PLEIADES: a numerical framework dedicated to the multiphysics and multiscale nuclear fuel behaviour simulation

Stephane Bernaud, Isabelle Ramiere, Guillaume Latu, Bruno Michel

Highlights

PLEIADES: a numerical framework dedicated to the multiphysics and multiscale nuclear fuel behaviour simulation

Stéphane Bernaud, Isabelle Ramière, Guillaume Latu, Bruno Michel

- Introduction of the PLEIADES framework, dedicated to nuclear fuel behavior simulation
- Structuring and computational choices of the PLEIADES framework are described
- The multiphysics coupling is solved through an accelerated block fixed-point strategy
- Multiscale global/local coupling and computational homogenization are also available
- Parallel PLEIADES features based on MPI communications are detailed

PLEIADES: a numerical framework dedicated to the multiphysics and multiscale nuclear fuel behaviour simulation

Stéphane Bernaud[a], Isabelle Ramière[a,*], Guillaume Latu[a], Bruno Michel[a]

[a]*CEA, DES, IRESNE, SESC, Saint-Paul-Lez-Durance, 13108, France*

Abstract

The aim of this paper is to introduce the PLEIADES framework offering a set of services and numerical tools to model and simulate the behavior of nuclear fuels of different concepts of reactors. The framework provides in particular the following features: interfaces to manipulate meshes and fields, services to deal with different physical solvers, setup of coupling trees to realize multiphysics partitioned (accelerated) fixed point couplings, automated time-marching algorithm, checkpoint/restart strategies, capability to realize on-the-fly multiscale couplings. It is built with a permanent concern for sustainability, scalability and maintainability. PLEIADES framework supports multidimensional simulations, typically 1D, 2D and 3D, possibly multilayered. To date, this framework relies on the generic thermomechanical finite elements solver Cast3M to deal with partial derivative problems (mechanical, thermal or diffusion problems) at the scale of the structure or the (heterogeneous) microstructure. It also makes use of so-called point models (mainly based on ODE–ordinary differential equations) to describe the local (mesoscale) evolution of the material through the physics of irradiation.
Several software applications are built on the PLEIADES framework, they can use its parallel features to use multiple processors. The multilayered calculation scheme provides a way to loosen the computations and access to a good parallel performance up to hundred cores. It is also possible to achieve efficient scalable concurrent multiscale simulations through finite element square (FE^2) computational homogenization algorithms.

*Corresponding author, isabelle.ramiere@cea.fr

Keywords: Fuel simulation, Software framework, Multiphysics, Multiscale, Multidimensional, Parallel computation

1. Introduction

Numerical simulation is an essential tool for fundamental sciences, industry and defense. It makes it possible to predict the behavior of complex objects, to understand phenomena that are difficult to access for which experiments in real conditions are very expensive, or even impossible. This is particularly the case for R&D on nuclear fuels. The interest in performing advanced fuel performance simulations becomes more and more significant. These type of simulations nowadays combine multidimensional, multiphysics and also sometimes multiscale capabilities. Several codes have been developped along this line such as BISON [1] and MARMOT [2], ALCYONE [3] and GERMINAL [4], DIONISIO [5], and more recently OFFBEAT [6]. In order to have the required computer and numerical capabilities, these codes are generally based on software frameworks: MOOSE [7, 8] for BISON and MARMOT, PLEIADES [9] for ALCYONE and GERMINAL, and OpenFOAM [10] for OFFBEAT.

The MOOSE (Multiphysics Object Oriented Simulation Environment) framework is a flexible (high abstraction level) HPC development environment for developing multiphysics applications. It provides an object-oriented interface that allows scientists and engineers to describe all aspects of a physical system relevant to numerical modeling. It is mainly based on a C++ finite element framework (open-source oriented), where the multiphysics coupling is generally solved through the so-called Jacobian-free Newton Krylov (JFNK) numerical method [11, 12]. This method avoids forming the full Jacobian matrix in a Newton solving approach so as to effectively save memory. To comply with the mathematical structure required by JFNK, MOOSE's physical expressions are modularized into 'kernels' that are required to supply a residual [7]. Hence, the JFNK strategy is rather structuring with respect to the interface/API of the physical components.

In the open-source C++ library OpenFOAM, governing equations are discretized with modern finite volume techniques using unstructured meshes. OpenFOAM is able to tackle 2D and 3D modelling. Its encompasses a C++ toolbox for the solution of PDEs and for the computation of multi-physics problem. Even solid mechanics problems are solved using the finite volume

method, which is a novelty compared to the computational solid mechanics community. The OpenFOAM framework includes an extensive list of features which can significantly accelerate the development of new simulation tools. Moreover, several sparse linear solvers (e.g. conjugate gradient, multigrid, GMRes, etc.) and preconditioners are accessible within OpenFOAM that are fully parallelized.

The PLEIADES (French acronym for Plateforme Logicielle pour les Éléments Irradiés dans les Assemblages, en Démonstration, en Expérimentation ou en Service) framework offers the main advantages of an object-oriented framework: modularity and easy integration of new modelling. It includes common services such as field access, time-marching management, Finite Element solver Cast3M [13], multiphysics and multiscale coupling algorithms, TFEL library [14] (mathematical and material knowledge library) including the open-source MFront tool [15, 16] for material properties and behavior laws. The modularity of the framework allows one to share modelling features between different applications of the PLEIADES platform (physical models, coupling scheme, post-processing, etc.). This originates from the adoption of common standards for: expected API of the underlying models, shared low-level API including field access, managing formatted input dataset.

Compared to MOOSE or OpenFOAM, the PLEIADES framework provides tools focused on fuel element simulation. PLEIADES comes with integrated specific multiphysics and multiscale coupling schemes as well as computational homogenization features dedicated to fuel modeling. On the other hand, as MOOSE and OpenFOAM, PLEIADES offers multilayered domain representation, mesh description with named selection also called multimesh capabilities (including typically fuel, gap and clad zones), extended mechanical and thermal analysis features.

The parallel features of PLEIADES mainly rely on the two possible multiscale descriptions of the fuel element. First, a layer-based[1] computational scheme is proposed that permits concurrent computations in multiple processes on several CPUs. Each process is typically responsible from one up to several layers. Moreover, in case of advanced heterogeneous microstructure analyses, typically via a multiscale FE^2 solution scheme [17], a nested parallelization strategy has been set up. It adds to the first parallelization level a

[1] a layer representing a partition of the domain

second one that distributes also the microscale calculations for a given layer. In this setting, a layer can then be handled by several processes (while it would be only one process at most in the first strategy). PLEIADES framework also handles automatic time-stepping facilities, checkpoint/restart capabilities as well as multidimensional (1D, 2D, 3D) modelling, possibly mixed thanks to the multilayered description. Compared to MOOSE, it is not possible to state the PDE using a high-level syntax and a variational statement. Instead of that, a set of specialized components dedicated to nuclear modelling are provided. One can combine them easily even if they are not based on the same discretization method (Finite Element or point models typically), but it is also possible to add new components. Hence, the PLEIADES framework provides a capacity to simulate the behaviour of nuclear fuel elements that is flexible, robust and parallelized.

Let us underline some software (possibly industrial) applications of CEA and EDF that are tightly connected with PLEIADES framework. Typically, each application is dedicated to a given concept of fuel element or modelling: ALCYONE [3, 9] (advanced multidimensional code) and CYRANO [18, 19] (industrial code) for Pressurized Water Reactors (PWR) fuel rods, GERMINAL [4, 20] for Sodium Fast Reactors (SFR) fuel pins, MAIA for Material Testing Reactor (MTR) fuels [21], ATLAS for TRISO fuels, LICOS [22] for innovative fuels or experimental setups and MEROPE [23, 24] for fuel microstructure behavior. All these tools constitute elements of the PLEIADES platform. This comprehensive set of numerical software contributes to the safety, security and performance of nuclear facilities, helps to consolidate and optimize safety and operating margins, to quantify and control uncertainties, to improve process efficiency and to reduce R&D costs through simulation.

It is worth pointing out that the PLEIADES platform has been in existence for more than twenty years (since 2003). However, the PLEIADES framework have never been described in detail in a journal publication. Hence the aim of this paper is to highlight the main features of the PLEIADES framework. This introduction has already established the scope and boundaries of the framework. The following section exhibits the main choices made to build the platform in term of mathematical schemes and computer science requirements. Section 3 gives a description of the computational framework and insights of the design and the main interfaces. Section 4 underlines the quality assurance around the PLEIADES platform. Finally, Section 5 illustrates some numerical examples based on the various tools of the PLEIADES' ecosystem.

2. Structuring choices of the PLEIADES framework

2.1. Multiphysics partitioned fixed-point algorithm

The multiphysics coupling scheme is a key element of fuel modeling. It establishes the numerical way to couple interdependent physical problems arising during irradiation situations (nominal, off-normal and accidental) but also in spent fuel management (storage) and experiments (thermal annealing, MTR situations, etc.). The multiphysics scheme proposed in the PLEIADES framework is a partitioned fixed-point algorithm. In this approach, the different physical models are solved in a sequential manner and a multiphysics loop is used to converge the whole nonlinear system at each time step (cf. Fig. 1). As each model is aware of the variables iteratively updated by the previous models, this type of coupling is known as block Gauss-Seidel fixed-point coupling.

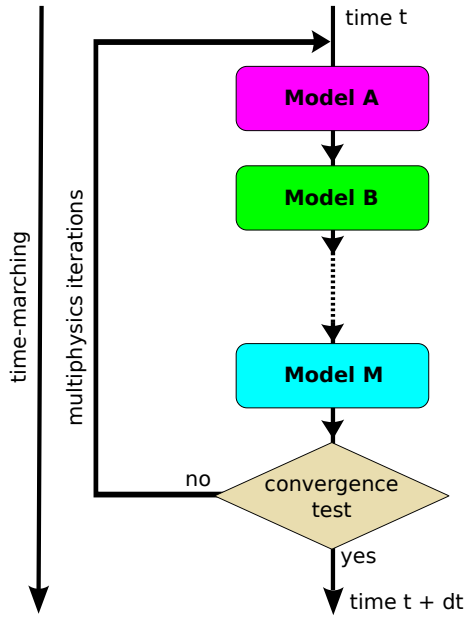


Figure 1: Schematic representation of a block Gauss-Seidel multiphysics fixed-point coupling

This multiphysics coupling strategy presents the main advantages of enabling the use on advanced single-physics solvers, being non-intrusive, easy-to-implement (plug and play) and requiring no a priori knowledge about the

multiphysics objective function (especially no Jacobian matrix is required). It is well-suited for implementation on a desktop computer, hence it requires less computation intensive kernels compared to monolithic Jacobian-based approaches (e.g. JFNK or Automatic Differentiation [25]).
However, this strategy generally leads to first-order convergence with a slow convergence speed. Moreover the convergence can often not be guaranteed. Fixed-point convergence acceleration algorithms have to be considered. The PLEIADES framework provides computer tools to easily build and call in a non-intrusive way fixed point accelerators and convergence tests. It is worth noting that the computational cost specifically attached to vector sequence acceleration methods [26] (mainly based on scalar products) is generally negligible in comparison to the cost of the evaluation of the physical solvers.

2.2. Multilayered geometric description and multiscale global/local couplings

The PLEIADES framework proposes to describe the whole geometry through a domain partition into layers (also called *slices*). Each of these layers can be modeled by a 1D (possibly axisymmetric), 2D (possibly axisymmetric) or 3D representation. In the fuel nuclear community, this geometrical assumption is classical for axial fuel rods or pins description, see for example the so-called 1.5-D representation [4, 9, 1] schematically drawn in Fig. 2, which is an axial multilayered 1-D modeling. Moreover, still for fuel rod/pin, a 3D layer may contain a part (fragment) of 3D pellet up to several stacked pellets, see for example [27].

In the PLEIADES framework, the multilayered notion is generic as it also applies on MTR fuels represented by concentric tubular layers or plate-type fuel elements.

In the computational coupling scheme, two main scales are considered: the global scale and the local scale. The global scale allows to describe the behavior of the whole fuel element by means of a partitioned coupling between the different local multiphysics resolutions on the layers. In addition some models can also be solved at this global scale, for example the calculation of the internal pressure in a rod. The local scale allows one to describe the multiphysics behavior of the layer through a partitioned fixed-point coupling as described in Section 2.1. From a physical point of view, each layer is considered as nearly independent. The physical coupling between the layers is only done through so-called global models. This approach has several advantages:

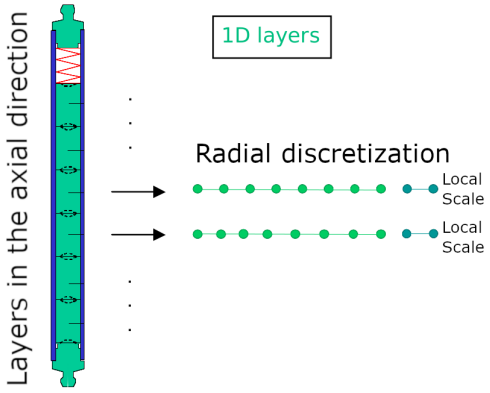


Figure 2: Illustration of a 1D multilayered description of a fuel rod.

- a heterogeneous description of the fuel element is straightforward, *i.e.* each layer can handle a dedicated combination of D models and a geometric (multidimensionnal) description,
- a parallel computation strategy based on the layers partition is quite easy to set-up.

On the counterpart, the fuel description through almost independent layers, which corresponds to a loose physical coupling between layers, implies some limitations, especially local fields discontinuity at layers interfaces.

2.3. Automatic time-stepping and checkpoint/restart

One of the main strength of the PLEIADES framework is to propose numerical tools to automatically drive the time-marching algorithm. Obviously, the traditional algorithm with *a priori* given time discretization is also available.

The automatic time-stepping is built on several ingredients, partially illustrated in Fig. 3:

- in case of convergence of all the coupling tree (global and local models):
  - a desired next time-step is proposed by each model;

  - the minimum value over all models is chosen to be the specified next time-step (see arrows in Fig. 3);
  - this specified time-step is possibly truncated to a maximal time-step imposed by the coupling tree (useful for time convergence studies for example);

- the non-convergence is detected by either:
  - reaching a maximal number of the multiphysics fixed-point iterations;
  - an internal solver non-convergence in a model;
  - detection that a model produced a non-physical solution;

- in case of non-convergence, three steps are carried out:
  - stop immediately the current time-step resolution; this is illustrated by the red crosses and the time node trials in Fig. 3;
  - split the time-step. It is possible to define the splitting algorithm; by default the new time-step to be tried is chosen as the minimum value between the reduced time-steps specified by the models (if any) and the current non-converged time-step divided by 2, see time nodes 2 and 7 in Fig. 3;
  - restart and synchronize: all the coupling tree is then restarted with a smaller time step.

Let us mention that it is possible to indicate some expected discretization times in the input file, where the time-marching algorithm must pass through. It is a common use when irradiation data are used. In this case, these discretization times are converted to imposed time nodes and a smart modification of the next time-step prediction is proposed:

- if the specified time-step is greater than the remaining time until the next imposed time node then the time-step is adjusted so that the next calculated time node fits to the next imposed time node (see time node 12 in Fig. 3);

- otherwise if the specified time-step can not be done at least twice until the next data time node, the effective next time-step corresponds to

the half of the time remaining until the next time node (see time node 5 in Fig. 3). This modification avoids to compute possibly very few remaining time steps;

- otherwise the specified time-step from models is kept (see for example time nodes 1, 3, 4, 8, in Fig. 3).

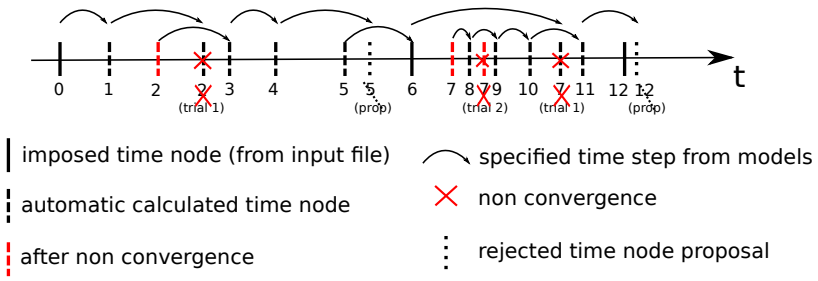


Figure 3: Schematic illustration of the automatic time-marching algorithm along a time line

This automatic time-marching algorithm associated to the multiphysics partitioned fixed-point acceleration strategy (see Section 2.1) offers a high robustness of the computation scheme. However if some recursive non-convergences occur (often due to ill-posed configurations or physical thresholds), a complete stop of the calculation is triggered when the split time-step becomes smaller than a given minimal time-step.

Furthermore, the PLEIADES framework proposes the feature of performing backups. The checkpoint time nodes have to be specified in the input data file and are then considered as imposed time nodes in the automatic time-marching algorithm. The checkpoint files enable to make calculation restarts from this time node. Note that these files are considered internal and are used for checkpoint/restart purposes only. A separate dedicated interface is provided for vizualization and data analysis (see Section 3.2.1). The checkpoint/restart capability is useful to evaluate different configurations from a given history time: end of the history, parameters values, coupling scheme (see ramp irradiations for example), etc. It also helps a lot for debugging, indeed one has access to the backup which is is automatically created in case of a complete stop due to non-convergence of the coupling tree (minimal time-step reached).

2.4. Hybrid computational homogenization

Various nuclear fuel materials have a heterogeneous microstructure. Let us mention for example MOX (Mixed OXyde) fuels with a matrice-inclusion type microstructure with plutonium agglomerates in a fuel matrix or TRISO fuel particles made up of different layers.

In standard computation schemes, the fuel heterogeneous behavior is taken into account through homogenized laws given effective properties. These approximations do not provide detailed local information, at best average estimates (moments of order 1 and 2) per phase for the so-called "mean-field" approaches [28, 29]. Morever, they require major developments (sometimes impossible) for some complex non-linear behaviors (creep, cracking, etc.) [30].

It then appears interesting to assess the nonlinear mechanical or thermal behavior at the microstructure scale, especially when so-called analytical homogenization approaches (with closed-form expression) are not available. This can been done through computational homogenization methods, which are based on obtaining the effective behavior on-the-fly, during the structure calculation, from microscopic calculations. In the PLEIADES framework, this feature is available. We have chosen to rely on a multiscale finite element square (FE^2) approach [17] which consists in calling heterogeneous microscale finite element computations on a Representative Volume Element (RVE) at each integration point of the macroscale finite element scheme, see Fig. 4. Due to the assumption of scale separation, the transfer of information between the different scales is achieved through homogenized quantities. The so-called localization step corresponds to the macro-to-micro mapping and aims to define RVE boundary conditions from the macroscopic strain tensor (at the corresponding integration point). The homogenization step, *i.e.* the micro-to-macro mapping, consists in setting the effective (homogenized) stress tensor as the spatial average over the RVE of the microscopic stress tensor. This full-field approach involves the numerical coupling of the two modeling scales in the same calculation scheme and thus make it possible to have average (effective behavior) and complete information on the local state of the microstructure. The only material laws to be provided are those of the microscopic scale.

The weak point of the FE^2 method is related to the computation time and memory space (storage from one time step to another of the equilibrium state and the internal variables of each RVE calculation) that it requires. In spite of the obvious parallelization of the algorithm, the additional cost

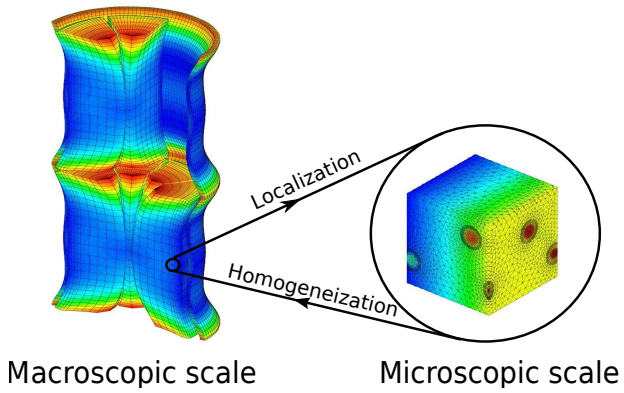


Figure 4: Finite element square computational homogenization for heteregeneous nuclear fuels

in computation time is increased by a multiplicative factor ranging from 10 to 1,000 (depending on the number of degrees of freedom and the computer resources) than the usual FE algorithm involving homogenized behavior laws.

We have therefore introduced in the PLEIADES framework a hybrid FE^2 approach [31, 32] consisting in calling the FE^2 model only in one or several regions of interest and a homogenized constitutive equation model in the rest of the computational domain. The local calculations remain independent of each other and can therefore always be parallelized. The macroscopic effective behaviour field (typically the stress field or the thermal flux field) is thus the composition of values coming from the two approaches. The resolution of the global macroscopic equilibrium allows the solution field (displacement or temperature) to take into account both approaches. The obvious advantage of this hybrid FE^2 approach is a gain in computation time and memory requirement compared to the standard FE^2 approach while still having complete information on the local state on strategic areas of the domain. On the other hand, it is necessary to have homogenized behavior laws at the macroscopic scale representative of the effective behavior in the complementary zone of the domain. Let us underline that the areas of interest can be automatically determined through an error indicator as proposed in [33].

In practice, in order to be as little intrusive as possible in the existing nonlinear FE solver, we have made in PLEIADES the algorithmic choice described in Algorithm 1 for this hybrid FE^2 strategy.

This algorithm only requires to be able to control the number of iterations

Algorithm 1: Low-intrusive implementation of the hybrid-FE^2 method

```
for k = 1 until convergence do
    Perform an iteration of the nonlinear FE algorithm on the
     macroscopic scale (involving a macroscopic residual as RHS).
    At each integration point in the area(s) of interest:
        Call the FE^2 model involving a (nonlinear) FE resolution on
         RVE
        Replace the effective behavior at this point by the one
         obtained via the FE^2 model
    Update the macroscopic residual
    Check the convergence
```

of the nonlinear FE resolution. Indeed, the residual update can be done outside the FE resolution, and the obtained residual imposed as a source term at the next macroscopic resolution.

Remark. This hybrid FE^2 strategy can be seen as an alternative to others approaches working on the intrinsic computational cost of an RVE calculation using fast microscopic solvers [34, 35] or reduced order modeling [36, 37].

3. Framework description

3.1. Overview

The PLEIADES framework proposes a C++ hierarchy of small manageable entities. The decomposition in classes has been thought to circumvent implementation issues, such as intricacies due to long compile-time, management of link dependencies, configuration management. A Python interface grants direct access to the core components of PLEIADES and allows a end user to build a full application upon it.

As PLEIADES framework relies on SALOME platform [38] for several libraries and tools (used as prerequisites), applications built on it can easily be coupled thanks to ICoCo API [39].

The initialization phase of a PLEIADES application consists of using the dataset, built upon an xml user input file, to parameterize and to instantiate the different subsystems of the framework. If needed, an initial value is given to the internal data from the inputs. The different models and computation

entities (predictors, accelerators, convergence evaluators, etc.) are instantiated in order to setup incrementally the so-called *coupling tree*. Once the initialization phase has ended, the coupling tree is finalized together with the instantiation of the fields. Afterwards, the execution phase is completely automated: the `execute()` method called on a coupling tree triggers the simulation run for the current time step until the whole coupling tree is declared converged. This process is repeated for each time step until the last one.

3.2. Multiphysics coupling bricks

3.2.1. Coupling element

The term *coupling element* refers to any entity of the computation scheme of the framework.

A coupling element is basically a software entity that receives parameters (once at initialize step) and receives and/or outputs fields when requested - see Figure 5.

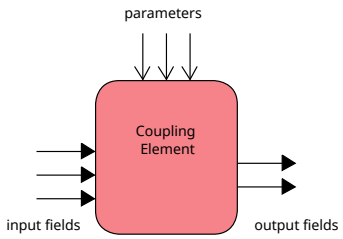


Figure 5: Dataflow of a coupling element

Different types of coupling element may exist. Therefore, the PLEIADES framework proposes interfaces that target different needs:

- `IModel`: computation of physical phenomena,
- `IPrediction`: solution predictors,
- `IAcceleration`: convergence accelerators,
- `IConvergence`: convergence evaluators,

- `IPostProcessing`: post-processing intermediate components (mainly dedicated to exporting fields or quantities for visualization or data analysis purposes).

Regardless of the type of a coupling element, inputs and outputs are always declared in the same way. Listing 1 shows the method `declareField` whereas Listing 2 shows `declareParameter`, both defined in the `ArchCouplingServices` class.

Listing 1 : declareField method

```
/*!
 * Declare a Field of a given type T.
 *
 * @code
 * declareField<vector2D>("TEMP", GlossaireField::Temperature,
↪  "CLAD", OUT)
 * @endcode
 * @param shortName local name
 * @param fieldName reference (Glossary) name
 * @param zoneName support name
 * @param fd field direction
 * @see setFieldDirection
 */
template <typename T>
void declareField(const std::string& shortName,
                  const std::string& fieldName,
                  const std::string& supportName,
                  FieldDirection fd)
```

Listing 2 : declareParameter method

```
/*!
 * Declare a parameter of a given type T whose short (local) name
 * is shortName and name is parameterName.
 *
 * if the parameter is not found during initialization step,
 * defaultValue is used.
 * @code
 * declareParameter<double>("SMAX", "SuperLongParameterName", 0.1);
 * @endcode
 *
 * @param shortName local name
 * @param parameterName parameter name
```

```
13      * @param defaultValue default value
14      */
15     template <typename T>
16     void declareParameter(const std::string& shortName,
17                           const std::string& parameterName,
18                           T defaultValue)
```

As you can see, the `std::string` type is widely used in this family of methods, to improve the user experience. We are used to identify things by their name (*e.g.* "Temperature on the cladding"). One might argue that this choice slows down searches, which is true in absolute terms, but is definitely insignificant compared to physical resolutions, especially since these lookups are never used within resolution loops.

3.2.2. Solvers

As multiple physical phenomena may require to reuse some specific solvers multiple times, the PLEIADES framework has the ability to encapsulate solvers within the `ISolver` interface. This mainly allows one to simplify (and optimize) the use of these solvers while providing a generic interfacing. The use of the `ISolver` interface is quite simple: the solver is parameterized by a series of calls to the `set()` method (inputs, outputs, parameters) while the execution of the solver is triggered by a call to the `solve()` method.

Several solvers are packaged and available: `CastemSolver` offering Finite Element thermo-mechanical solution, `CastemFE2Solver` providing modelling based on FE^2 method (see section 2.4). The DLSODA library [40] for solving ODE (ordinary differential equations) is also available within PLEIADES framework. It is mainly used in so-called point models to typically deal with the physics of irradiation.

Listing 3 shows the use of `CastemSolver` from the `execute` method of an `IModel`: PLEIADES fields and parameters are set using a `config` object and then the `solve()` method is called. Note that calling the solver during the initialization step may be relevant and can be done in a similar way.

As you can see from this listing, once fields or parameters are declared (using the methods mentioned above), they are identified by their short name (the first parameter of `declare` methods), chosen by the user himself. Thus, when defining input or output fields or parameters to be passed to / from the solver, using the `config` object, short names are used. And since C++11, the language allow to write them in a rather elegant syntax.

Listing 3 : Using `ISolver` interface

```
CastemSolverConfiguration config;
config.procName = "exec";
config.procFile = "./data/ThermalCube.dgibi";
config.inputFields = {"T", "TExt", "K", "vP"};
config.parameters = {"I", "td", "ti"};
config.outputFields = {"T", "fa"};
auto s = getSolver();
s->set(config);
s->solve();
```

3.2.3. Multiphysics coupling scheme

In the context of computer modeling, multiphysics coupling involves scheduling the different physical models one after the other, but it is also essential to manage the numerical convergence. In the PLEIADES framework, this is orchestrated by the mean of composing coupling elements (see Section 3.2.1), where the outputs of ones are used as inputs of others.

As shown in Figure 6, coupling elements all inherit `ICoupling` interface. This interface is also implemented by two final classes `GlobalCoupling` and `LocalCoupling` (as well as an intermediate base class `CouplingBase`) that act as ordered containers for any `ICoupling` (including themselves). This exhibits the *Composite* structural design pattern and, by that mean, allows one to build an entire so-called *coupling tree* as we shall see at the end of this subsection.

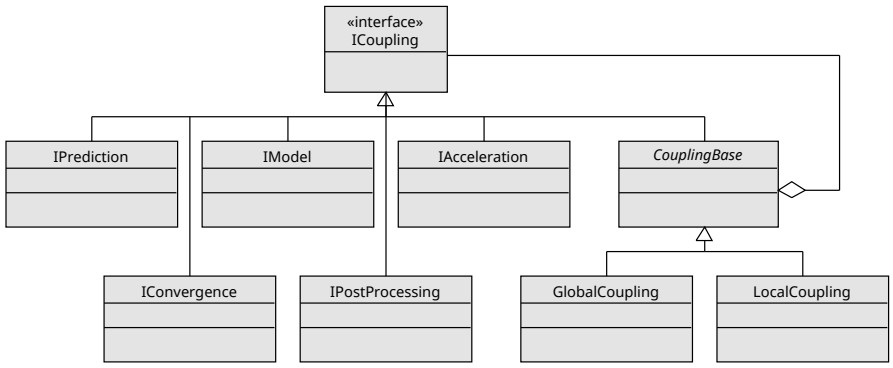


Figure 6: Simplified UML class diagram of ICoupling family

A `GlobalCoupling` gives the ability to define coupling tree's root, it also hosts coupling elements related to the entire computation scope (*i.e.* global scale introduced in Section 2.2 that is associated to the whole considered system). A `LocalCoupling` element (associated with a locality identifier), when inserted in a `GlobalCoupling`, allows one to define a computation scheme on an arbitrary narrower scope (such as the local scale defined in Section 2.2). As we shall see in Fig. 7, `GlobalCoupling` and `LocalCoupling` elements can be combined recursively in the coupling tree. This design takes advantage of the nearly independent computations at local scale and allows to spread local computations on multiple local cores or remote hosts using MPI communication library.

Figure 7 shows a simple coupling tree with a `GlobaCoupling` (`G`), a model (`A`), three `LocalCouplings` (`L1, L2, L3` on localities `1, 2, 3`) containing a (local) model (`B` or `D`) and a convergence evaluator (`C`). It is worth noting that LocalCouplings are not required to contain the same set of coupling elements. This functionality is useful when modeling heterogeneous fuel element (*e.g.* axially heterogeneous fuel pins [4]). In order to allow re-usability, coupling elements that are written (and compiled) with fields lying on a given support can be rebound at runtime by the PLEIADES framework to match support names defined in the current input mesh-support (see Section 3.3.1).

When a coupling element is added to a `LocalCoupling`, support names are also transformed according to the mesh naming convention (see Section 3.3.1) and to the `LocalCoupling` identifier (*i.e.* locality), achieving the objective to write coupling elements once and apply them anywhere.

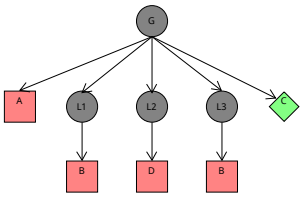


Figure 7: Basic coupling tree

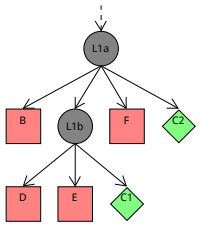


Figure 8: Zoom on an elaborate sub-coupling

By convention, the coupling tree is recursively traversed in depth-first order with the pre-order variant, from left to right. Thus, it defines precisely and uniquely the computation sequence. It is also possible to isolate some models in a sub-coupling (nested coupling) to fit convergence strategies as

in the example represented in Figure 8. In this example, once the fixed point algorithm begins in `L1a`, the Model `B` is executed, then convergence is searched in `L1b` (there are multiple local iterations) before executing the Model `F`.

The coupling tree can be defined with a C++ or Python high level interface or in a user data file. Thus, applications writers can define fixed and validated coupling scheme while numericians and modelers can independently experiment coupling scheme optimizations by quickly modifying or reordering the coupling tree.

The Listing 4 is an example of the use of the C++ high level interface to define the coupling tree represented in Figure 7.

Listing 4 : Building a coupling tree with C++ interface

```
auto coupling = createCoupling("GlobalCoupling");
coupling->addModel(createModel("A"));
auto models = std::vector<std::string> {"B", "D", "B"};
for (std::size_t i = 0 ; i < 3 ; i++) {
    auto lc = createCoupling("LocalCoupling");
    lc->setLocality(i + 1);
    lc->addModel(createModel(models[i]));
    coupling->addCoupling(lc);
  }
coupling->addConvergence(createConvergence("C"));
```

Finally, from a coupling element point a view, some available steps, corresponding to overridable methods, are illustrated in Figure 9 (not including those related to time step resizing discussed later in 3.4.1).

If the coupling element is itself a coupling (i.e. a `GlobalCoupling` or a `LocalCoupling`), the default behavior when a given step (i.e. method) is triggered is to call that step (i.e. method) sequentially for all of its coupling elements, achieving the recursive tree traversal described above.

However, this simple behavior is modified when the execute step (i.e. main computation step) is triggered because a coupling is not just a tree traversal operator. Its main task is to achieve convergence. So, the execute step is extended by some non-recursive steps consisting of numerical or state handling hooks (prediction, acceleration, convergence evaluation, etc.) and a recursive one (execute) to compute physical phenomena for coupling elements that are not couplings or to trigger a convergence loop for sub-couplings (this is where recursion occurs). The dashed rectangle in 9 illustrates the extension

of execute step for a coupling.

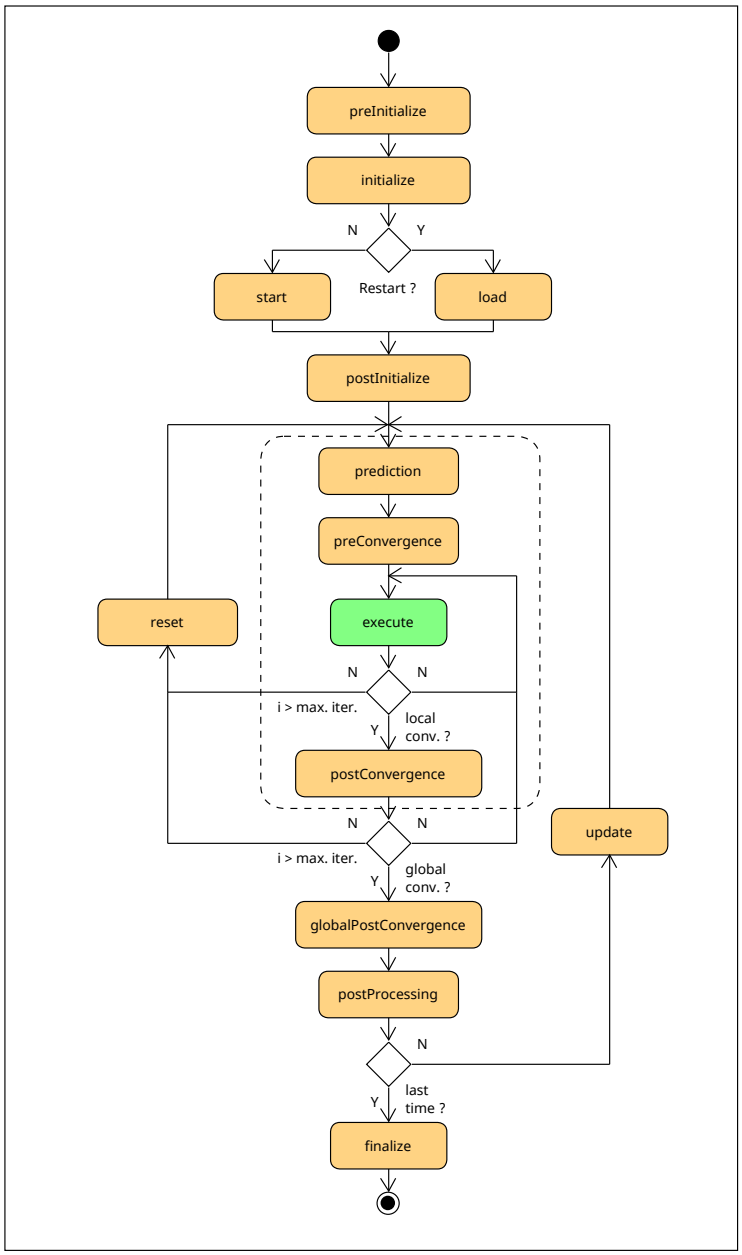


Figure 9: Computational sequence for a coupling element

3.3. Multilayered geometric description and multiscale global/local couplings

3.3.1. Spatial discretization

The `IMesh` interface conceptualizes a mesh, *i.e.* a discretization of the space in one, two or three dimensions. A mesh is identified by a name and can contain a set of named sub-meshes. Typically, each sub-mesh relies on a set of discretization nodes or elements. Typically *fuel*, *clad*, and any other zone of interest can be identified with sub-meshes.

Each PLEIADES application can define its own mesh naming convention (by implementing `IMeshNamingScheme` interface) to identity zones related to the same layer (or *slice*). For example, an application can define `Fuel_1` and `Clad_1` to indicate that they belong to the same first layer. Let us underline that the mesh dimension (1D, 2D or 3D) can vary between layers of a same fuel element description. Combined with the facilities of the LocalCoupling scheme (see Section 3.2.3), this framework offers great flexibility for designing fuel simulations.

Each field quantity is defined on a given mesh-support, it represents a discrete description of a physical quantity over a spatial domain. The size of the mesh-support and the number of components of the field determine the number of values of the field to be stored, unless the field is declared as *uniform* which means that the same value applies on the whole mesh-support.

Concerning mesh objects, they can be generated within PLEIADES framework using `PleiadesMesh` facility or can be imported from an external mesher at the `med` format (the native format of SALOME platform, see 3.1).

As input data have no reason to match the spatial discretization, PLEIADES framework defines the concept of `Loading` that aggregates input data and interpolator (e.g. histogram, linear, kriging). These loadings put users at ease to explore various interpolations strategies. The Loading concept also holds for inter-layer interpolation, useful for example for any axial loading (e.g. Linear Power Profile) in a fuel rod/pin.

3.3.2. Parallelism

Two levels of parallelism are proposed to distribute local computations in the PLEIADES framework: a task-based one dedicated to have concurrent computations associated to each layer and a data-based one in order to define a parallel computational homogenization algorithm (see Section 3.5). Those parallelism levels can be nested, they benefit from a fine control over resource allocation at each level.

The scattering of layers essentially involves implementing a parallel recursive tree traversal with data transfers at carefully selected locations and stages.

Figure 10 illustrates how the simplified coupling tree of Figure 7 is assigned to multiple MPI processes. The communication layer between global and local computations in represented by a blue line. As each coupling element declares its own inputs and outputs, dependencies are known, the framework (thanks to the `SynchronizedFieldsManager` class) is able to compute the required data (*i.e.* fields) to be transferred between processes.

Note that the PLEIADES framework is able to spread multiple local computations per MPI process, allowing one to take advantage of available resources (from a laptop to a cluster).

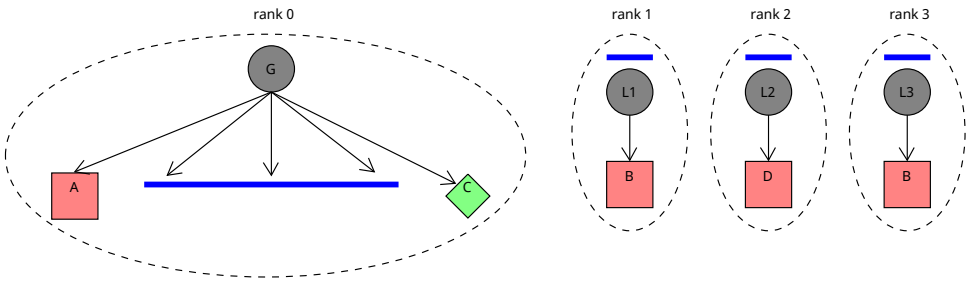


Figure 10: Parallel Coupling Tree for the basic tree of Fig. 7

The parallel algorithm uses the master-slave paradigm.

The structure of the coupling tree is known to all MPI processes (real coupling elements are replaced by placeholders if they are not to be computed by the current process). This allows sharing and reuse of recursive tree traversal code and makes synchronization points to explicitly appear at the same place in the coupling tree for master and slaves. Note that communications always take place between `GlobalCoupling` (i.e. master) and `LocalCoupling` (slaves). `LocalCouplings` do not exchange data and do not synchronize directly between themselves.

The communication protocol is therefore fairly simple. When a synchronisation point is reached, the global scale (master) put itself in listen mode, awaiting messages from local scales.

The mapping of couplings to MPI ranks is defined in implementations of the `ILocalityToSlaveMapper` interface. The default implementation dis-

tributes `LocalCouplings` to MPI nodes using a round robin algorithm and books a dedicated process (rank 0) for the global scale. This master/slave design pattern tends to favor communication efficiency over CPU usage for the master. Note that this underutilization becomes relatively less crucial as the number of compute units increases.

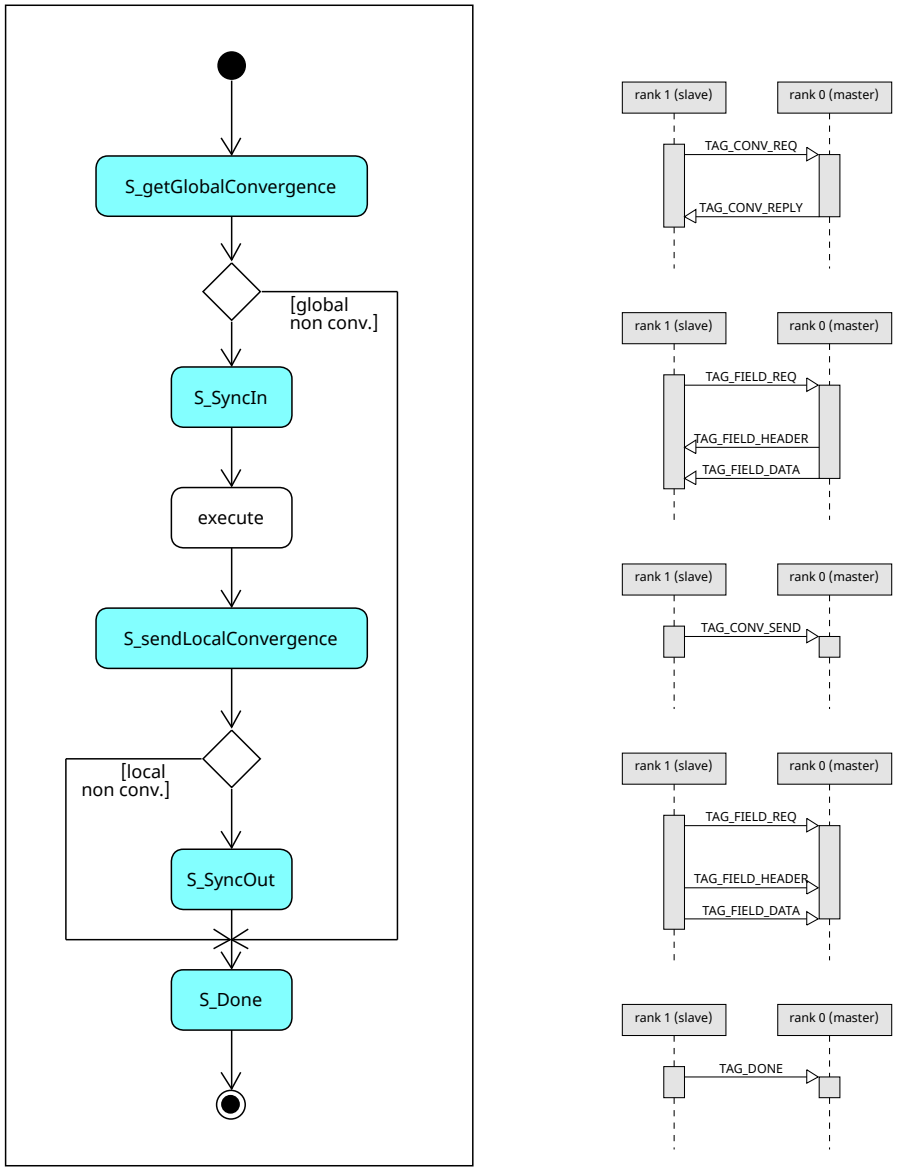
Nevertheless, this design responds to the need of spreading hundreds of localities (or thousands of micro structures for EF^2 computations) to so many MPI processes, but it's clearly not designed to achieve massively parallel computations. The first obstacle to overcome is the centralisation of data in memory (essentially the mesh description) at the global scale. Each exchange between master and slaves is then identified by a specific MPI tag. The Figure 11 shows the communication protocol used (from the local scale point of view) when a transition occurs from global scale to local scale.

The Figure 11a shows the activity diagram of a remote `LocalCoupling` (the one at the global - local transition) around its execute step. As we can see, the coupling retrieves information concerning the global convergence status and directly steps to done if a non convergence is detected elsewhere. Otherwise, the `LocalCoupling` retrieves necessary fields from the master, achieve its local (and potentially recursive) execute step before sending its own convergence status and the necessary modified fields to the master.

The Figure 11b shows how the MPI tags are used for each communication step between a remote `LocalCoupling` (slave) and the `GlobalCoupling` (master). Each protocol is facing the step it corresponds. Thus, the top most protocol in Figure 11b corresponds to the step it is facing in Figure 11a, i.e. `S_getGlobalConvergence`.

Several MPI routines are used to implement these behaviors. When the master is in listen mode, it uses `MPI_Probe` to receive arbitrary tags (`MPI_ANY_TAG`), from arbitrary source (`MPI_ANY_SOURCE`) and then calls a dedicated method corresponding to the selected protocol. For example, if the received tag is `TAG_FIELD_REQ`, the master sends two messages (using `MPI_Send`): the first one with the tag `TAG_FIELD_HEADER` to provide field lengths and uniformity status, the second one with the tag `TAG_FIELD_DATA` to transmit field values.

One of the challenge with a master-slave architecture is that some bottleneck may happen at the master side. Indeed, the master manages the allocation of the tasks and gathers the slave results. If the number of slaves gets too large or if there are frequent exchanges, the master may struggle to manage incoming requests or dealing with receiving/processing the results



(a) Activity diagram of a remote LocalCoupling

(b) MPI protocol

Figure 11: MPI activity and protocol

provided by the slaves. Also, network latency and bandwidth may have an impact on the system performance depending on the granularity of the compute tasks and the data to be transferred. In practice, with typically a few

hundreds of slaves, we did not encounter any of these issues.

3.4. Automatic time-stepping and checkpoint / restart

3.4.1. Temporal discretization

The `IRunner` interface defines a contract for a set of methods that should achieve simulation runs for a predefined time discretization (defined in user input file). The time discretization can be strictly iterated through, as in the `DefaultRunner` implementation or considered as coarse guidelines, and resized if necessary as proposed in the `AdaptiveRunner` class.

Note that thanks to the `Loading` concept (see Section 3.3.1), input data can be used together with interpolators to match temporal discretization (as for spatial discretization). These interpolators are likely to be called during the computation.

The resizing policy is defined by implementations of the `ITimeStepResizer` interface, as for the `DefaultTimeStepResizer` which implements the policy defined in Section 2.3. Any `IModel` (see Section 3.2.1) in the *coupling tree* can provide a hint on the next time step size through the `ITimeStepHint` interface, for the standard situation (`getTimeStepHint()`) and the non-convergence situation (`getNonConvergedTimeStepHint()`).

The process of resizing time step sometimes requires to revert back in time, and to forget the failed time step effects. The PLEIADES framework resets fields values to their previous values. It is also possible for models and solvers, that store their own private data, to restore/reset internal state with the `IHoldable` and `IResetable` interfaces and their respective `hold()` and `reset()` methods.

Remark : A test mode allows users to choose a given time step to be repeated in order to verify that the whole coupling tree is stable (identical results) regarding reset data.

As stated in Section 2.3, automatic time-stepping algorithm also relies on efficient convergence anomaly detection. PLEIADES framework provides `IInternalConvergence` and `IPhysicalConvergence` interfaces to allow a model to respectively reports:

- an internal solver non-convergence,
- a non-physical solution (even if the time step is numerically converged).

Note: As already mentioned, PLEIADES framework globally fails the computation if the requested time step becomes too small (lower to 10^{-6} s by default).

Figure 12 summarizes the logical sequence close to the `execute` step (core of the computation) implemented by `AdaptiveRunner` to perform adaptation of the time step.

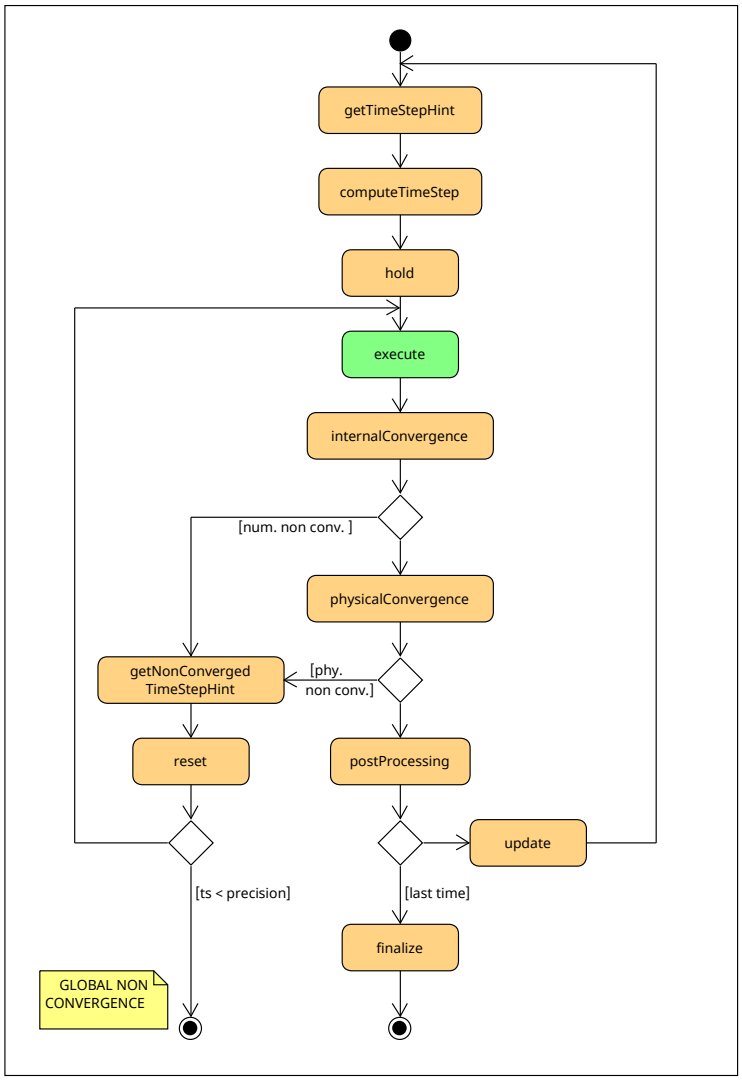


Figure 12: AdaptiveRunner activity diagram

3.4.2. Checkpoint / restart

As introduced in Section 2.3, PLEIADES framework provides a checkpoint/restart service. As the computation can be spread over multiple MPI agents (see Section 3.3.2), backup policy takes care of this. So, when a checkpoint is required (because stated in user input file or because of a global non convergence), thanks to the `PersistenceManager` class, parallel tasks are triggered: each locality saves data relevant to its scope in a dedicated file (`pleiades@<locality>.h5`) and, of course, the global level saves its data too.

The `IPersistence` interface, implemented using HDF5 functionalities [41], provides save / restore methods for basic types (`bool`, `int`, `double`, `string`, etc.) and standard containers. It allows PLEIADES classes to build their own (de)serialization upon it with the help of the `SerializationHelper` utility class.

The Listing 5 illustrates how a PLEIADES field is saved, using `IPersistence` interface.

Listing 5 : Serialization of a variable field

```
template <typename T>
void VField<T>::save(IPersistence& p) {
    SerializationHelper<std::vector<std::string>> sh;
    auto s = sh.serialize(this->getComponentNames());
    p.save("componentNames", s);

    double* d = reinterpret_cast<double*>(values_);
    p.save("nbC", this->getNumberOfComponents());
    p.save("value", d,
           this->getNumberOfComponents()
             * this->getSupport()->getNumberOfElements());
  }
```

When a previously checkpointed computation has to be restarted, the same principles apply. Also, PLEIADES framework provides various restart schemes (which are implementations of `IRestartScheme` interface):

- Strict: restart without any change in the data file
- Continue: changing parameter values is allowed
- AlternateEnd: changing parameter values, Loadings and time discretization are allowed

- Initial: restart using a geometric subset of the initial computation (ramp irradiations use case). Almost everything can be redefined except meshes.

As each locality has its own backup file, restart can be done with a modified set of MPI processes.

3.5. Hybrid computational homogenization

With hybrid computational homogenization (FE^2), another level of parallelism is available in PLEIADES framework. As it is about distributing data sets (values on integration points) across multiple computation nodes where same operations are performed (data parallelism), MPI collective routines are used.

The basic solution scheme of our thermo-mechanical solver (based on the Cast3m finite element solver [13]) is illustrated in Figure 13 and can be summarized as:

- an initialization step (`PL_INIT`),
- solution of global equilibrium (`PL_ESTIM`),
- behavior law integration (`PL_COMP`),
- convergence evaluation (`PL_CONV`),
- post-processing (`PL_SAUV`).

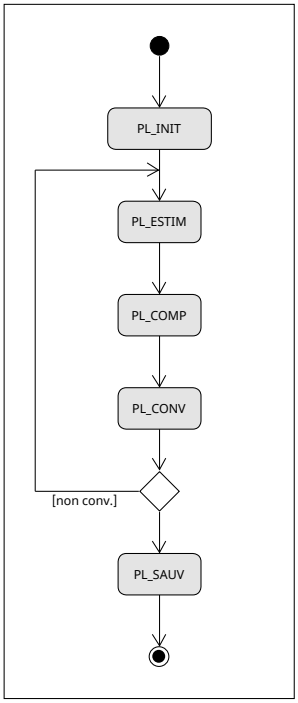


Figure 13: Thermo-mechanical solver computation scheme

For the needs of FE^2 method, as illustrated in Figure 14, this computation scheme has to be opened to send values on integration points of interest to the microstructure scale computation, after `PL_ESTIM` step and to retrieve (and merge) homogenized data after `PL_COMP` step. To minimize the amount of data to be transferred, the association between an integration point and its MPI process is kept, allowing to instantly recover internal solver state. Note that one MPI process can handle multiple integration points at the cost of a larger execution time (the optimal situation is, of course, one MPI process per integration point). This allows one to use standard hardware (mainly laptops and desktop computers) for small to medium FE^2 computations.

Finally, as illustrated in Figure 15, both parallelism (`LocalCouplings` and FE^2) can be nested. A command line parameter allows the user to inform PLEIADES framework about how to partition MPI processes. One can infer that the shortest computation time should be reached with the maximal

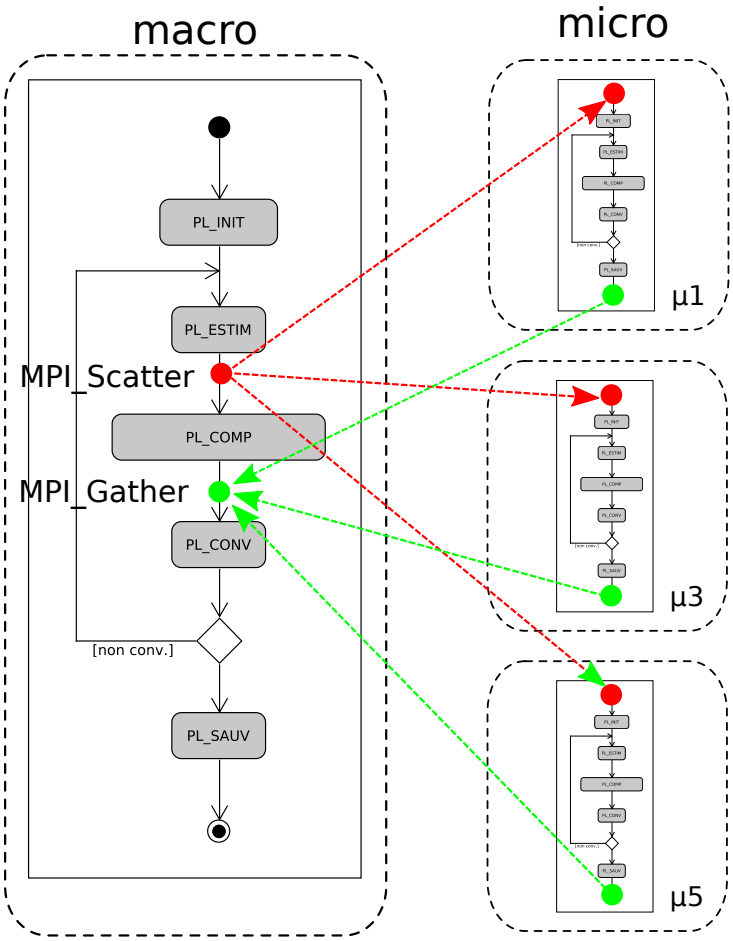


Figure 14: FE^2 macro and micro scales

number of MPI processes which is:

$$\text{MPI processes} = \text{nb. of LocalCouplings} + \text{nb. of integration points} + 1 \quad (1)$$

From the user point of view, using FE^2 computation consists in selecting `CastemFE2Solver` as a solver in a model (l.7 in Listing 6) and describe how to initialize the solver, execute it, and validate the time step.

Initialization is the most complicated step as two kinds of settings have to be defined: a set of fixed parameters that shapes the whole computation (solver procedure to be used at macro and micro scale, size of data, integration points identification, etc.) and a set of volatil parameters that define the

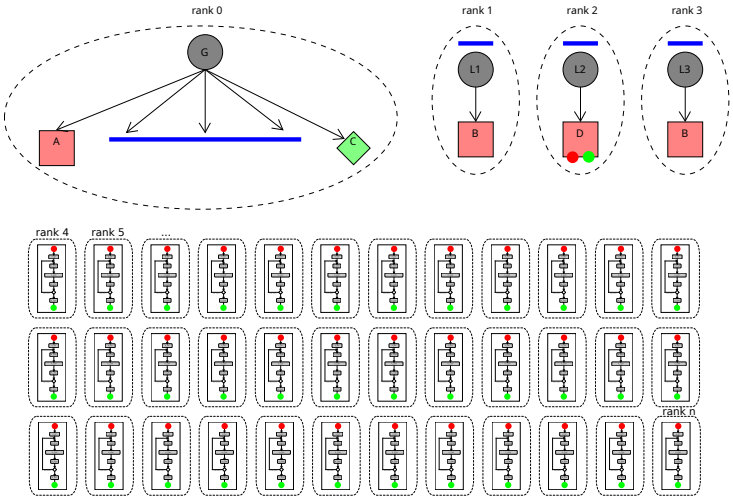


Figure 15: Mixing coupling and FE^2 parallelism

current solver call behavior. In the example, we set initialize step at micro scale (l.22) and declare that macro scale will receive `TF` an `TC` as input fields (l.23). Finally, solver execution is triggered with its `solve` method (l.25).

The execute step is much simpler as it resembles classic solver use (in PLEIADES framework), with the addition of the `FE2Step` parameter set to `EXECUTE` (l.31) to trigger the FE computation at the microstructure level for each integration point of interest on the next `solve` method call. `solve` returns when all micro scale computation has ended and all data are retrieved and homogenized. The execution flow then continues on the next coupling element.

Then, when the whole coupling tree reaches convergence, eventually after resizing the time step, the `globalPostConvergence` method is called, giving us the opportunity to inform the micro scale that the current solution can be validated and used as initial conditions for the next time step with the `FE2Step` parameter set to `COMMIT` (l.41).

Listing 6 : Model using an FE^2 solver

```
1  MyFE2Model::declare() {
2   declareField<double>("TF", Glossary::Temperature, "FUEL", IN);
3   declareField<double>("TC", Glossary::Temperature, "CLADDING", IN);
4   declareField<double>("GAPI", "GAP", "PCI", OUT);
```

```
  declareField<double>("GAPE", "GAP", "PCE", OUT);
  // declare other fields and parameters
  declareSolver("CastemFE2Solver");
}

MyFE2Model::initialize(const IArgumentMetier& arg) {
  auto s = getSolver(); // handy shortcut
  // one time configuration
  CastemFESolverPreConfiguration fixedConfig;
  fixedConfig.pl_init_MacroProc = "initialize";
  fixedConfig.pl_init_MacroFile = "init.proc";
  // declare all fixed settings

  s->preset(fixedConfig);

  // initialize micro scale
  CastemFESolverConfiguration config;
  config.FE2Step = "INITIALIZE";
  config.inputFields = {"TF", "TC"};
  s->set(config);
  s->solve(); // solver execution
}

MyFE2Model::execute() {
  auto s = getSolver();
  CastemFESolverConfiguration config;
  config.FE2Step = "EXECUTE";
  config.inputFields = {"TF", "TC"};
  config.outputFields = {"GAPI", "GAPE"};
  s->set(config);
  s->solve(); // solver execution
}

MyFE2Model::globalPostConvergence() {
  auto s = getSolver();
  CastemFESolverConfiguration config;
  config.FE2Step = "COMMIT";
  s->set(config);
  s->solve(); // solver execution
}
```

4. Quality assurance

PLEIADES-based codes are scientific calculation tools used, among others, in nuclear safety demonstrations and therefore follow recommandations of the guide 28 [42] published by the french nuclear safety authority concerning qualification of such tools, defining and highlighting the role of verification, validation and quantification of uncertainties steps.

Beyond its multiphysics, multiscale and multidimensional (1D, 2D, 3D) simulation capabilities, one of the strengths of PLEIADES platform is the wide validation database, including separate and integral effects tests. This database relies on nearly fifty years of collaboration with various industrial and academic partners: modelling, operation of reactors, experimentation and examinations in high-activity and conventional laboratories, tests on large instruments, etc. For example, the ALCYONE validation database now includes several thousand objects from operating feedback from the international reactor fleet, tests in experimental reactors, and analytical experiments. These objects are capitalized in dedicated databases with traceability between the object, the databases and the performance code. The integral validation of the ALCYONE code is based on several hundreds of study cases, with input data and post irradiation examination results stored in the experimental data base (named CRACO). Several PLEIADES applications have been benchmarked in the framework of various international collaborations, for which they have contribute to improve the understanding of fuel behavior under irradiation.

From an engineering perspective, PLEIADES development environment is a pretty vast ecosystem where many models, accelerators, predictors, convergence evaluators, post-processings, applications, frontends, databases, document generators, utilities are in active development or need to be maintained in operational conditions. This quickly led us to write comprehensive development guidelines and to automatize various tasks from code style checks and code coverage analysis to extensive testing on various operating systems and indeed, PLEIADES has now a long-standing tradition of using continuous integration tools.

Since 2021, PLEIADES has managed to migrate to a private instance of GitLab which provides a central point to host our code repositories (git), manage issues, publish informations and drive our continuous integration pipelines (using containerization - lightweight virtualization).

5. Numerical illustrations

5.1. Software applications

PLEIADES is dedicated to the numerical simulation of fuel elements during their various life phases: from manufacture, through irradiation, to storage. Each reactor type has its own dedicated software application. For each of these applications, common components are shared, laws and material behavior models (fuel and cladding), the entire set of numerical methods accessible within PLEIADES.

Nevertheless, the coupling tree is peculiar and related to physics, time and spatial scales of each setting. For example, let us focus on ALCYONE application dedicated to fuel modelling in pressurised water reactors (PWR). The coupling scheme for normal and transient (or off-normal – in red) operating conditions is shown in Fig. 16. To minimize execution times, the convergence acceleration algorithms were tuned specificaly for the targeted use cases. The ALCYONE application performs simulation of normal, off-normal and accidental situations with models dedicated to these conditions such as an axial gas circulation model and a modeling of heaters for LOCA (Lost of Coolant accident) experiments. Several hundreds of rods and experimental measures are used regularly for validating the code. A wide range of irradiation conditions is covered and the tool is reliable, robust and used in the scope of a fruitful collaboration between CEA, EDF and Framatome.

Another example concerns the GERMINAL application [4, 9] that simulates the in-pile behaviour of mixed oxide fuel pins for Sodium-cooled Fast Reactors (the acronym is SFR). GERMINAL was initially designed to simulate the fuel pin behaviour of the Phénix and Super-Phénix reactors, which were built in France. It integrates the modelling knowledge of SFR oxide fuel pin behaviour, acquired by the CEA over the years. The classical way of using GERMINAL employs multilayered axisymmetric 1D representation of the fuel pin geometry, see Fig. 2. The heat transport by the coolant makes the link between the radial layers, by fixing the boundary conditions at the axial positions of the layers. The SFR dedicated multiphysics couplings are based on specific models of fuel behavior under high temperature and high fast neutron flux conditions, see Fig. 17. Among these physical aspects, fuel restructuring and oxide-cladding joint (JOG for 'joint oxide-gaine') formation are important specific components. Let us mention that the validation database for GERMINAL gathers data from several thousands of characterized pins irradiated in SFR, including a wide range of fuel element features.

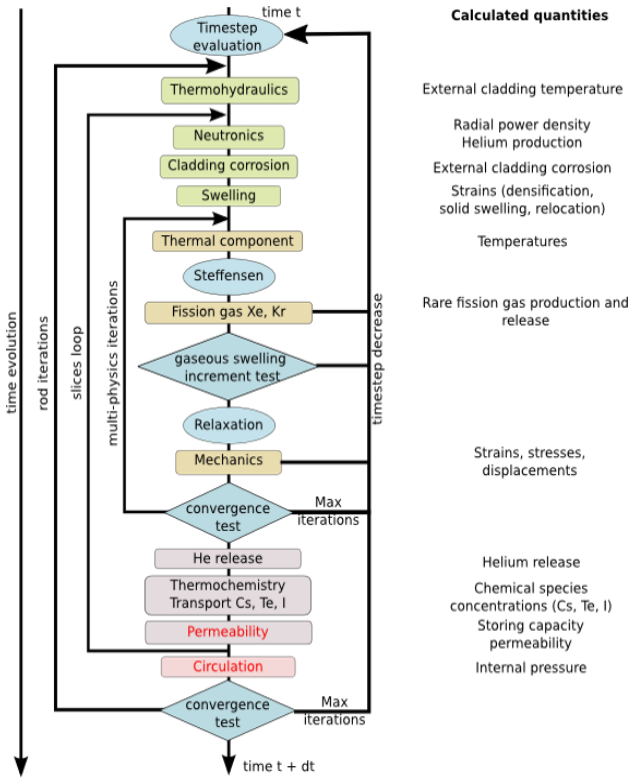


Figure 16: Illustration of a typical coupling tree for ALCYONE (V2 - second release) application, from [3]

Experimental results come mainly from irradiations achieved in France but also from other experimental facilities in the framework of international collaborations.

Another interesting application to be mentioned is the MIRABEL mini application (Mini App), which enables the on-the-fly construction of the coupling scheme. This Mini App is based on a 1D multilayered representation and proposes some simplified physical models derived from the rod/pin fuel performance codes (ALCYONE and GERMINAL). The main ones are namely burnup, gap heat transfer, thermal, mechanics, fission gas and internal pressure models. MIRABEL is mainly used to set up and test some advanced numerical methods in a fast-running multiphysics coupling configuration that is representative of fuel performance codes.

Also, from a software and test engineering perspective, having this simplified but physical application is a great benefit. It gives us a framework

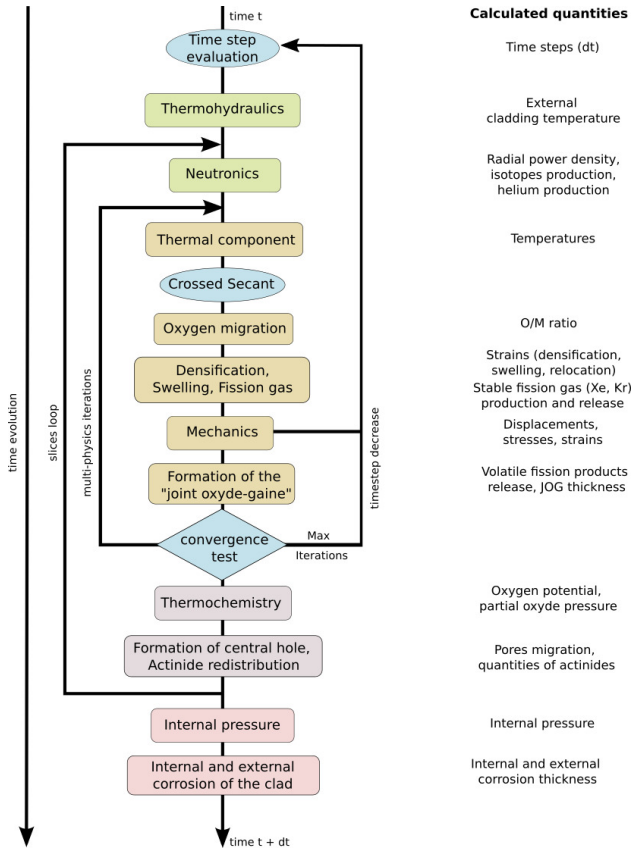


Figure 17: Illustration of a typical coupling tree for GERMINAL (V3 - third release) application

user's point of view on usability, on the impact of API changes (or any other change), on solver testing, on performance, on reproducibility, etc. and allows us to extend the framework's testing scope on a larger scale (integration testing).

5.2. Preprocessing tool

A preprocessing tool, named XPLEIADES, is helpful to feed in applications and is part of the PLEIADES ecosystem. It is developed in the Python language and simplify the generation of datafiles. With XPLEIADES, technological data and irradiation history can be filled in automatically from the experimental databases. Using its graphical or command-line interface, it allows the definition of coherent studies including fuel data, irradiation history,

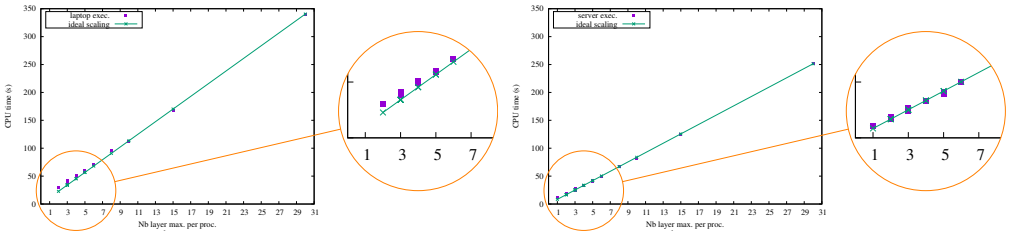
mesh, material laws and calculation options. The output of XPLEIADES is usually a file in XML format, which is transmitted to PLEIADES applications. Furthermore, the XSQL software is also used to make comparisons between calculations and measurements. This is part of the validation process required for quality assurance.

5.3. Examples of multilayered parallelization performances

The first example focuses on the multilayered parallelization of a typical thermo-physico-mechanical coupling performed through the MIRABEL Mini App for representative normal PWR irradiation conditions (5 cycles of irradiation). The typical 30-slice axial description is used. Figure 18 represents strong scaling tests performed on two different machines. First, a laptop targeted for numerical experiments equipped with a Intel Xeon E5-2630 v4 chip has been used. The clock speed is 2.20GHz, there are 2 sockets of 10 cores each (for a total amount of 20 cores) sharing 63 GB of memory. On this machine, hyperthreading is activated with 2 threads per physical core. Secondly, a linux server has been employed for running the MIRABEL typical test case. It is a performant computing machine that features the Intel Xeon Platinum 8268 processor. The base clock speed is 2.90GHz, and the compute node has a 376 GB RAM capacity. It is equipped with two sockets of 24 cores each (48 cores per node) and the hyperthreading is not activated.

For this strong scaling study, the number of MPI processes was increased in steps of 1, from 2 to 16 (i.e. 1 to 15 slave processes for the local layers calculation) for both laptop and server machine. The optimal total number of 31 MPI processes (cf. Section 3, Eq. (1)) was also performed on the server. However increasing the number of MPI processors by 1 does not necessarily reduce by 1 the maximum number of layers to be treated each process (cf. rest of the Euclidean division of 30 by the number of slave MPI processes). To have a fair scaling representation, each figure then plots the CPU time versus the maximum number of layers per MPI process. The ideal scaling curve is also represented.

The point at abcissa 30 corresponds to only one process managing the 30 layers, whereas the point at abcissa 2 means that 15 processes are responsible for 2 layers each. On the other hand, it may be that several calculations with different numbers of slave processes lead to the same maximum number of layers per processor: this is the case, for example, for abcissa 3, which corresponds to cases where 10, 11, 12, 13 or 14 processors handle the local calculations.



(a) On a linux laptop

(b) On a linux server

Figure 18: Strong scaling test on a typical 1D 30-layered calculation of a fuel rod irradiation - MIRABEL Mini App

As expected, there is a obvious difference in performance (CPU time) between laptop and server. However, for both machines, these examples show strong scaling profiles close to the ideal scaling curves. A speed-up of 12 (resp. 14) is obtained for 16 MPI processes on the laptop (resp. on the server), while the speed-up reaches 23 on the server for 31 MPI processes. Note a slight slope break in the laptop execution case (see Figure 18a), which is certainly be due to processor cache size.

A second performance test have been achieved for the 3D multilayered computational scheme of the ALCYONE application. The latter enables a direct computation at the fuel rod scale with a local description based on a 3D finite element model. An illustration of the results obtained during a power ramp transient is given in the Fig. 19 with the temperature distribution and the cladding ridges induced by pellet cladding mechanical interaction. In this figure, the temperature field is plotted for several pellets fragments per slice using a post-processing based on the symmetry assumptions of the solution obtained for a quarter of a pellet fragment. Let's notice than the PLEIADES framework can also handle asymmetries considering several pellet fragments of various angles in the azimuth, as for example shown in [43].

The study case used for performance and speed-up assessment represents a PWR fuel rod with 100 layers under base irradiation. The computations were performed on a cluster enabling the use of 128 cores with a distributed memory of 2 GB per core. The first scaling test was done with a local 3D mesh of 400 nodes (*i.e.* the global 3D mesh comprises 40,000 nodes) in order to have a reasonable computation time when testing a small number of cores. The observed execution times are plotted on the Fig. 20. The perfect scaling curve assumes that time is reduced by a factor 2 whenever the number

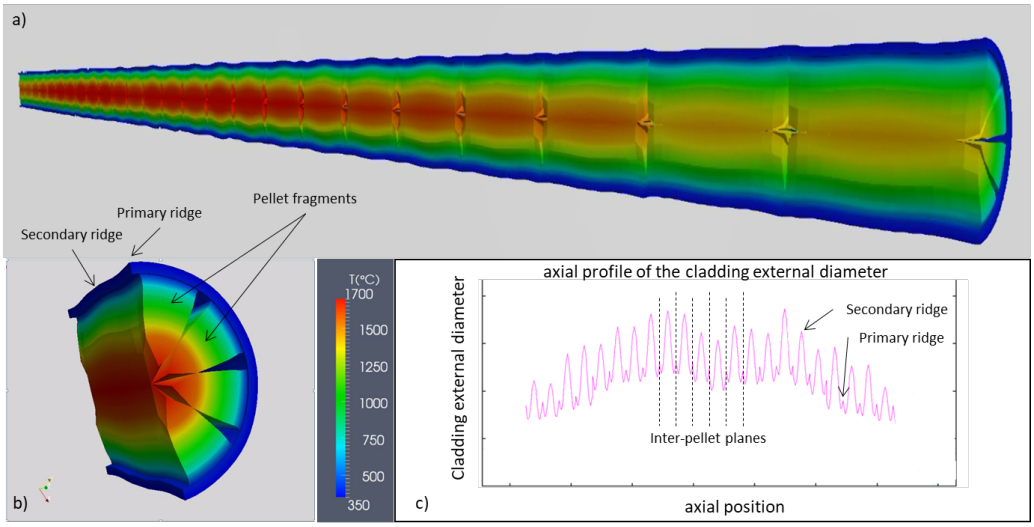


Figure 19: Simulation results obtained with the 3D multi layered computational scheme in the ALCYONE application. a) Temperature distribution in the refabricated fuel rod during the power ramp test b) detail of the temperature and deformation in one pellet c) Simulated axial profile of the cladding external diameter after the power ramp test

of cores is doubled. The strong scaling results show that the decrease of computation time follows the perfect speed-up curve with respectively 100 and 10 min for 5 and 50 cores.

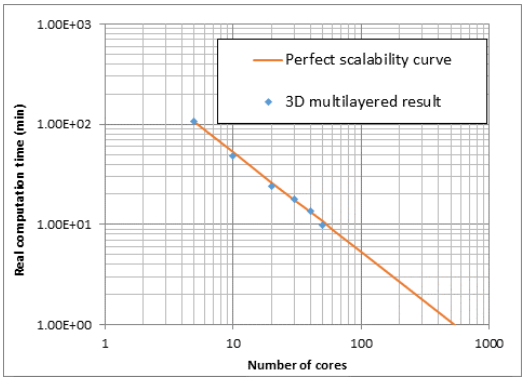


Figure 20: Strong scaling test of the 3D multilayered computational scheme with a local 3D mesh of 400 nodes and 100 layers

A weak scaling test was also performed with a higher mesh refinement of the local description (15,000 nodes). The speed-up assessment is based

Weak scaling test		
Number of slices	Number of cores	Real computation time
1	1	7h
100	101	16h

Table 1: Weak scaling test of the 3D multi layered computational scheme with a refined local 3D mesh (within each single slice) of 15,000 nodes

on two computations using respectively 1 and 100 slices (keeping a local 3D mesh within each slice of 15,000 nodes). The results, reported in the table 1, show that the real computation time is increased only from a ratio of two when comparing a 100 slices computation to a single slice computation while adapting the number of cores in order to have one core per slice (i.e. number of slice + 1 in case of multilayered parallel execution, see Eq. (1)). We consider this is a reasonable increase in term of computational overhead, even if the ideal would be a constant execution time between 1 and 101 cores.

All these examples confirm that the PLEIADES framework can be applied for different purposes, from laptops up to clusters.

5.4. Illustration of adaptive time marching

In this section, we illustrate the powerful and the robustness of the automatic time-stepping algorithm introduced in Section 2.3. The same typical 5-cycle PWR nominal irradiation as in Section 5.3 is simulated through the MIRABEL Mini App. A 30-layers axial description is used. The times of the irradiation data and the time nodes added automatically through the numerical time-stepping algorithm are shown in Figure 21 on a dimensionless linear power profile. Here the default splitting algorithm has been used: divide the time-step by 2 in case of non-convergence of the multiphysics loop after a maximal number of iterations set to 10. The prediction of the time step is based on the maximum value of the time increment of burnup, linear power and gaseous swelling. Multiphysics convergence is checked using fixed-point iteration discrepancies on temperature, pellet-cladding gap and gaseous swelling. If necessary, a Steffensen acceleration is performed on gaseous swelling.

One can notice that the time steps are well redivided in the intercycle phases (where the convergence is hard to obtain due to the strong change of regime) and tend to become uniform over power levels.

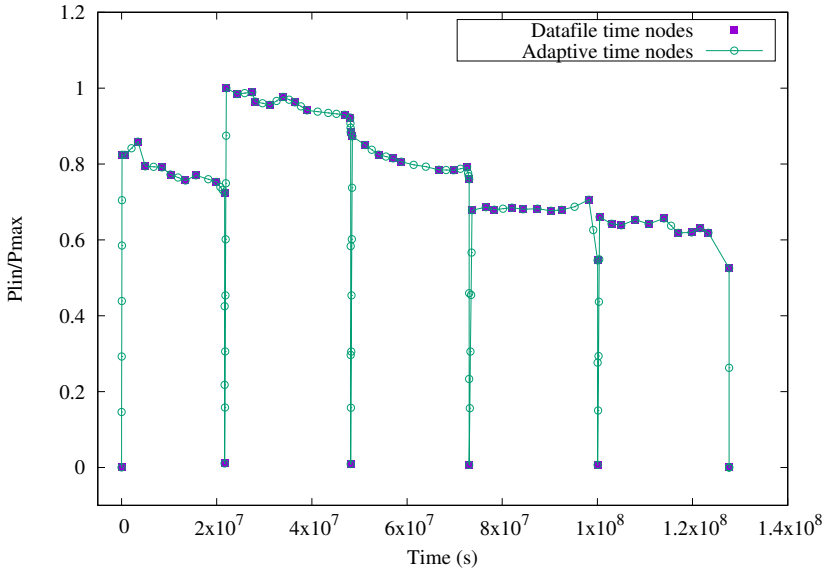
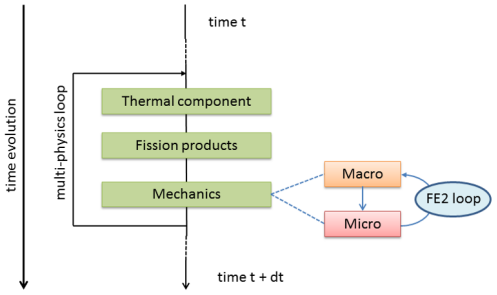


Figure 21: Example of the adaptive time-marching algorithm on a (normalized) Linear Power Profile - Layer #15 (over 30) - MIRABEL Mini App.

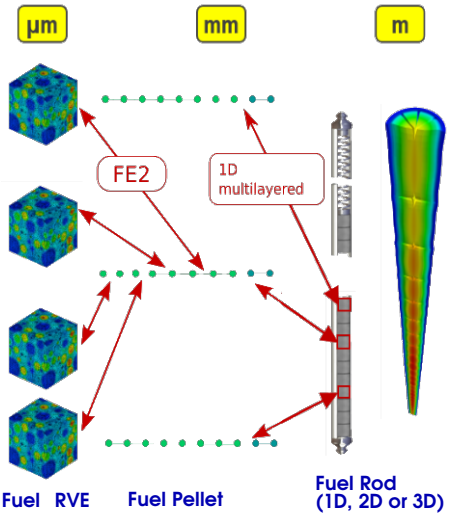
5.5. Scalable computational homogenization

In this section, we focus on the multiscale simulation of the mechanical behavior of heterogeneous fuels using the hybrid FE^2 algorithm introduced in Section 2.4. With a multiphysics PLEIADES application, the calculation scheme then features a double nested fixed-point loop, see Figure 22a. Through the multilayered description of the fuel rod, three scales are considered: the rod, the pellet and the microstructure scale, see Figure 22b. The following results come from FE^2 multiscale couplings, which are also multidimensional couplings, since they have been implemented it in a 1D multilayered representation of the fuel rod. The structure mesh (pellet scale) is 1D, while the microstructure mesh (RVE) is 3D, see Figure 22b. This results in a multiscale 1D/3D mechanical coupling iteratively managed, unlike Arlequin-type methods [44] where the composite system is solved. Finally, as detailed in Section 3.5, two nested level of parallelism are used to efficiently solve the resulting system.

Representative 3D microstructures of MOX (Mixed OXide) fuel were generated from 2D images of the real microstructure (microprobe images). This methodology and its validation (comparison of experimental and numerical covariances) are described in detail in [45]. In the following examples, we



(a) Multiscale scheme nested in a typical multi-physics coupling

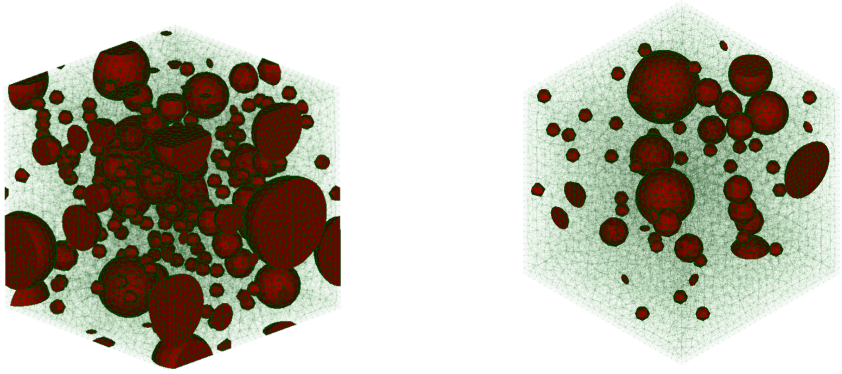


(b) PLEIADES multilayered and multiscale parallelization (3 scales considered)

Figure 22: Illustration of the intricated multiphysics, multiscale and multilayered couplings in PLEIADES

adopt a 2-phase description of the MOX microstructure, in which the inclusions representative of the plutoniferous clusters are preserved while the matrix is made up of the uraniferous clusters and the coating, the behavior of this second phase called matrix being homogeneous. This microstructure is ultimately a random microstructure of the matrix-inclusion type, with inclusions having a size distribution (polydisperse inclusions), see figure 23. We will focus here on results obtained with polydisperse inclusions, even if other types of distribution (monodisperse or periodic) have been studied, see [31].

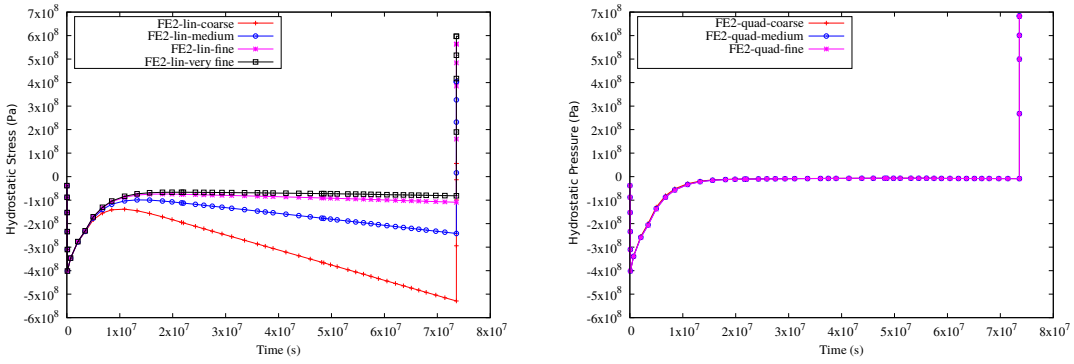
It is known that the inclusion phase is extremely sensitive to the mesh refinement, especially when the material behavior is nonlinear, much more so than the matrix phase or the apparent homogeneous behavior. This can be explained by the fact that the inclusionary phase has the smallest connected part and is therefore obviously the most sensitive to mesh variations.
A first advantage of the FE^2 integrated approach is to evaluate the influence of the microstructure mesh size and the order of the RVE FE approximation during the coupled multiscale and multiphysics calculation. For this verification, the MIRABEL Mini App was used. The 1D macroscopic mesh is fixed



(a) fv = 14% (b) fv = 6%

Figure 23: Examples of two-phase MOX RVEs used in the FE2 model with different volume fraction of inclusions (fv). Size $l = 150\mu m$. Matrix in light green, Inclusions in red.

at 48 elements. Four different RVE mesh refinements (from 2,000 to 100,000 elements) have been generated, and linear or quadratic tetrahedral finite elements have been tested. A local representative nonlinear heterogeneous mechanical behavior (imposed strains, creep, fission rate, etc.) is performed on the RVE. A simplified power history is simulated: power ramp-up, keeping power hot for around 3 years (constant power) and cold return. The use of quadratic finite elements makes the mean behavior of the inclusionary phase much less sensitive to mesh size, see Figure 24.



(a) First-order (linear) Finite Elements (b) Second-order (quadratic) Finite Elements

Figure 24: Mesh convergence of the mean hydrostatic stress of the inclusionary phase obtained with Periodic Boundary Conditions. MOX Microstructure with $fv = 14\%$.

Another advantage of using FE^2 is the ability to analyze local fields during a calculation. The following example has been carried out in the ALCYONE application on a two cycles irradiation history, coming from the validation data base. The burnup achieved was 24.6 GWd/tM. For this standard ALCYONE calculation, the fuel rod is divided into 9 layers. Here, the FE^2 calculation scheme was performed on slice number 4 (maximum neutron flux). The volume fraction of plutonium clusters is around 6%. A polydisperse RVE with 80,000 elements and quadratic finite elements is used, see Figure 23b. In Figure 25, the local distribution and frequency histogram of hydrostatic stress are represented at the end of the first irradiation cycle for a RVE located around the central Gauss point. The equivalent phase contrast, taking into account differences in imposed free strains, is around 3. In average, we

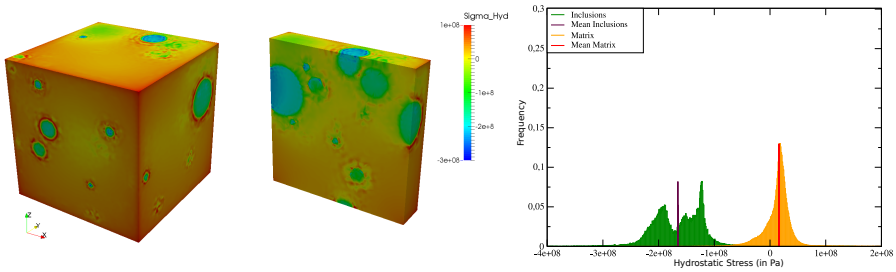


Figure 25: Hydrostatic stress fields in the RVE and frequency histograms. Left: full VER (view y=0%), middle: cross-section at y=80%, right: frequency histogram by phase and mean values.

can see that the matrix is in traction while the inclusions are in compression. That is due to a more important gaseous swelling in the inclusion phase. Furthermore this local view enables us to see a traction coating around inclusions. This zone, where the matrix is highly disturbed, can possibly be the site of local induced phenomena (cf. crack initiation).

We also carried out (quasi[2]) weak scaling tests on the cluster described in Section 5.3. For this end, we performed a multiphysics calculation with hybrid FE^2 mechanical model with 1, 4, 16, 32 or 64 FE^2 integration points

[2]We named it quasi weak scaling because the calculation cost is not strictly constant per process, as the number of processes increases. As the calculation time of FE^2 processes is much greater than that of processes assigned to macroscopic multilayered computation, only the number of FE^2 calculation and processes are considered in this weak scaling study

for a 1D-discretized pellet with 64 integration points. On integration points outside the FE^2 zone of interest, the homogenized Incremental Mori-Tanaka law, well suited to MOX mechanical behavior [28], is used. The 1-FE^2 Gauss point calculation is performed on the first Gauss point of the pellet (near the central point), while the other calculations use FE^2 Gauss points equally spaced along the pellet radius. Each FE^2 point is assigned to a different processor. The results are reported in Figure 26, where the measured "wallclock" times (i.e. the time elapsed between the start and end of the calculation) has been divided by the wallclock time obtained for 64 FE^2 points.

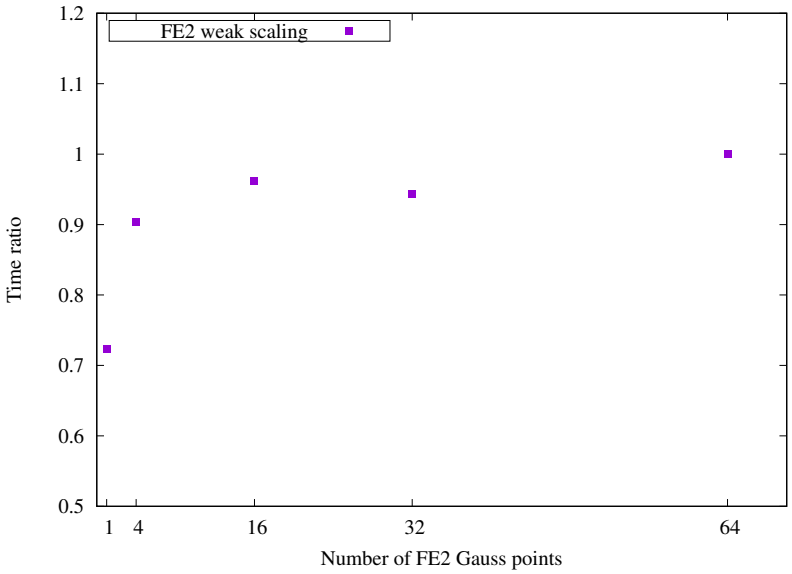


Figure 26: Weak scaling with respect to the number of FE^2 Gauss points. ALCYONE application.

Very good scaling of calculation times can be observed, particularly from 4 FE^2 integration points. The difference in execution time between the calculation with 1 or 4 FE^2 integration points is certainly due to the fact that the calculation load slightly differs depending on the integration point considered. In this case, it would appear that the central Gauss point (first FE^2 integration point considered) is not the most critical in terms of calculation time (less mechanical loads) compared to Gauss points located towards the outside of the pellet.

In addition, these results show that the time associated to the MPI data transfer is negligible compared to calculation time. These results confirm the

validity of the proposed hybrid FE^2 approach.

6. Conclusion

The PLEIADES framework provides a set of numerical tools to tackle irradiation of nuclear fuel modelling. It uses some open-source generic utilities such as SALOME or MFront, but proposes a unique feature to implement coupling trees associated to multiphysics partitioned fixed-point algorithm. In addition, multidimensional computational schemes with 1D up to 3D modelling and multiscale approaches are available. The parallel features of PLEIADES are very useful to shorten execution time of simulation tools.

The PLEIADES platform has been developed and used in close collaboration with the French nuclear industry over the past twenty years through several modeling codes such as ALCYONE (for PWR) and GERMINAL (for SFR). Using the modeling and numerical bricks connected through the PLEIADES framework, we aim to achieve an ever better understanding and representation of the mechanisms governing the behavior of fuel elements in the reactor core. The typical conditions under investigation are the behaviour in the reactor under normal conditions, off-normal and accidental conditions, as well as the backend of the fuel cycle (transport and waste storage configurations). Beyond the continuous integration natively included in the PLEIADES framework, the applications are continuously checked with non-regression and verification processes. Obviously, we also focus on validation against experimental data collected from various national and international irradiation programs with each new version of the modelling tools.

The current parallelized PLEIADES framework allows the use of a few hundred cores. PLEIADES-based applications exploit some assumptions to reduce the amount of computation and to ease parallelization, notably symmetry hypotheses and multilayered geometric description. This approach provides powerful tools in plenty of cases. However, it prevents the capture of some axial or azimuthal effects such as the dissymmetry of ballooning in LOCA (Lost Of Coolant Accident) and of HBS (High-Burnup Structure) restructuring zones, or the modelling of some heterogeneities in hydraulic flows. An evolution of the PLEIADES software platform has recently been undertaken. It focuses on the use of hundreds of thousands of computing cores and the removal of the assumptions mentioned above. The challenge is to meet a number of identified needs: 3D calculation of a complete rod, modelling of multidimensionnal heterogeneities, increasing the number of

unknowns for calculations on microstructures (RVE), or reducing calculation times for large computational domains. Highlights include support for multidimensional partitioning schemes (instead of multilayered decomposition) and upgrading the thermomechanical solver in order to ensure proper scaling on large supercomputers.

References

[1] R. L. Williamson, J. D. Hales, S. R. Novascone, G. Pastore, K. A. Gamble, B. W. Spencer, W. Jiang, S. A. Pitts, A. Casagranda, D. Schwen, A. X. Zabriskie, A. Toptan, R. Gardner, C. Matthews, W. Liu, H. Chen, BISON: A flexible code for advanced simulation of the performance of multiple nuclear fuel forms, Nuclear Technology 207 (2021) 954–980. doi:10.1080/00295450.2020.1836940.

[2] M. R. Tonks, D. Gaston, P. C. Millett, D. Andrs, P. Talbot, An object-oriented finite element framework for multiphysics phase field simulations, Computational Materials Science 51 (2012) 20–29. doi:https://doi.org/10.1016/j.commatsci.2011.07.028.

[3] V. Marelle, P. Goldbronn, S. Bernaud, E. Castelier, J. Julien, K. Nkonga, L. Noirot, I. Ramière, New developments in ALCYONE 2.0 fuel performance code, in: Top Fuel 2016 - Light Water Reactor (LWR) Fuel Performance Meeting, Boise, United States, 2016. URL: https://hal-cea.archives-ouvertes.fr/cea-02439467.

[4] M. Lainet, B. Michel, J.-C. Dumas, M. Pelletier, I. Ramière, GERMINAL, a fuel performance code of the PLEIADES platform to simulate the in-pile behaviour of mixed oxide fuel pins for sodium-cooled fast reactors, Journal of Nuclear Materials 516 (2019) 30–53. doi:https://doi.org/10.1016/j.jnucmat.2018.12.030.

[5] A. Soba, A. Denis, Dionisio 2.0: New version of the code for simulating a whole nuclear fuel rod under extended irradiation, Nuclear Engineering and Design 292 (2015) 213–221. doi:https://doi.org/10.1016/j.nucengdes.2015.06.008.

[6] A. Scolaro, I. Clifford, C. Fiorina, A. Pautz, The OFFBEAT multidimensional fuel behavior solver, Nuclear Engineering and Design 358 (2020) 110416. doi:https://doi.org/10.1016/j.nucengdes.2019.110416.

[7] D. Gaston, C. Newman, G. Hansen, D. Lebrun-Grandié, Moose: A parallel computational framework for coupled systems of nonlinear equations, Nuclear Engineering and Design 239 (2009) 1768–1778. doi:https://doi.org/10.1016/j.nucengdes.2009.05.021.

[8] C. J. Permann, D. R. Gaston, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner, R. C. Martineau, MOOSE: Enabling massively parallel multiphysics simulation, SoftwareX 11 (2020) 100430. doi:10.1016/j.softx.2020.100430.

[9] B. Michel, I. Ramière, I. Viallard, C. Introini, M. Lainet, N. Chauvin, V. Marelle, A. Boulore, T. Helfer, R. Masson, J. Sercombe, J. Dumas, L. Noirot, S. Bernaud, Chapter 9 - Two fuel performance codes of the PLEIADES platform: ALCYONE and GERMINAL, Woodhead Publishing Series in Energy, Woodhead Publishing, 2021, pp. 207–233. doi:https://doi.org/10.1016/B978-0-12-818190-4.00009-7.

[10] OpenCFD Ltd., Openfoam, `https://www.openfoam.com/about`, 2023.

[11] D. Knoll, D. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, Journal of Computational Physics 193 (2004) 357–397. doi:https://doi.org/10.1016/j.jcp.2003.08.010.

[12] J. Hales, S. Novascone, R. Williamson, D. Gaston, M. Tonks, Solving nonlinear solid mechanics problems with the jacobian-free newton krylov method, Computer Modeling in Engineering & Sciences 84 (2012) 123–154. doi:https://doi.org/10.3970/cmes.2012.084.123.

[13] CEA, Cast3m, `http://www-cast3m.cea.fr/`, 2023.

[14] CEA, Tfel, `https://thelfer.github.io/tfel/web/`, 2024.

[15] CEA, Mfront, `https://tfel.sourceforge.net/`, 2023.

[16] T. Helfer, B. Michel, J.-M. Proix, M. Salvo, J. Sercombe, M. Casella, Introducing the open-source mfront code generator: Application to mechanical behaviours and material knowledge management within the pleiades fuel element modelling platform, Computers & Mathematics with Applications 70 (2015) 994–1023. doi:10.1016/j.camwa.2015.06.027.

[17] F. Feyel, Multiscale FE^2 elastoviscoplastic analysis of composite structures, Computational Materials Science 16 (1999) 344 – 354. doi:10.1016/S0927-0256(99)00077-4.

[18] P. Thevenin, R. Masson, D. Baron, B. Petitprez, D. Plancq, CYRANO3: the industrial PLEIADES fuel performance code for EDF PWR studies, Proceedings of International Meeting nn LWR Fuel Performance, Topfuel2006, ENS, Salamanca, Spain (2006) 22–27.

[19] R. Largenton, K. Audic, F. Douchin, C. Pétry, CYRANO3 the EDF fuel code performance: Global overview and recent developments on MOX fuel, in: Top Fuel 2016: LWR Fuels with Enhanced Safety and Performance, 2016, p. 621 – 630.

[20] T. Barani, F. Bernachy-Barbe, N. Chauvin, C. Fillaux, M. Lainet, F. Marconi, I. Ramière, I. Viallard, Assessing the PLEIADES/GERMINAL V3 fuel performance code against transmutation experiments in sodium fast reactors, Annals of Nuclear Energy 203 (2024) 110526. URL: https://www.sciencedirect.com/science/article/pii/S0306454924001890. doi:https://doi.org/10.1016/j.anucene.2024.110526.

[21] D. Lorenzo, G. Marois, H. Palancher, , S. Valance, B. Ye, S. Shu, A. Yacout, Benchmark between the PLEIADES/MAIA and DART fuel performance codes on the e-future U-MO/Al dispersion fuel test, in: International Meeting on Reduced Enrichment for Research and Test Reactors, 2022.

[22] T. Helfer, S. Bejaoui, B. Michel, Licos, a fuel performance code for innovative fuel elements or experimental devices design, Nuclear Engineering and Design 294 (2015) 117–136. doi:10.1016/j.nucengdes.2015.07.070.

[23] Mérope, https://github.com/MarcJos/Merope, 2023. Accessed: 2023-12-19.

[24] M. Josien, Mérope : a microstructure generator for simulation of heterogeneous materials (2024). Hal-04486524.

[25] A. Lindsay, R. Stogner, D. Gaston, D. Schwen, C. Matthews, W. Jiang, L. K. Aagesen, R. Carlsen, F. Kong, A. Slaughter, C. Permann,

R. Martineau, Automatic differentiation in metaphysicl and its applications in moose, Nuclear Technology 207 (2021) 905–922. doi:10.1080/00295450.2020.1838877.

[26] I. Ramière, T. Helfer, Iterative residual-based vector methods to accelerate fixed point iterations, Computers and Mathematics with Applications 70 (2015) 2210 – 2226. doi:10.1016/j.camwa.2015.08.025.

[27] P. Goldbronn, J. Sercombe, B. Michel, Avancées de la simulation du comportement du combustible nucléaire en 3d et en transitoire rapide, in: Proceeding of the 21st French Congress of Mechanics, Bordeaux, 2013. URL: https://hal.science/hal-03441407/document.

[28] J. Ricaud, R. Masson, Effective properties of linear viscoelastic heterogeneous media : Internal variables formulation and extension to ageing behaviours, International Journal of Solids and Structure 46 (2009) 1599–1606.

[29] R. Masson, M. Seck, J. Fauque, M. Garajeu, A modified secant formulation to predict the overall behavior of elasto-viscoplastic particulate composites, Journal of the Mechanics and Physics of Solids 137 (2020) 103874. doi:10.1016/j.jmps.2020.103874.

[30] P. Ponte Castañeda, Stationary variational estimates for the effective response and field fluctuations in nonlinear composites, Journal of the Mechanics and Physics of Solids 96 (2016) 660 – 682. doi:https://doi.org/10.1016/j.jmps.2016.06.010.

[31] I. Ramière, R. Masson, B. Michel, S. Bernaud, Un schéma de calcul multi-échelles de type Éléments Finis au carré pour la simulation de combustibles nucléaires hétérogènes., in: 13e colloque national en calcul des structures, 126623, Université Paris-Saclay, May 2017, Giens, Var, France., Giens, Var, France, 2017. URL: https://csma2017.sciencesconf.org/126623.

[32] D. Koliesnikova, I. Ramière, F. Lebon, Analytical comparison of two multiscale coupling methods for nonlinear solid mechanics, Journal of Applied Mechanics 87 (2020) 094501.

[33] M. Praster, M. Klassen, S. Klinkel, An adaptive FE^2 approach for fiber–matrix composites, Computational Mechanics 63 (2019) 1333–1350.

[34] C. Gierden, J. Kochmann, J. Waimann, B. Svendsen, S. Reese, A review of FE-FFT-Based two-scale methods for computational modeling of microstructure evolution and macroscopic material behavior, Archives of Computational Methods in Engineering (2022). doi:10.1007/s11831-022-09735-6.

[35] H. Gu, J. Réthore, M.-C. Baietto, P. Sainsot, P. Lecomte-Grosbras, C. Venner, A. Lubrecht, An efficient multigrid solver for the 3D simulation of composite materials, Computational Materials Science 112 (2016) 230–237. doi:https://doi.org/10.1016/j.commatsci.2015.10.025.

[36] R. Largenton, J.-C. Michel, P. Suquet, Extension of the nonuniform transformation field analysis to linear viscoelastic composites in the presence of aging and swelling, Mechanics of Materials 73 (2014) 76–100.

[37] F. Fritzen, M. Hodapp, The finite element square reduced (FE^{2R}) method with GPU acceleration: towards three-dimensional two-scale simulations, International Journal for Numerical Methods in Engineering 107 (2016) 853–881. doi:10.1002/nme.5188.

[38] CEA, EDF, Salome, the open source platform for numerical simulation, `https://www.salome-platform.org`, 2023. [Online; accessed 30-May-2023].

[39] CEA, Interface for code coupling - icoco api, `https://github.com/cea-trust-platform/icoco-coupling`, 2023. [Online; accessed 30-May-2023].

[40] A. Hindmarsh, Odepack, a systematized collection of ode solvers, `https://people.sc.fsu.edu/ jburkardt/f77_src/odepack/odepack.html`, 1983. [Online; accessed 30-May-2023].

[41] The HDF Group, Hdf5, high-performance data management and storage suite, `https://www.hdfgroup.org/solutions/hdf5/`, 2023. [Online; accessed 30-May-2023].

[42] Autorité de Sûreté Nucléaire, Institut de Radioprotection et de Sûreté Nucléaire, Qualification des outils de calcul scientifiques utilisés dans la démonstration de sûreté nucléaire - guide 28, https://www.irsn.fr/sites/default/files/documents/expertise/demarches-de-suret 2017. [Online; accessed 30-May-2023].

[43] J. Sercombe, T. Helfer, E. Federici, D. Leboulch, T. Le Jolu, A. H. de Ménibus, C. Bernaudat, 2D simulation of hydride blister cracking during a RIA transient with the fuel code ALCYONE, EPJ N-Nuclear Sciences & Technologies 2 (2016) 22.

[44] H. Ben Dhia, G. Rateau, The arlequin method as a flexible engineering design tool, International Journal for Numerical Methods in Engineering 62 (2005) 1442–1462. doi:https://doi.org/10.1002/nme.1229.

[45] R. Largenton, Modélisation du comportement effectif du combustible MOX en service: analyse micromécanique en champs de transformation non uniformes, Ph.D. thesis, Université de Provence, 2012. In French.

Acknowledgments

This work has been mainly achieved in the framework of the French Nuclear Institute (Institut Tripartite CEA-EDF-Framatome).