**1. INTRODUCTION**

The continuous improvement in computers speed and storage capacity have had a deep impact in the way engineers and scientists work on their problems. The nuclear engineering field have been benefiting from computers continuous increasing in processing power, turning former expensive calculation methods into useful and practical tools to solve many different problems in the field. A recent focus is the use of thermal-hydraulics and neutronics codes to solve nuclear reactor problems in a coupled way. These coupled calculations approaches, also called multi-physics (cite Leppannen), offers an interesting way of modeling the feedback from thermal-hydraulics to neutronics and vice versa.

Another interesting aspect of the computer hardware evolution, is the changes in the associated software. The effect of software evolution in order to follow the respective hardware improvement is often neglected, specially in the nuclear engineering field. Despite the healthy discussion on the safety and the advantages and disadvantages of open software in many fields (cite Androutsellis-Theotokis), it's impossible to deny open software influence in the society nowdays. However, more than software itself, open software brings a new culture of software development and, as one would expect, this cultural change reached scientists and engineers in the nuclear field. It goes from Monte Carlo criticality calculations like proposed by the OpenMC project (citar Romano Forget 2013), passing through deterministic approaches for neutron calculations (citar Boyd 2014), to full nuclear cycle calculations (citar Huff\_etal\_2016). All of these softwares share a common aspect: they are all open, freely available and, most important, their development is decentralized and anyone interested in participating can download it, make changes and improvements and submit the contribution to the community. This development cycle is, probably, the main strength of an open software project.

This work discusses some efforts initiated as a cooperation development between the Nuclear Engineering Department, Federal University of Minas Gerais and Reactors Technology Service, Nuclear Technology Development Center of Brazilian National Nuclear Commission. In the light of the open software development, the present work aims to introduce a coupled neutronics and thermal-hydraulics framework for nuclear reactors calculations. The thermal-hydraulics calculations are performed by an open software Computational Fluid Dynamics (CFD) toolbox called *OpenFOAM* (citar OpenFOAM). For neutronics calculations, a free nuclear reactor core analysis code called *milonga* solves the steady-state multigroup neutron transport equation (citar milonga). In this coupled framework, both codes use the same mesh for domain discretization, allowing both codes to solve their problems with the same degree of detail. Data is shared between them through a shared memory scheme, which poses no overhead for communication larger than any ordinarymemory access. The cross-sections processing is performed using *Serpent* continuous-energy Monte Carlo reactor physics code (citar Serpent). From continuous cross-section libraries a set of two-groups cross-sections is generated for three different materials at different temperatures, corresponding to the modeled fuel element temperatures variation.

In order to validate the coupled framework, a geometry representing a quarter of a TRIGA reactor fuel element is modeled and meshed. It is worth noting that validation in the context means to guarantee that data is properly exchanged from thermal-hydraulics to neutronics and the vice versa, calculations are correctly carried and numerical convergence is achieved in both codes. In other words, the validation of the methodology and the correctness of the implementation are presented for this framework of three-dimensional multi-physics calculations.

In Section 2 the models, codes used and methodology for cross-sections generation, are briefly described. In Section 3, the coupled aspects are discussed involving details and implementation decisions. Section 4 is dedicated to describe the application and results from separated and coupled calculations. Finally, in Section 5, the conclusions and results are presented along with the discussion of future development paths for the framework.

**2. MODELS AND METHODOLOGY**

Before diving into the details of the coupled calculations, it is important to describe the models and methodology used for each code in stand-alone mode. It is also fundamental to remark that both codes solve their equations in a steady-state mode.

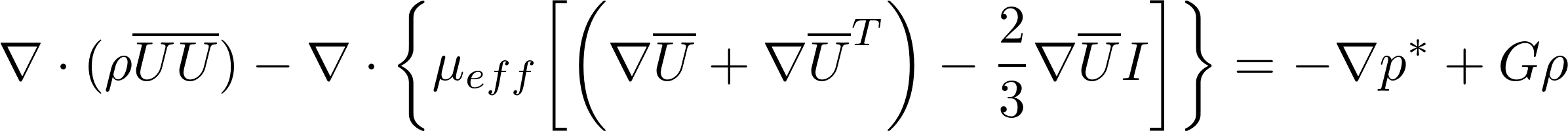
2.1 Thermal-hydraulics

The *OpenFOAM* toolbox offers numerical solvers for many common problems, like heat transfer, two phase fluid flow, financial problems and many others. In order to solve a one phase fluid flow including conjugated heat transfer from three different materials, the so-called chtMultiRegionSimpleFoam was the choice.

The fluid dynamics is governed by the momentum equation, continuity equation and energy equation (i.e. Navier-Stokes equation – CITAR). The chosen solver separates fluid regions from solid regions and then solves the appropriated equations for each type of region, taking into account the thermal and physical properties of each material, defined for every region.

For the fluid regions, and based in previous values of pressure and density, it solves respectively the equations related to momentum, energy and pressure. Differently from common numerical tools, the solver used, as all other solvers available in OpenFOAM, allows equations assembly directly using objects including discretization methods. That said, the respective equations solved for the momentum are:

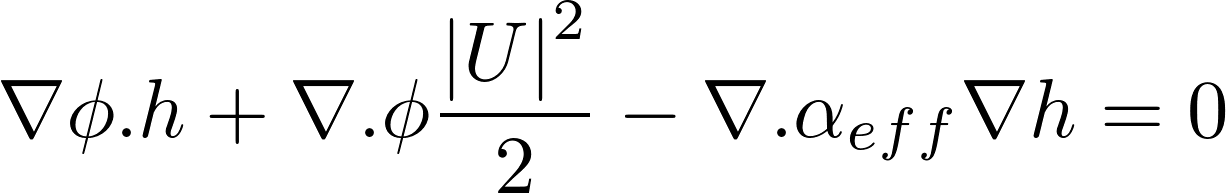
(1)



where the *μeff* term is the effective dynamic viscosity, which is the sum of viscosity and turbulent viscosity. The *Gρ* term comes from a buoyancy adjustment.

The energy equation is implemented in a flexible way, allowing for calculations using enthalpy or internal energy. In this work enthalpy is used and the energy equation is described accordingly:

(2)



where the second therm is the contribution from kinetic energy and the third therm accounts for the heat flux, being *αeff* is the sum of laminar and turbulent thermal diffusivities. It is worth noting that the velocities field is previously calculated as the equations are implemented in a specific order. The next equation to be described is the so-called continuity equation.

(3)



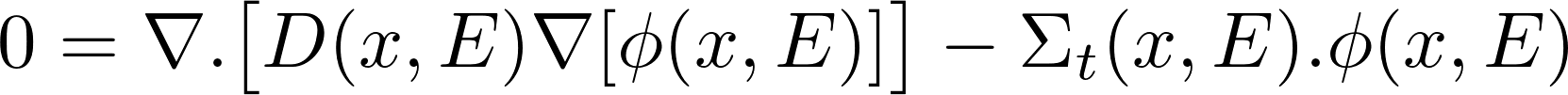
The transient term is not present for the steady-state problem and the continuity is assured when the divergent of momentum is zero.

OpenFOAM has its own particular way of coupling pressure to velocity (citar Karrolhm) which is beyond the scope of this text.

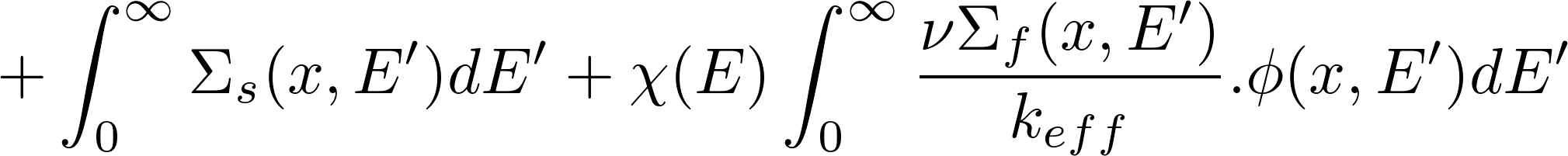
FIGURA do ALGORITMO

2.2 Neutronics

The *milonga* nuclear code is an open software released under GNU license. It heavily relies on other well known GNU libraries, like PETSc (citar), SLEPc (citar) and GNU Scientific Library (citar), bringing to it the robustness of these established software and keeping the principles of freedom of software use that GNU enforces. It solves the steady-state multi-group neutron transport equation, either using the diffusion approximation or discrete ordinates *Sn* method. Its ability to use finite-volumes discretization schemes over unstructured grids makes it possible, regarding the present problem, to have the same mesh for both thermal-hydraulics and neutronics. With identical meshes, the dependency between fields in each tool can be solved with the same degree of detail. Despite the inaccurate results and limitations of the diffusion approximation under some circumstances (citar Travis\_Tranjan\_2014), it is the method used in this work due to its faster execution time compared to the *Sn* method. Moreover, with a proof-of-concept methodology in mind, any changes in power and flux calculations can be observed using the diffusion approximation for steady-state:

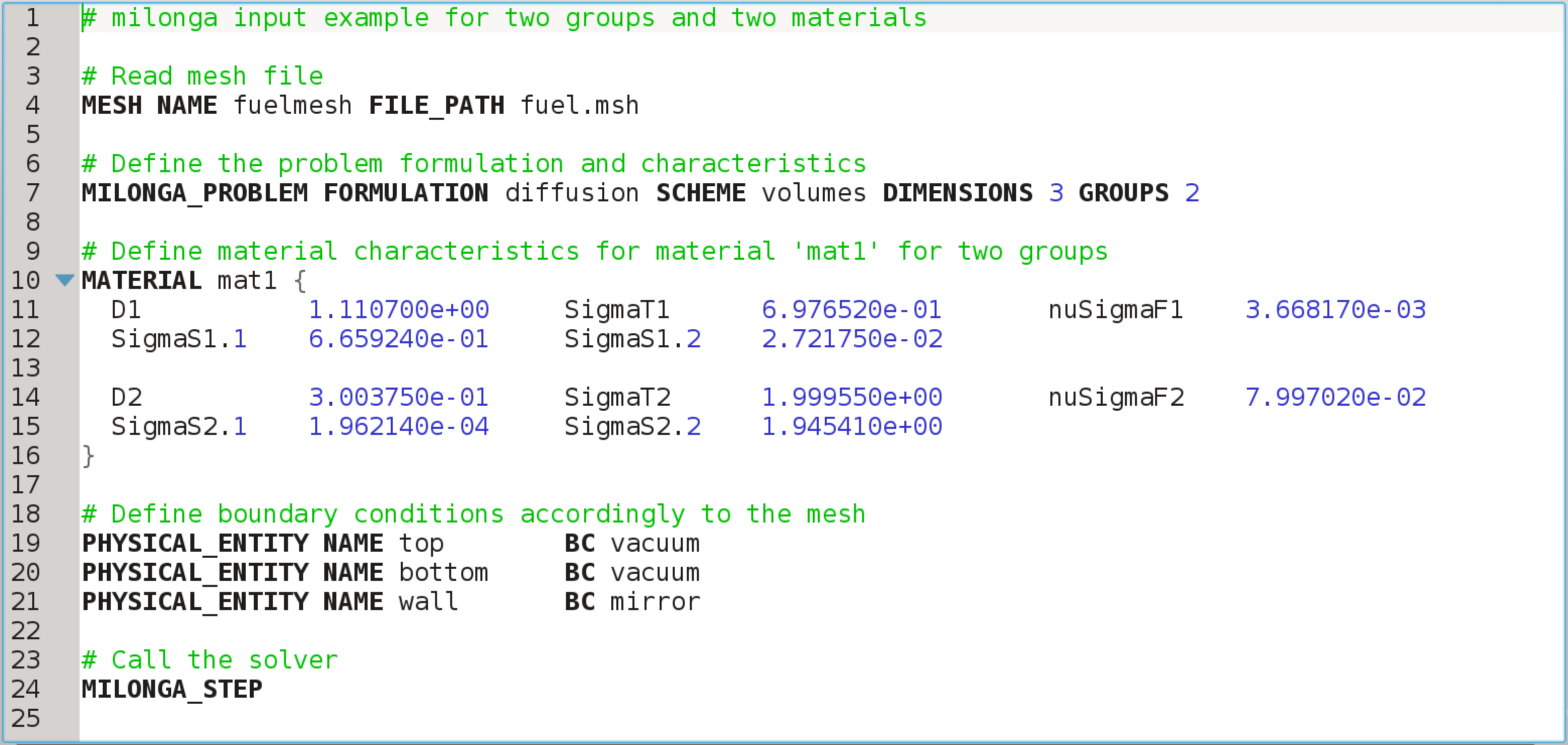


(4)



Equation 4 is discretized using the chosen formulation, finite volumes in the present case. Despite the cross-sections dependence on temperatures, the diffusion equation can be formulated as linear since the coefficients are kept constant during each step in the iterative calculation.

In a stand-alone case, *milonga* works reading an input file which models the problem to be solved. The simplest case for the present problem consists in the basic problem configuration: read the mesh, define the numerical formulation and chose the number of neutron groups for the problem. The next step is to define the coefficients for each material matching the materials defined in the mesh. These coefficients are used in the solution of the diffusion equation and must match the material characteristics, like the diffusion coefficients, absorption macroscopic cross sections, scattering cross-sections. All of them defined for each group. After materials definitions, the appropriate boundary conditions are set matching the physical entities defined in the mesh. In order to solve the problem, a simply call to a command builds and solve the eigenvalue problem. Figure 2 shows a simple input file for milonga.

**Figure 2: A simple input file for milonga with line number counting.**

It is important to mention that *milonga* has a wide range of commands and primitives, ranging from meshing pre-processing to output in default formats for data visualization software.

A key feature of milonga which makes external coupling straightforward is the option to get data in a cell wise way when discretizing the calculation domain in a mesh. In other words, is possible to have a spacial function dependence for any variable. However, to have the corresponding values of cross-sections (and all other coefficients for the diffusion approximation) for each cell, a properly set of these neutronic coefficients had to be generated in advance.

2.2.1 Neutronics methodology

Aknowledges

CAPES