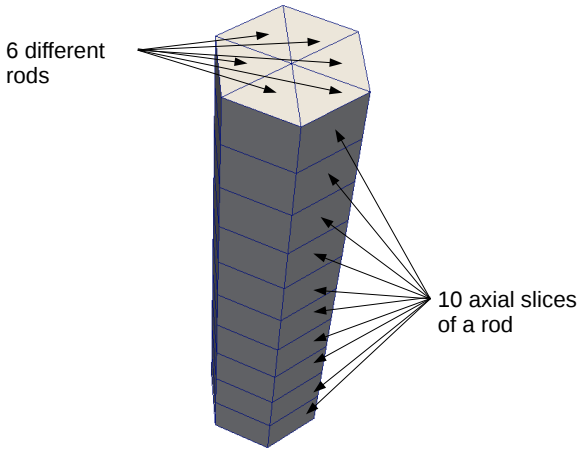

FIGURE 1. EXAMPLE OF A MESH FOR SIMULATING 6 RODS IN A HEXAGONAL ASSEMBLY

Fuel rod reconstruction In spite of the use of an apparently structured mesh, the general routines for the handling of unstructured meshes will be unaware of the fact that different cells represent axial slices of the same "average" rods. This problem can be overcome by performing, at the beginning of the simulation, a loop over all the cells in the mesh in order to identify cells (viz., axial slices) with the same X-Y position in the core and assign these cells to the same rod. In this way a map is created that associates each cell to its own rod.

Parallelization OpenFOAM employs a standard MPI-based domain decomposition technique for parallelization. According to this technique, the computational domain is split into several sub-domains. Each sub-domain is assigned to a single MPI thread that solves locally the PDE problem and iterates the solution with neighboring domains by exchanging the necessary information (e.g., fluxes) through artificial inter-processor boundaries using MPI messages.

In case of a 1.5-D fuel performance solver there is no need to exchange information between inter-processor boundaries, since there is no local exchange of information between neighboring cells. On the other hand a strategy is needed for gathering the information necessary to evaluate the evolution of gas pressure in each rod. In particular, for each rod one has to gather the terms $\frac{V}{T}$ and the released fission gases μ_{FGR} to be used in Eq. 1.

In case every rod is entirely included in one single sub-domain of the decomposed mesh (Fig. 2), the procedure is trivial and consists into looping over all the axial slices of each rod. This operation is made possible by the cell-rod map that is created at the beginning of the simulation. However, domain decomposition is typically achieved via semi-automatic routines that can cause single rods to be split into multiple sub-domains of the decomposed mesh (Fig. 3). In this case, every single sub-

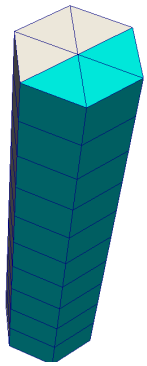

FIGURE 2. EXAMPLE OF A DOMAIN DECOMPOSITION WITH ONLY ENTIRE RODS FOR EACH DOMAIN

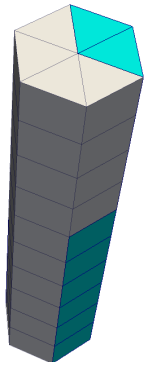

FIGURE 3. EXAMPLE OF A DOMAIN DECOMPOSITION WITH SINGLE RODS SPLIT IN MULTIPLE DOMAINS

domain will contain portions of different rods. These partial rods would be unaware of their missing portions and would be treated as entire (short) rods unless the information about their missing portions (contained in other sub-domains and MPI threads) is made available via MPI messages. To solve this issue, the following algorithm has been implemented:

1. Each MPI thread calculates the number of rods contained in its sub-domain.
2. A list is created with a size equal to the number of sub-domains (and MPI threads) and containing the number of rods for each sub-domain.
3. The total number of rods in all the sub-domains is calculated via appropriate MPI calls. It should be noted that this does not correspond to the total number of simulated rods in the reactor, since rods that are split in multiple sub-domains will be counted multiple times.
4. Three matrices are created for gathering information about the $\frac{V}{T}$ terms, the fission gas release, and the X-Y position associated to each cell in the overall computational domain. These matrices will have a number of rows equal to the number of sub-domains and a number of columns equal to the total number of rods previously calculated. The two matrices are filled by each MPI thread with information related to its own sub-domain.
5. By using appropriate MPI calls, the information gathered by each MPI thread for its own domain is scattered/gathered to all others MPI threads. Each MPI thread will now have a complete set of information about the $\frac{V}{T}$ terms and the released fission gas μ_{FGR} in the whole computational domain, and about their radial position (and thus the rod they belong to).
6. Every MPI thread performs a loop over the rods it contains and employs the previously built matrices to gather the $\frac{V}{T}$ terms and the released fission gas μ_{FGR} that belongs to the portions of its rods in other sub-domains;

In the current implementation, this procedure is performed once at the beginning of each time step, which results in an explicit coupling between the solution of the 1-D fuel performance problem and the solution for the gas pressure in the rods. This is considered to be a sufficiently accurate coupling scheme in view of the typically slow change of gas pressure. An additional coupling loop would be needed for specific transients involving a strong coupling between fuel/clad behavior and gas pressure (viz., ballooning).

According to the implemented algorithm, the time required by each MPI thread to solve the fuel-performance problem in its own sub-domain will essentially be proportional to the number of cells (i.e., rod slices) in the sub-domain. Limited parallel inefficiencies are expected with a growing number of sub-domains since: 1) the information that is exchanged via MPI calls is limited and reduces proportionally with the number of cells, which strongly limits the MPI load; 2) the number of iterations for the rod pressure coupling is set to one and it does not grow with a growing number of sub-domains. It is worth noting that this is not the case for standard PDE solvers that solve for 3-D problems and that employ a domain decomposition technique. In these cases, the amount of information exchanged between MPI threads is significant (typically, one value for each inter-processor face and for each quantity solved for) and its relative weight grows with the number of sub-domains. In addition, iteration needs to be performed between the different domains, and the number of iteration to achieve a target convergence grows with the number of sub-domains.

INTEGRATION INTO GeN-Foam

The developed FRED-based class has been included in the GeN-Foam solver. It interacts with other solvers as follows: it takes the values of coolant temperature, coolant pressure, coolant-clad heat transfer coefficient, and fuel volumetric power from the thermal-hydraulic sub-solver; it solves for burnup, temperatures, stresses and strains in fuel and clad; it provides back to

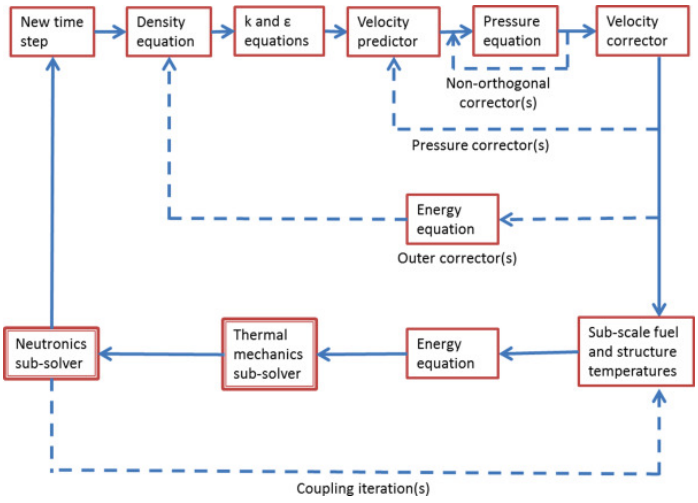

FIGURE 4. COUPLING SCHEME OF GeN-Foam

the thermal-hydraulic sub-solver the power flowing from the fuel to the coolant; and it provides to the neutronic sub-solver temperature and expansion of fuel and clad. The coupling scheme follows the same logic employed for the existing simplified solver for fuel and clad temperatures [5] and reported in Fig. 4 .

RESULTS AND DISCUSSION

The correct implementation of the developed solver has first been verified by comparing results for a single rod with the results obtained with the original FRED implementation, which has shown identical results. After this sanity check, the solver has been applied for a preliminary analysis of the European Sodium Fast Reactor (ESFR) [9], with the main objective of testing the solver for a large scale problem, and including a coupling with the other sub-solvers. The performances of the solver in terms of computational requirements and parallel scalability have then been investigated.

Application to the ESFR

The developed solver has been employed on a significant test case by employing the ESFR core. As far as the neutronics and thermal-hydraulics are concerned, the same computational assumptions as in Ref. [5] have been made. As regards the fuel behavior, 6 rods have been simulated for each assembly (corresponding to 2700 rods in the core), with 10 axial slices per rod (Fig. 1). The mesh employed for the simulation is reported in Fig. 5. For each cell in the core, the corresponding 1-D sub-scale FRED model is solved using a relatively rough discretization of 10 nodes.

Figs. 6, 7 and 8 report an example of results obtained at the end of a 6 months irradiation period. Thanks to the use of a modern library like OpenFOAM it was possible to post-process

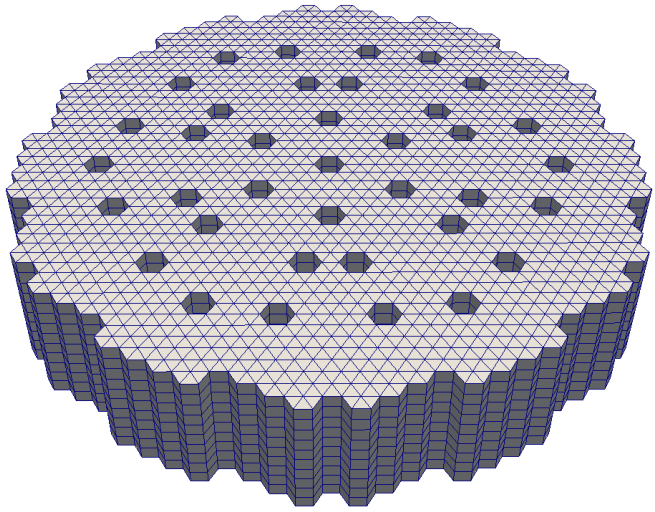

FIGURE 5. MESH EMPLOYED FOR SIMULATING THE ESFR FUEL

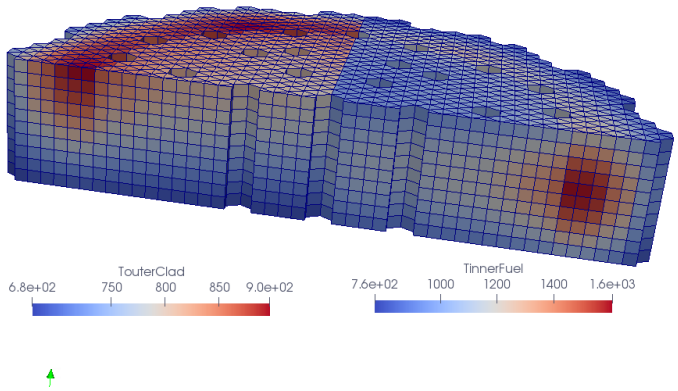

FIGURE 6. TEMPERATURES [K] OF OUTER CLAD SURFACES AND OF FUEL CENTRAL POINT IN THE ESFR AFTER 6 MONTHS OF OPERATION

the results with a readily available and powerful tool like paraview [10], which represents an important asset of the developed solver compared to the previous FRED implementation or other legacy solvers that provide results exclusively as text files.

Performances and scalability

The time required for a solution clearly depends on the number of simulated rods, their axial discretization, and the sub-scale radial discretization employed for fuel and clad. The objective of this paragraph is to provide a preliminary overview about the computational requirements expected for the developed solver, and its scaling behavior. For these tests, a standard workstation equipped with two 12-cores Intel Xeon E5-2650-v4 processors

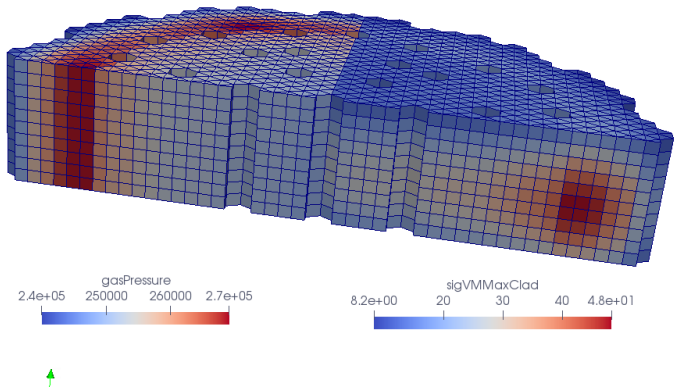

FIGURE 7. GAS PRESSURE [Pa] AND MAX VON MISES STRESSES [MPa] IN THE CLAD IN THE ESFR AFTER 6 MONTHS OF OPERATION

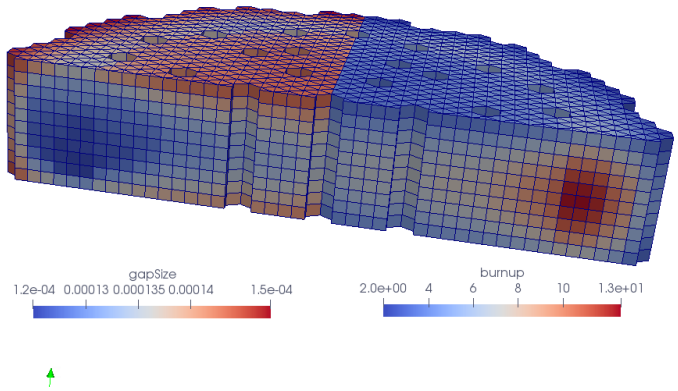

FIGURE 8. GAP SIZE [m] AND BURNUP [MWD/kg] IN THE ESFR AFTER 6 MONTHS OF OPERATION

has been used.

As a first trivial test, computing time has been evaluated for an increasing number of simulated rods, from 1 to a 1000. Computing time has been observed to grow linearly, which is consistent with the fact that rods and slices are solved sequentially by the solver.

As a second, less trivial test, the time required for a single time step iteration has been evaluated with a varying number of sub-scale radial nodes in fuel and cladding. In particular, the nodes employed in the fuel have been varied from 8 to 64, with the nodes in the cladding correspondingly increasing from 2 to 8. Fig. 9 shows that the growth in computing time is slightly less than cubic, with the cubic growth representing a theoretical limit for the Gauss elimination method for matrix solution.

The last test that has been carried out for a preliminary assessment of the solver performances is related to its parallel scalability. As discussed previously, an essentially linear scalability (computing time inversely proportional to the number of MPI

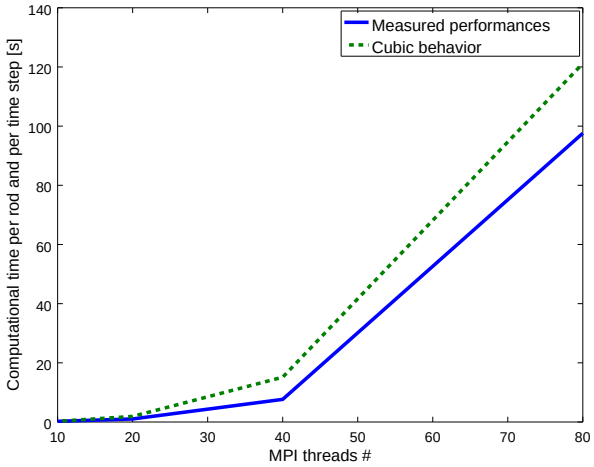

FIGURE 9. SOLUTION TIME FOR A SINGLE ROD FOR A VARIABLE NUMBER OF RADIAL NODES

threads) is expected, since the domain decomposition will divide upon the different MPI threads the overall number of slices to be solved for. However, a possibly growing MPI load and imbalance may negatively impact the solution time. To investigate these issues, a weak scalability test has been performed by running 20 time steps of $5 \cdot 10^4$ seconds for the ESFR core, with 8 radial nodes for fuel and 2 for cladding. The number of MPI threads (and decomposed sub-domains) has been varied from 1 to 32. Results are shown in Fig. 10, which shows an essentially linear scaling and excludes major implementation problems or an excessive load for MPI communications. By deeper investigation, it was observed that a main source of scaling inefficiency is related to the MPI imbalance that arise from the subdivision of the computing domain in sub-domains of non-equal size. As a consequence, the MPI threads are forced to periodically wait for the MPI thread with the highest number of rod slices to finish its computation. This causes a non-optimal utilization of resources and a less-than-linear scaling. MPI communication has also been observed to have a non-negligible impact, which will be the subject of future studies for the authors.

From the preliminary results reported above, it is possible to roughly estimate the computational time required for a given problem, and for a given computational infrastructure as:

$$\frac{time}{time_{ref}} = \frac{\#rods}{\#rods_{ref}} \frac{\#silces}{\#slices_{ref}} \frac{(\#nodes)^3}{(\#nodes_{ref})^3} \frac{\#MPIthreads_{ref}}{\#MPIthreads} \tag{2}$$

where ref indicate a reference case. For the case here considered of a large SFR and 6 simulated rods for each assembly (for a total of 2700 rods in the core), a single-core computing time

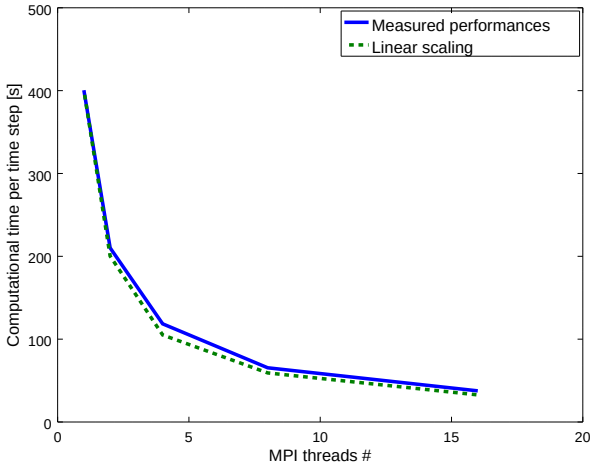

FIGURE 10. PARALLEL SCALABILITY FOR THE ESFR CASE

of 6 months can be estimated for a standard radial discretization of fuel and cladding into 25 and 5 nodes, respectively, and for a typical 1000 time steps (one per day for approximately 3 years of irradiation). This confirms the prohibitive computing time for single-core runs of fuel performance codes. However, and thanks to the scalability of the developed solver, an acceptable computing time of approximately 1 week can be estimated when using a standard 24-cores workstation. In case the good scaling behavior of the solver were confirmed for a large number of cores, the computing time would reduce to half a day on a standard 300-cores cluster, or to several tens of minutes for HPC computations involving several thousands of cores.

CONCLUSIONS

In this paper, the re-implementation of the FRED fuel performance code as an OpenFOAM class has been presented. The objective is to obtain an implementation of FRED which could easily be coupled with solvers for other "physics" and that would feature a good parallel scalability for tackling modern full-core multi-physics problems.

The newly implemented solver has been successfully benchmarked with the original FRED version, which proved a correct re-implementation of the solver, with the added value of allowing the use of advanced post-processing tools like paraview. In addition, the performances of the solver have been tested, which allowed to verify its scalability and the possibility to perform full core coupled simulations of a reactor core in reasonable time scales.

Although preliminary, the work discussed in this paper represents a first promising step towards the feasibility of full-core implicitly-coupled multi-physics simulations of nuclear reactors including detailed fuel performance analyses. In view of the promising results here presented, future work will be dedicated to testing the solver on large HPC infrastructures, and to carry out proper validation and verification.

ACKNOWLEDGMENT

The research leading to these results has received partial funding from the Euratom research and training programme 2014-2018 under grant agreement No 754501.

REFERENCES

[1] Gaston, D., Newman, C., Hansen, G., and Lebrun-Grandi, D., 2009. "Moose: A parallel computational framework for coupled systems of nonlinear equations". *Nuclear Engineering and Design,* ***239***(1), pp. 1768–1778.

[2] OpenFOAM, 2018. OpenFOAM. See URL `http://www.openfoam.org`.

[3] SIG, 2018. OpenFOAM nuclear special interest group. See URL `https://openfoamwiki.net/index.php/SIG_Nuclear_Simulations`.

[4] Fiorina, C., and Mikityuk, K., 2015. "Application of the new gen-foam multi-physics solver to the european sodium fast reactor and verification against available codes". In Proceedings of the ICAPP 2015 Conference.

[5] Fiorina, C., Clifford, I., Aufiero, M., and Mikityuk, K., 2015. "Gen-foam: a novel openfoam based multi-physics solver for 2d/3d transient analysis of nuclear reactors". *Nuclear Engineering and Design,* ***294***, pp. 24–37.

[6] Fiorina, C., Kerkar, N., Mikityuk, K., and Pautz, A., 2016. "evelopment and verification of the neutron diffusion solver for the gen-foam multi-physics platform". *Annals of Nuclear Energy,* ***96***, pp. 212–222.

[7] Fiorina, C., Hursin, M., and Pautz, A., 2017. "Extension of the gen-foam neutronic solver to sp3 analysis and application to the crocus experimental reactor". *Annals of Nuclear Energy,* ***239***, pp. 419–428.

[8] Mikityuk, K., and Shestopalov, A., 2011. "Fred fuel behaviour code: Main models and analysis of halden ifa-503.2 tests". *Nuclear Engineering and Design,* ***241***, pp. 2455–2461.

[9] Fiorini, G., and Vasilev, A., 2011. "European commission 7th framework programme: The collaborative project on european sodium fast reactor (cp-esfr)". *Nuclear Engineering and Design,* ***241***, pp. 3461–3469.

[10] Ahrens, James, Geveci, Berk, Law, and Charles, 2005. *ParaView: An End-User Tool for Large Data Visualization*. Elsevier.