Hindawi Publishing Corporation
Science and Technology of Nuclear Installations
Volume 2013, Article ID 641863, 26 pages
http://dx.doi.org/10.1155/2013/641863

Research Article

Unstructured Grids and the Multigroup Neutron Diffusion Equation

German Theler

TECNA Estudios y Proyectos de Ingeniería S.A., Encarnación Ezcurra 365, C1107CLA Buenos Aires, Argentina

Correspondence should be addressed to German Theler; gtheler@tecna.com

Received 22 May 2013; Revised 20 July 2013; Accepted 20 July 2013

Academic Editor: Arkady Serikov

The neutron diffusion equation is often used to perform core-level neutronic calculations. It consists of a set of second-order partial differential equations over the spatial coordinates that are, both in the academia and in the industry, usually solved by discretizing the neutron leakage term using a structured grid. This work introduces the alternatives that unstructured grids can provide to aid the engineers to solve the neutron diffusion problem and gives a brief overview of the variety of possibilities they offer. It is by understanding the basic mathematics that lie beneath the equations that model real physical systems; better technical decisions can be made. It is in this spirit that this paper is written, giving a first introduction to the basic concepts which can be incorporated into core-level neutron flux computations. A simple two-dimensional homogeneous circular reactor is solved using a coarse unstructured grid in order to illustrate some basic differences between the finite volumes and the finite elements method. Also, the classic 2D IAEA PWR benchmark problem is solved for eighty combinations of symmetries, meshing algorithms, basic geometric entities, discretization schemes, and characteristic grid lengths, giving even more insight into the peculiarities that arise when solving the neutron diffusion equation using unstructured grids.

1. Introduction

The better we engineers are able to solve the equations that model the real physical plants we design and build, the better services we can provide to our customers, and thus, general people can be benefited with better nuclear facilities and installations. The Boltzmann neutron transport equation describes how neutrons move and interact with matter. It involves continuous energy and space-dependent macroscopic cross-sections that should be known beforehand and gives an integrodifferential equation for the vectorial flux as a function of seven independent scalar variables, namely, three spatial coordinates, two angular directions, energy, and time. It represents a balance that holds at every point in space and at every instant in time. Such an equation may be tackled using a variety of approaches; one of them is a simplification that leads to the so-called neutron diffusion approximation that states that the neutron current is proportional to the gradient of the neutron flux by means of a diffusion coefficient, which is a function of the macroscopic transport cross-section. When this approximation—which is analogous to Fick's law in species diffusion and to the Fourier expression of the heat flux—is replaced into the transport equation, a partial differential equation of second order on the spatial coordinates is obtained. Formally, the neutron diffusion equation may be derived from the transport equation by expanding the angular dependance of the vectorial neutron flux in a spherical harmonics series and retaining both the zero and one-moment terms, neglecting the contributions of higher moments [1, 2]. As crude as it may seem, this diffusion equation gives fairly accurate results when applied under the conditions in which thermal nuclear reactors usually operate. Indeed, it can be shown that in a homogeneous bare critical reactor, the neutron current is proportional to the neutron gradient for every neutron energy [3].

The energy domain is usually divided into a finite number of groups, thus transforming one partial differential equation over space and energy into several coupled equations—one

for each group—containing differential operators applied only over the spatial coordinates. This resulting set of second-order PDEs is known as the multigroup neutron diffusion equation and is usually used to model, design, and analyze nuclear reactor cores by the so-called core-level calculation codes. These programs take homogenized macroscopic cross-sections (which may depend on the spatial coordinates through changes of fuel burnup, materials temperature or other properties) computed by lattice-level codes as an input and solve the diffusion equation to obtain the flux (and its related quantities such as power, xenon, etc.) distribution within the core.

Given a certain spatial distribution of materials and its properties inside a reactor core, chances are that the resulting reactor will not be critical. That is to say, in general, the rate of absorptions and leakages will not exactly overcome the neutrons born by fissions sources, and some kind of feedback—either through an external control system or by means of an inherent stability mechanism of the core [4]—is needed to keep the reactor power within a certain interval. Mathematically, this means that the transport—and thus the diffusion—equation does not have a steady-state solution. In practice, given a certain reactor configuration, its steady-state flux distribution is computed by setting all the time derivatives to zero as usual but also by dividing the fission sources by a positive value k_{eff} called the effective multiplication factor, which becomes also one unknown and turns the formulation into a eigenvalue problem. The largest (or smallest, depending on the formulation) eigenvalue is therefore the effective multiplication factor. If $k_{\text{eff}} < 1$, the original configuration was subcritical, and conversely. As successive configurations make $k_{\text{eff}} \to 1$, the associated eigenfunctions approach the steady-state critical flux distribution [5].

Core-level codes traditionally use regular grids to discretize the differential operators over the space. Depending on the characteristics and symmetry of the reactor core, either squares or hexagons are used as the basic shape of the mesh. Usual discretization schemes involve cell-centered finite differences or two-step coarse-mesh/coupling coefficient methods [6–8], which are accurate and efficient enough for most applications. There are, nevertheless, certain limitations that are not inherently related to structured grids but to the way their coarseness is utilized and how they are computed and applied to the reactor geometry. These issues can be overcome by discretizing the spatial operators using a scheme which could be applied to nonstructured grids, namely, finite volumes or finite elements.

This way, unstructured grids may be used to study, analyze, and understand the numerical errors introduced by the discretization of the leakage operator with a difference-based scheme over a coarse structured grid by successively refining the mesh whilst comparing the solutions with the structured one, as depicted in Figure 1. Another example of application may be the improvement of the response matrix of boundary conditions over cylindrical surfaces, which is the case for most of the nuclear power plants cores. When a coarse structured mesh is applied to a curved geometry, there appears a geometric condition known as the staircase effect. Even though arbitrary shapes cannot be perfectly tessellated with unstructured grids, for the same number of unknowns, they better represent the geometry than structured meshes, as Figure 2 illustrates. Again, by refining the mesh and comparing solutions, the matrix responses used to set the boundary conditions on the coarse meshes can be optimized. Finally, other complex geometries such as slanted control rods (Figure 3) or irradiation chambers can be directly taken into account by unstructured grids, possibly by refining the mesh in the locations around said complexities.

2. The Steady-State Multigroup Neutron Diffusion Equation

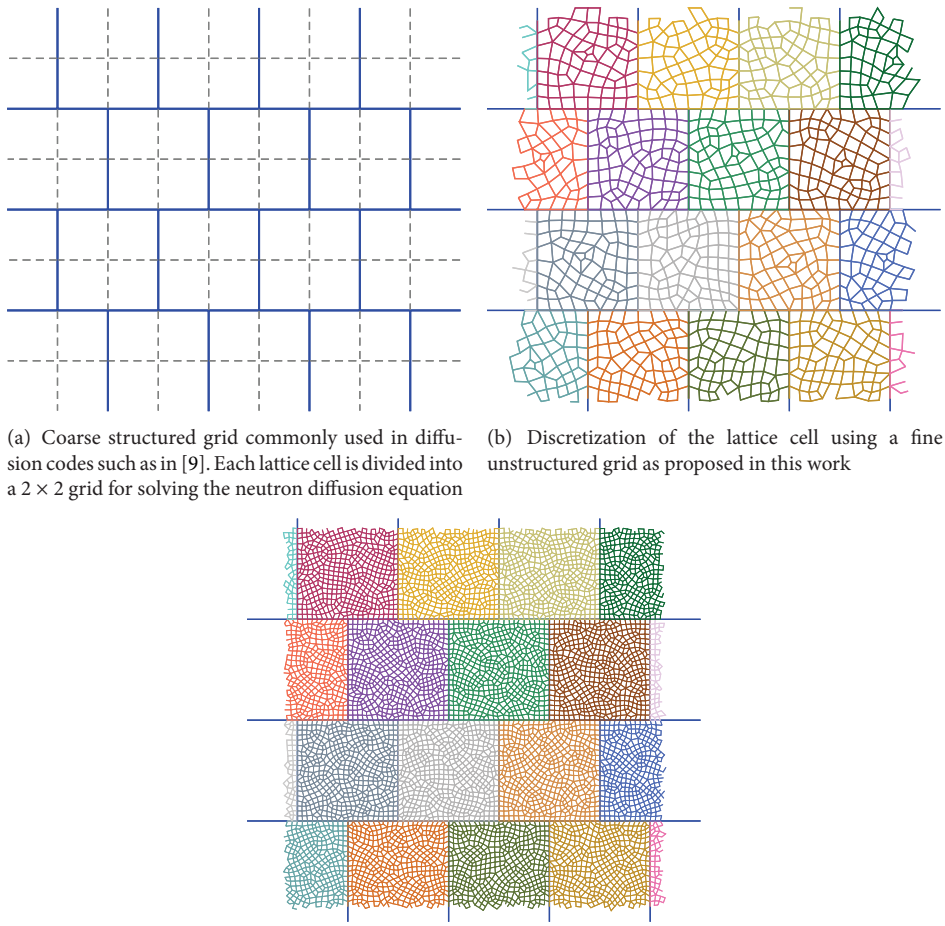
In the present work, we take the steady-state multigroup neutron diffusion equation for granted. That is to say, we focus on the mathematical aspects of the eigenvalue problem and make no further reference to its derivation from the transport equation nor to the validity of its application to reactor problems, as these subjects that are extensively discussed in the classic literature [1, 5, 11].

The differential formulation of the steady-state multigroup neutron diffusion equation over an m-dimensional domain U with G groups of energy is the set of G coupled differential equations:

$$\begin{aligned} &\operatorname{div}\left[D_g(\mathbf{x}) \cdot \operatorname{grad} \phi_g(\mathbf{x})\right] - \Sigma_{ag}(\mathbf{x}) \cdot \phi_g(\mathbf{x}) \\ &\quad + \sum_{g' \neq g}^{G} \Sigma_{sg' \to g}(\mathbf{x}) \cdot \phi_{g'}(\mathbf{x}) \\ &\quad + \chi_g \cdot \sum_{g'=1}^{G} \frac{\nu \Sigma_{fg'}(\mathbf{x})}{k_{\text{eff}}} \cdot \phi_{g'}(\mathbf{x}) = 0, \end{aligned} \tag{1}$$

where ϕ_g are the $g = 1, \ldots, G$ unknown flux distributions and the Σs and the Ds are the macroscopic cross-sections and diffusion coefficients, which ought to be computed by a lattice-level code and taken as an input to core-level codes. However, for the purposes of fulfilling the objectives of this work, we take the macroscopic cross-sections as functions of the spatial coordinate $\mathbf{x} \in \mathbb{R}^m$ which is known beforehand. The coefficient χ_g represents the fission spectrum and is the fraction of the fission neutrons that are born into group g. As stated above, the ν-fissions are divided by a positive effective multiplication factor k_{eff} and all the time derivatives, that is, the right hand of (1) is set to zero. We note that, in order to define a consistent nomenclature, we use the absorption cross section Σ_{ag} of the group g, which is equal to the total cross section Σ_{tg} minus the self-scattering cross section $\Sigma_{sg \to g}$. Therefore, the sum over the scattering sources excludes the term corresponding to $g' = g$. If we had used the total cross section in the second term of (1), we would have had to include the self-scattering term as a source.

If the cross sections depend only in an explicit way on the spatial coordinate $\mathbf{x}$, then (1) is linear. If, as is the general case, the cross sections depend on $\mathbf{x}$ through the flux $\phi_g(\mathbf{x})$ itself—such as by means of the xenon concentration or by local temperature distributions—then (1) is nonlinear. Nevertheless, this general case can be solved by successive



(a) Coarse structured grid commonly used in diffusion codes such as in [9]. Each lattice cell is divided into a 2×2 grid for solving the neutron diffusion equation

(b) Discretization of the lattice cell using a fine unstructured grid as proposed in this work

(c) Further refinement of the unstructured grid

Figure 1: Discretization of a homogenized PHWR array of fuel channels for a core-level diffusion computation. Each square is a lattice-level cell comprising one fuel channel and the surrounding moderator.

linear iterations so the basic problem can be regarded as being purely linear.

It should be noted that, if at least one of the diffusion coefficients $D_g(\mathbf{x})$ is discontinuous over space, then the divergence operator is not defined at the discontinuity points. Therefore, the differential formulation—also known as the strong formulation—is not complete when there exist material discontinuities that involve the diffusion coefficients. At these material interfaces, the differential equation has to be replaced by a neutron current conservation condition:

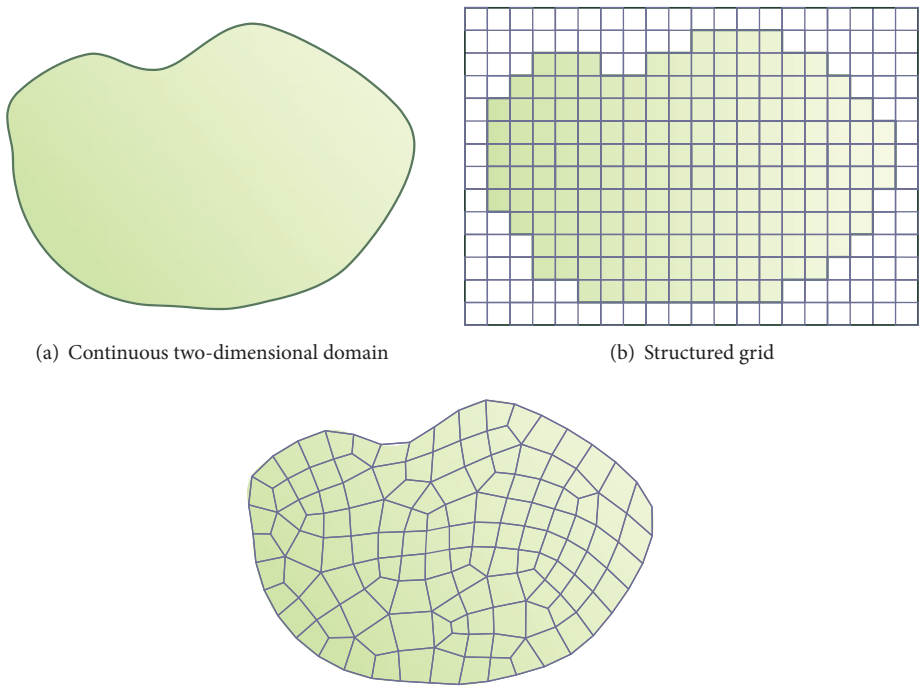
$$D_g^+(\mathbf{x}) \cdot \text{grad}\ \phi_g^+(\mathbf{x}) = D_g^-(\mathbf{x}) \cdot \text{grad}\ \phi_g^-(\mathbf{x}), \tag{2}$$

where the plus and minus sign denote both sides of the interface. As the diffusion coefficients are different, the resulting flux distribution $\phi_g(\mathbf{x})$ ought to have a discontinuous gradient at the interface in order to conserve the current.

When transforming the strong formulation into a weak formulation—not just into an integral formulation—both (1) and (2) can be taken into account by a single expression. In effect, let $\varphi_g(\mathbf{x})$ be arbitrary functions of $\mathbf{x}$ for $g = 1, \ldots, G$. Multiplying each of the G equations (1) by $\phi_g(\mathbf{x})$, integrating over the domain $U \in \mathbb{R}^m$, and applying Green's formula [12] to the leakage term, we obtain

$$\begin{aligned}
&\int_U D_g(\mathbf{x}) \cdot \left[\text{grad}\ \varphi_g(\mathbf{x}) \cdot \text{grad}\ \phi_g(\mathbf{x})\right] d^m\mathbf{x} \\
&\quad + \int_{\partial U} \varphi_g(\mathbf{x}) \cdot D_g(\mathbf{x}) \cdot \left[\text{grad}\ \phi_g(\mathbf{x}) \cdot \hat{\mathbf{n}}\right] dS \\
&\quad + \int_U \varphi_g(\mathbf{x}) \cdot \Sigma_{ag}(\mathbf{x}) \cdot \phi_g(\mathbf{x})\, d^m\mathbf{x} \\
&\quad + \int_U \varphi_g(\mathbf{x}) \cdot \sum_{g' \neq g}^{G} \Sigma_{sg' \to g}(\mathbf{x}) \cdot \phi_{g'}(\mathbf{x})\, d^m\mathbf{x} \\
&\quad + \chi_g \cdot \int_U \varphi_g(\mathbf{x}) \cdot \sum_{g'=1}^{G} \frac{\nu\Sigma_{fg'}(\mathbf{x})}{k_{\text{eff}}} \cdot \phi_{g'}(\mathbf{x})\, d^m\mathbf{x}.
\end{aligned} \tag{3}$$

These G coupled equations should hold for any arbitrary set of functions $\varphi_g(\mathbf{x})$. Making an analogy between (3) and the principle of virtual work for structural problems, we call these functions virtual fluxes. This formulation does not involve any differential operator over the diffusion coefficient and yet,



(a) Continuous two-dimensional domain

(b) Structured grid

(c) Unstructured grid

Figure 2: When a continuous domain (a) is meshed with an unstructured grid, there appears a geometric condition known as the staircase effect (b). For the same number of nodes, unstructured grids reproduce the original geometry better (c).

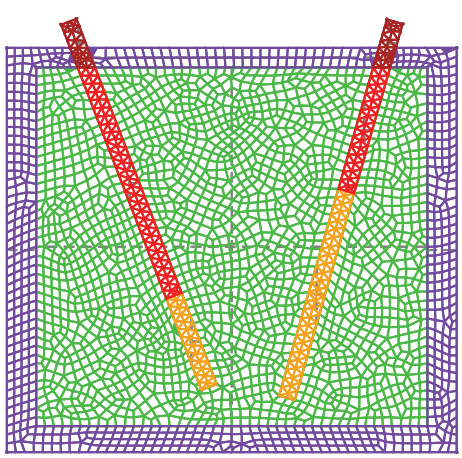


Figure 3: Cross section of a hypothetical reactor in which the control rods enter into the core from above with a certain attack angle with respect to the vertical direction.

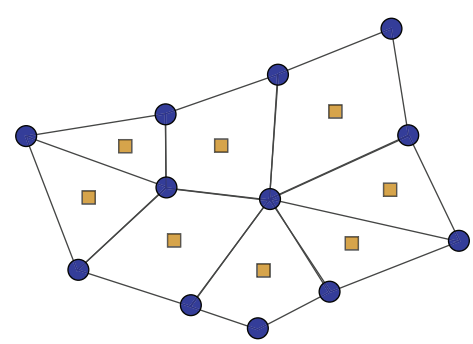
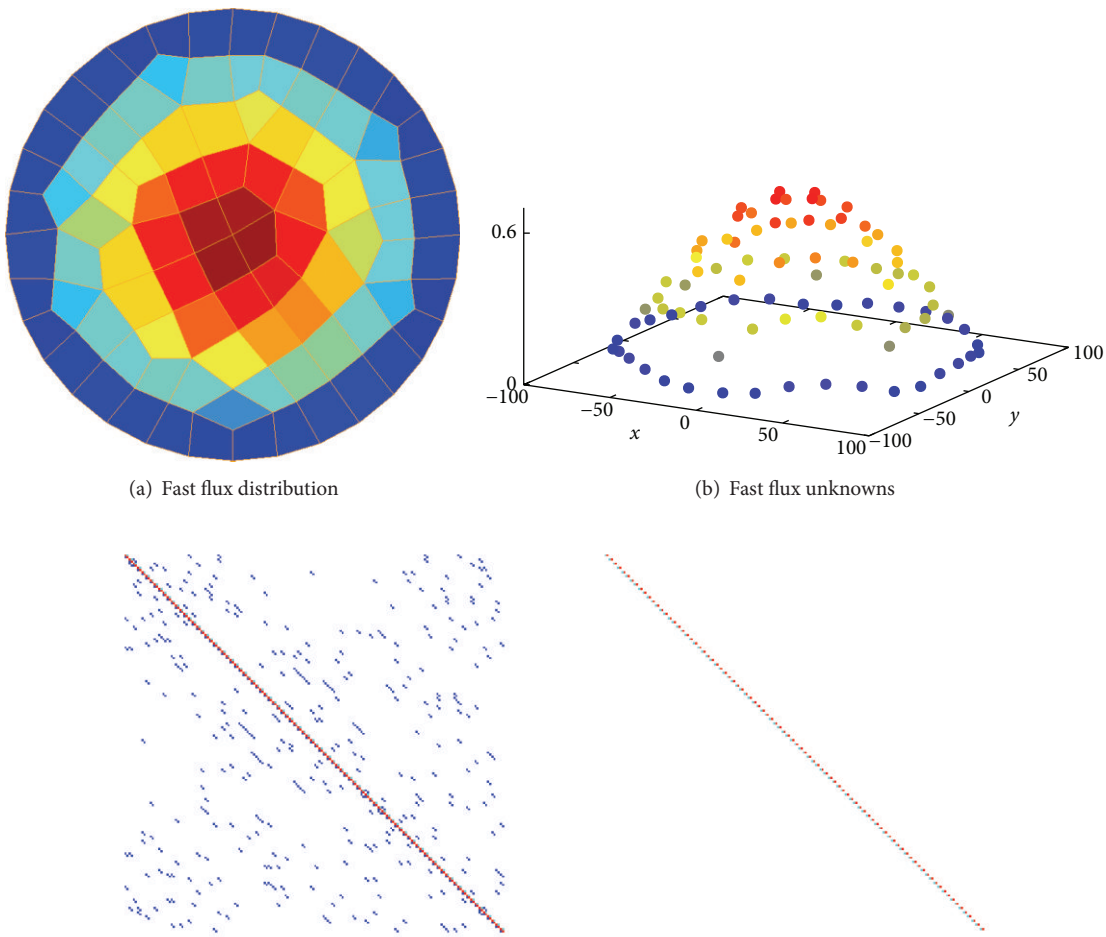


Figure 4: The finite volumes method computes the unknown flux in the cell centers (squares), whilst the finite elements method computes the fluxes at the nodes (circles).

at the same time, can be shown to be equivalent to the strong formulation given by (1).

In any case, both formulations involve the computation of G unknown functions of $\mathbf{x}$ and one unknown real value k_{eff}. Except for homogeneous cross sections in canonical domains U, the multigroup diffusion equation needs to be solved numerically. As discussed below, any nodal-based discretization scheme replaces a continuous unknown function $\phi_g(\mathbf{x})$ by N discrete values $\phi_g(i)$ for $i = 1, \ldots, N$. If we arrange these unknowns into a vector $\phi \in \mathbb{R}^{NG}$ such as

$$\phi = \begin{bmatrix} \phi_1(1) \\ \phi_2(1) \\ \vdots \\ \phi_G(1) \\ \phi_1(2) \\ \vdots \\ \phi_g(i) \\ \vdots \\ \phi_G(N) \end{bmatrix}, \tag{4}$$



(a) Fast flux distribution

(b) Fast flux unknowns

(c) Matrix R structure

(d) Matrix F structure

Figure 5: A bare homogeneous circle solved with finite volumes (184 unknowns).

then the continuous eigenvalue problem—in either formulation—can be transformed into a generalized matrix eigenvalue/eigenvector problem casted in either of the following forms:

$$\begin{aligned} R \cdot \phi &= \frac{1}{k_{\text{eff}}} \cdot F \cdot \phi, \\ F \cdot \phi &= k_{\text{eff}} \cdot R \cdot \phi, \\ R^{-1} \cdot F \cdot \phi &= k_{\text{eff}} \cdot \phi, \end{aligned} \tag{5}$$

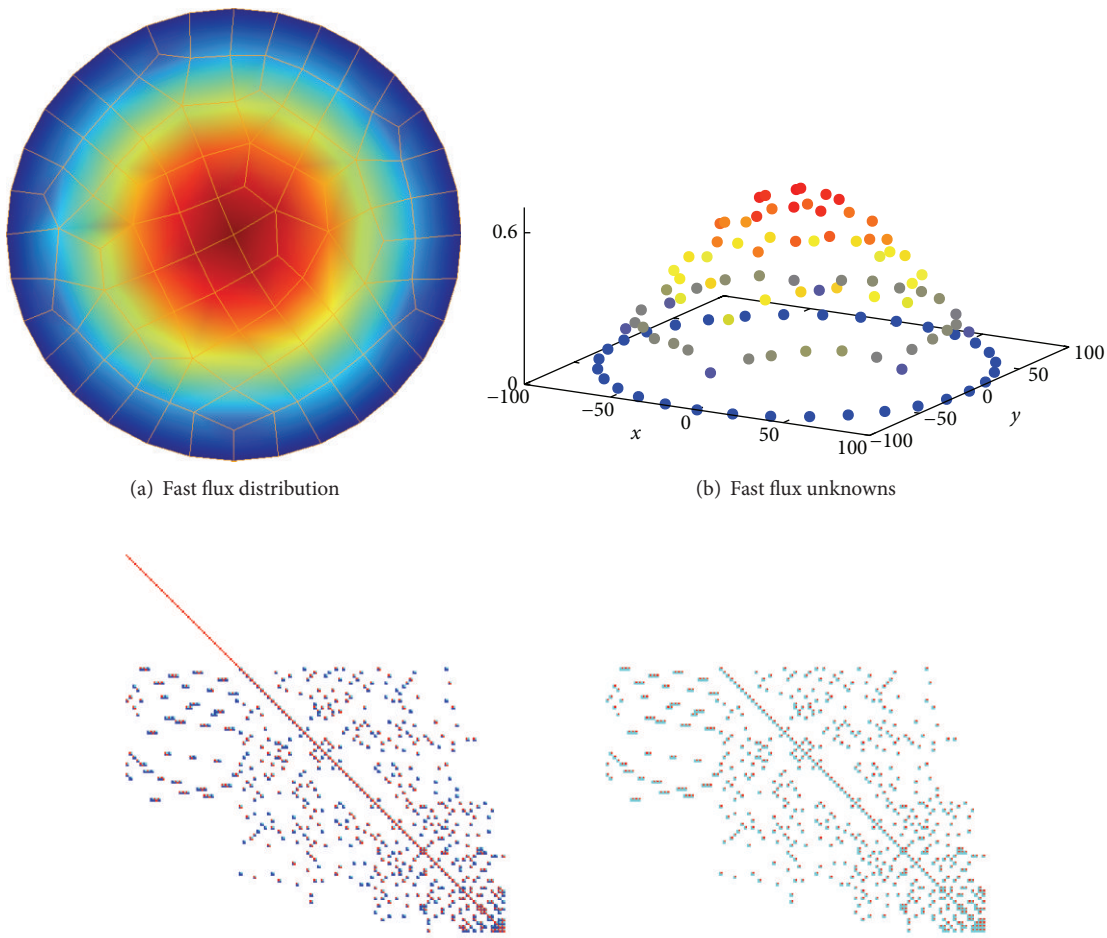
where R and F are square $NG \times NG$ matrices. We call F the fission matrix, as it contains all the ν-fission terms which are the ones that we artificially divided by k_{eff}. The rest of the terms are grouped into the removal or transport matrix R, which includes the rest of the neutron-matter interaction mechanisms. It can be shown [11] that, for any real set of cross sections, R^{-1} exists and that the NG pairs of eigenvalue/eigenvector solutions of (5) satisfy that

(1) there is a unique real positive eigenvalue greater in magnitude than any other eigenvalue,

(2) all the elements of the eigenvector corresponding to that eigenvalue are real and positive,

(3) all other eigenvectors either have some elements that are zero or have elements that differ in sign from each other.

2.1. Boundary Conditions. Being a differential equation over space, the neutron diffusion equation needs proper boundary conditions to conform a properly defined mathematical problem. These can be imposed flux (Dirichlet), imposed current (Neumann), or a linear combination (Robin). However, due to the fact that in the linear problem in absence of external sources—such as (1) or (3)—the problem is homogeneous; if $\phi_g(\mathbf{x})$ is a solution, then any multiple $\alpha\phi_g(\mathbf{x})$ is also a solution. That is to say, the eigenvectors are defined up to a multiplicative constant whose value is usually chosen as to obtain a certain total thermal power. Thus, the prescribed boundary conditions should also be homogeneous and be defined up to a multiplicative constant. Therefore, the allowed Dirichlet conditions are zero flux at the boundary; the Neumann conditions should prescribe that the derivative in



(a) Fast flux distribution

(b) Fast flux unknowns

(c) Matrix R structure

(d) Matrix F structure

Figure 6: A bare homogeneous circle solved with finite elements (218 unknowns).

the normal direction should be zero (i.e., symmetry condition), and the Robin conditions are restricted to the following form:

$$\text{grad}\ \phi_g(\mathbf{x}) \cdot \hat{\mathbf{n}} + a_g(\mathbf{x}) \cdot \phi_g(\mathbf{x}) = 0, \tag{6}$$

$\hat{\mathbf{n}}$ being the outward unit normal to the boundary ∂U of the domain U.

2.2. Grids and Schemes. One way of solving the neutron diffusion equation—and in general any partial differential equation over space—is by discretizing the differential operators with some kind of scheme that is applied over a certain spatial grid. Given an m-dimensional domain, a grid defines a partition composed of a finite number of simple geometric entities. In structured grids, these elementary entities are arranged following a well-defined structure in such a way that each entity can be identified without needing further information than the one provided by the intrinsic structure. On the other hand, the geometric entities that compose an unstructured grid are arranged in an irregular pattern, and the identification of the elementary entities needs to be separately specified, for example, by means of a sorted list. For instance, Figure 2(b) shows a structured grid that approximates the continuous domain of Figure 2(a). Each square may be uniquely identified by means of two integer indexes that indicate its relative position in each of the horizontal and vertical directions. In the case of Figure 2(c) that shows an unstructured grid, there is no way to systematically make a reference to a particular quadrangle without any further information.

Almost all of the grid-based schemes—which are known as nodal schemes, which are to be differentiated from modal schemes based on series expansions—are based in either the finite differences, volumes, or elements method. Finite differences schemes provide the most simple and basic approach to replace differential (i.e., continuous) operators by difference (i.e., discrete) approximations. However, they are not suitable for unstructured meshes and may introduce convergence problems with parameters that are discontinuous in space, which is the case for any reactor core composed of at least two different materials. Moreover, boundary conditions are hard to incorporate and usually give rise to incorrect results.

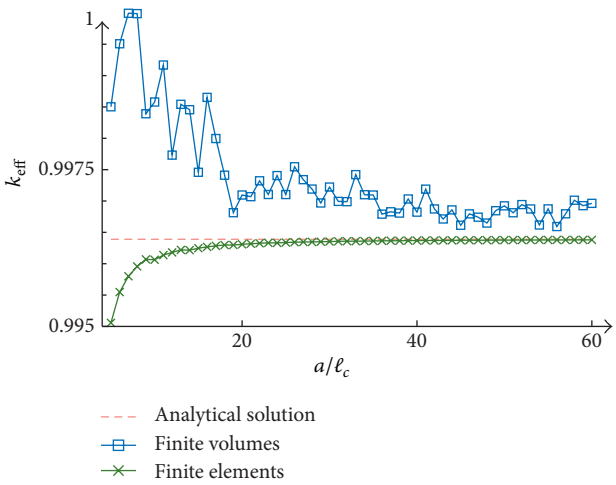


Figure 7: The effective multiplication factor of a two-group bare circular reactor of radius a as a function of the quotient a/ℓ_c between the radius and the characteristic length of the cell/element.

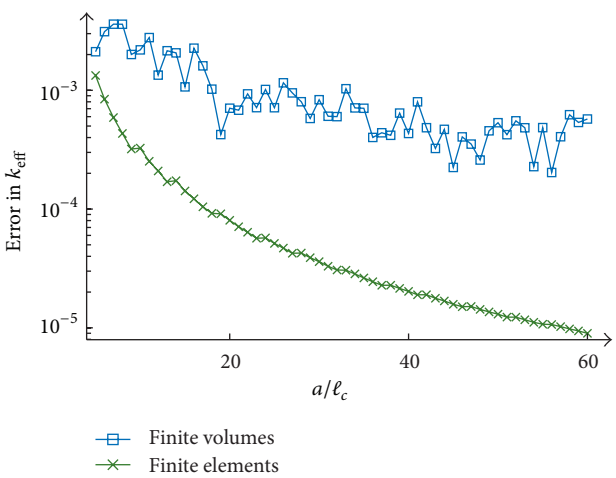


Figure 8: Absolute value of the error committed in the computation of k_{eff} versus a/ℓ_c.

Methods of the finite volumes family involve the integration of the differential equation over each elementary entity, applying the divergence theorem to transform volume integrals into surface integrals and providing a mean to estimate the fluxes through the entity's surface using information contained in its neighbors. In this context, each elementary entity is called a cell, and finite volumes methods give the mean value of each of the group fluxes $\phi_g(i)$ at the ith cell, which may be associated to the value of $\phi_g(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the location of the cell barycenter. Being an integral-based method, spatial-discontinuous parameters are handled more efficiently than in finite differences, and boundary conditions can be easily incorporated as forced cell fluxes. Nonetheless, even though these methods may be applied to unstructured grids, the estimation of the fluxes on the cells' surfaces is usually performed by using some geometric approximations that may lose validity as the quality of the grid is worsened.

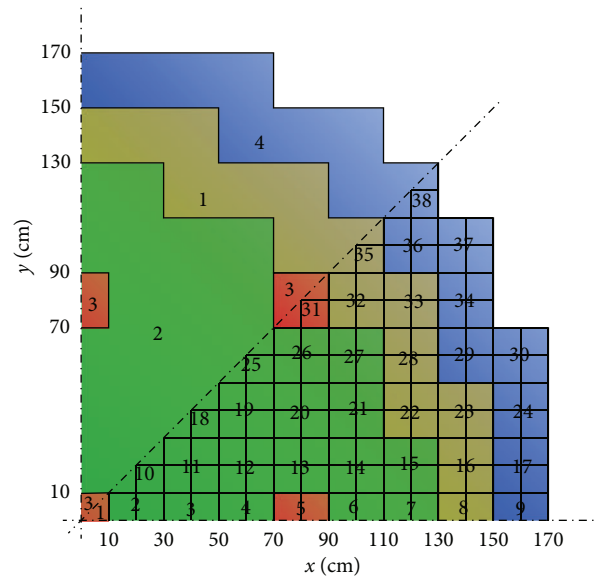
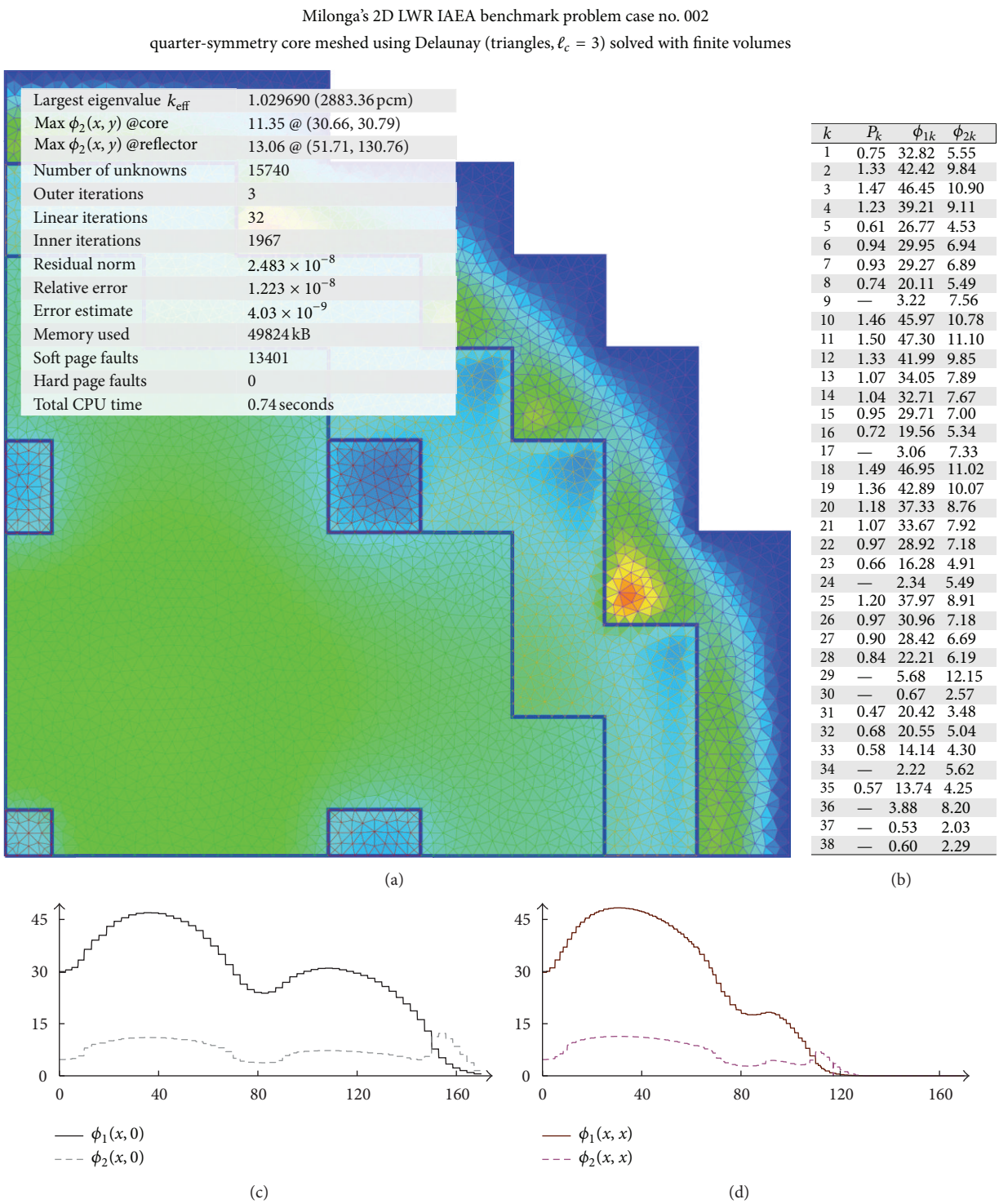


Figure 9: The 2D IAEA PWR benchmark geometry.

Moreover, the simple integral approach cannot take into account discontinuous diffusion coefficients, so when two neighbors pertain to different materials the flux has to be computed in a certain particular way to conserve the neutron current according to (2).

Finally, finite elements methods rely on a weak formulation of the differential problem similar to (3) that maintains all the mathematical characteristics of the original strong formulation plus its boundary conditions. The method is based on shape functions that are local to each elementary entity—now called element, defined by nodes as corners—and on finding a set of nodal values such that a certain condition is met, which is usually that the residual of the solution has to be orthogonal to each of the shape functions. This condition is known as the Galerkin method and implies that the error committed in the approximate solution of the continuous problem is confined into a small subset of the original vector space of the continuous functions. These mathematical properties make finite elements schemes very attractive. However, these features depend on a large number of integrations that ought to be performed numerically, so a computational effort/desired accuracy tradeoff has to be considered. Not only does the finite elements method give the values of the flux $\phi_g(\mathbf{x}_i)$ at the ith node but also the shape functions provide explicit expressions to interpolate and to evaluate the unknown functions at any location $\mathbf{x}$ of the domain U. Boundary conditions are divided into essential and natural. The first group comprises the Dirichlet boundary conditions which are satisfied exactly—within the precision of the eigenvalue problem solver—by the obtained solution. The latter include the Neumann and the Robin conditions that are satisfied only approximately by the derivatives of the interpolated solution through the shape functions with finer meshes giving better agreement with the prescribed values.

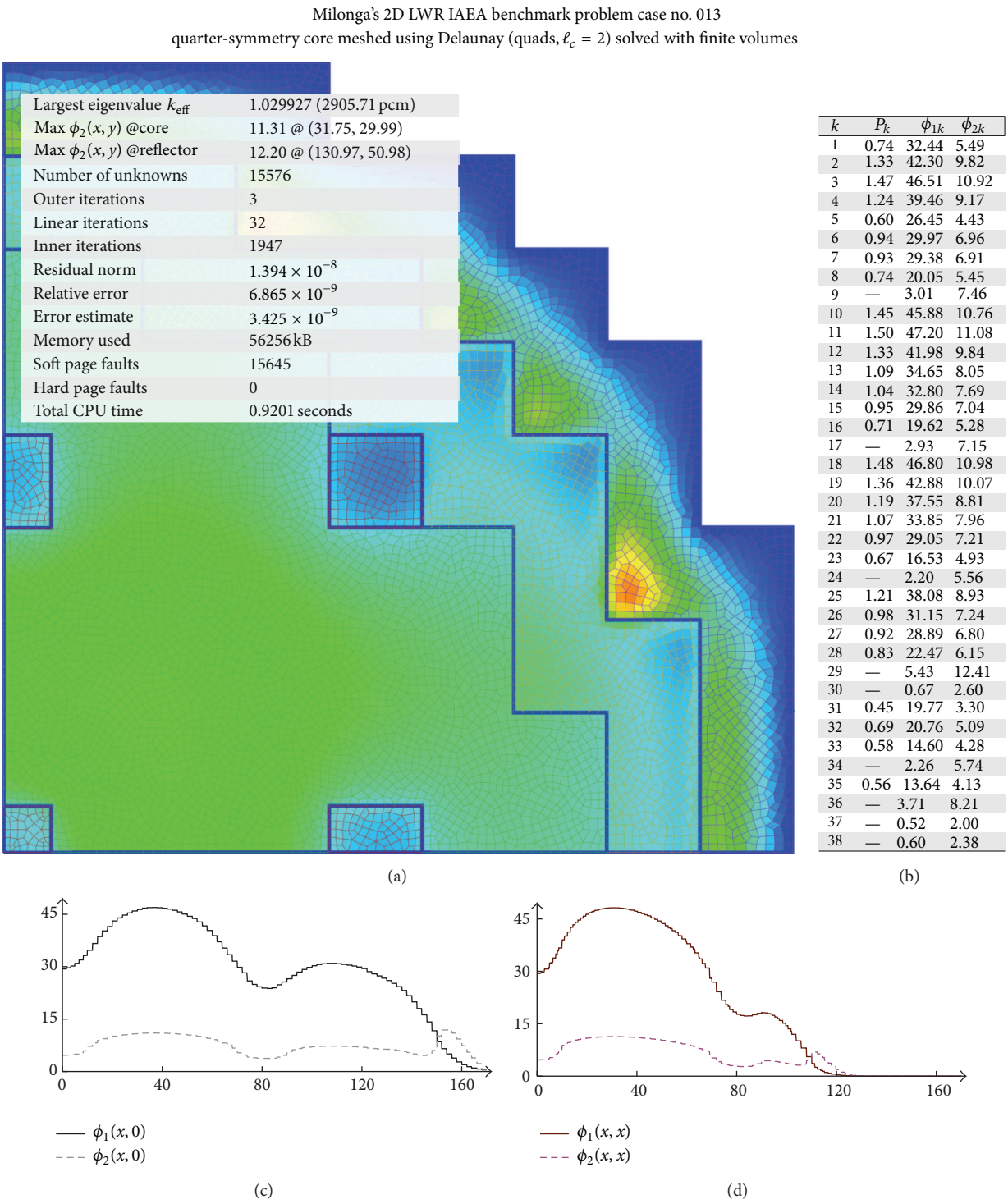
The same unstructured grid may be used either for the finite volumes or for the finite elements method. In the first



k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.75	32.82	5.55
2	1.33	42.42	9.84
3	1.47	46.45	10.90
4	1.23	39.21	9.11
5	0.61	26.77	4.53
6	0.94	29.95	6.94
7	0.93	29.27	6.89
8	0.74	20.11	5.49
9	—	3.22	7.56
10	1.46	45.97	10.78
11	1.50	47.30	11.10
12	1.33	41.99	9.85
13	1.07	34.05	7.89
14	1.04	32.71	7.67
15	0.95	29.71	7.00
16	0.72	19.56	5.34
17	—	3.06	7.33
18	1.49	46.95	11.02
19	1.36	42.89	10.07
20	1.18	37.33	8.76
21	1.07	33.67	7.92
22	0.97	28.92	7.18
23	0.66	16.28	4.91
24	—	2.34	5.49
25	1.20	37.97	8.91
26	0.97	30.96	7.18
27	0.90	28.42	6.69
28	0.84	22.21	6.19
29	—	5.68	12.15
30	—	0.67	2.57
31	0.47	20.42	3.48
32	0.68	20.55	5.04
33	0.58	14.14	4.30
34	—	2.22	5.62
35	0.57	13.74	4.25
36	—	3.88	8.20
37	—	0.53	2.03
38	—	0.60	2.29

FIGURE 10: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

case, the unknowns are the mean value of the fluxes over each cell, whilst in the latter the unknowns are the fluxes evaluated at each node. Therefore, the number of unknowns NG is different for each method, even when using the same grid. Figure 4 shows this situation, and in Section 3.1, we further illustrate these differences. A fully detailed mathematical description of the actual algorithms for both volumes and elements-based proposed discretizations can be found in an academic monograph written by the author of this paper [13].

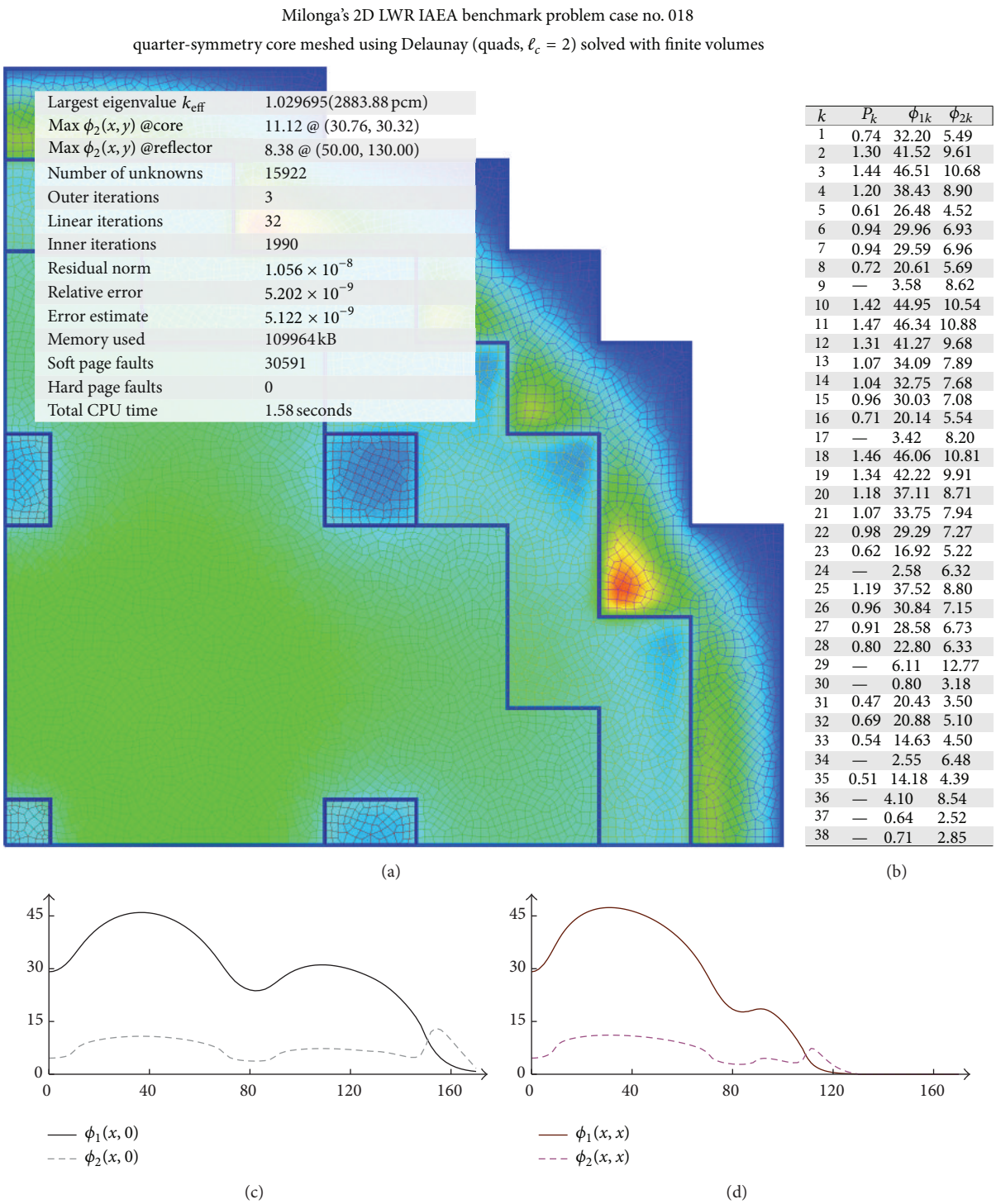


k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.74	32.44	5.49
2	1.33	42.30	9.82
3	1.47	46.51	10.92
4	1.24	39.46	9.17
5	0.60	26.45	4.43
6	0.94	29.97	6.96
7	0.93	29.38	6.91
8	0.74	20.05	5.45
9	—	3.01	7.46
10	1.45	45.88	10.76
11	1.50	47.20	11.08
12	1.33	41.98	9.84
13	1.09	34.65	8.05
14	1.04	32.80	7.69
15	0.95	29.86	7.04
16	0.71	19.62	5.28
17	—	2.93	7.15
18	1.48	46.80	10.98
19	1.36	42.88	10.07
20	1.19	37.55	8.81
21	1.07	33.85	7.96
22	0.97	29.05	7.21
23	0.67	16.53	4.93
24	—	2.20	5.56
25	1.21	38.08	8.93
26	0.98	31.15	7.24
27	0.92	28.89	6.80
28	0.83	22.47	6.15
29	—	5.43	12.41
30	—	0.67	2.60
31	0.45	19.77	3.30
32	0.69	20.76	5.09
33	0.58	14.60	4.28
34	—	2.26	5.74
35	0.56	13.64	4.13
36	—	3.71	8.21
37	—	0.52	2.00
38	—	0.60	2.38

Figure 11: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

3. Results

We now proceed to show two illustrative results that are to be taken as an overview of the possibilities that unstructured grids can provide in order to tackle the multigroup neutron diffusion problem. The examples are two-dimensional problems, as they contain some of the complexities a real three-dimensional reactor posse, yet the reported results are not so complicated as to be easily understood and analyzed. In particular, we state some basic differences between the finite

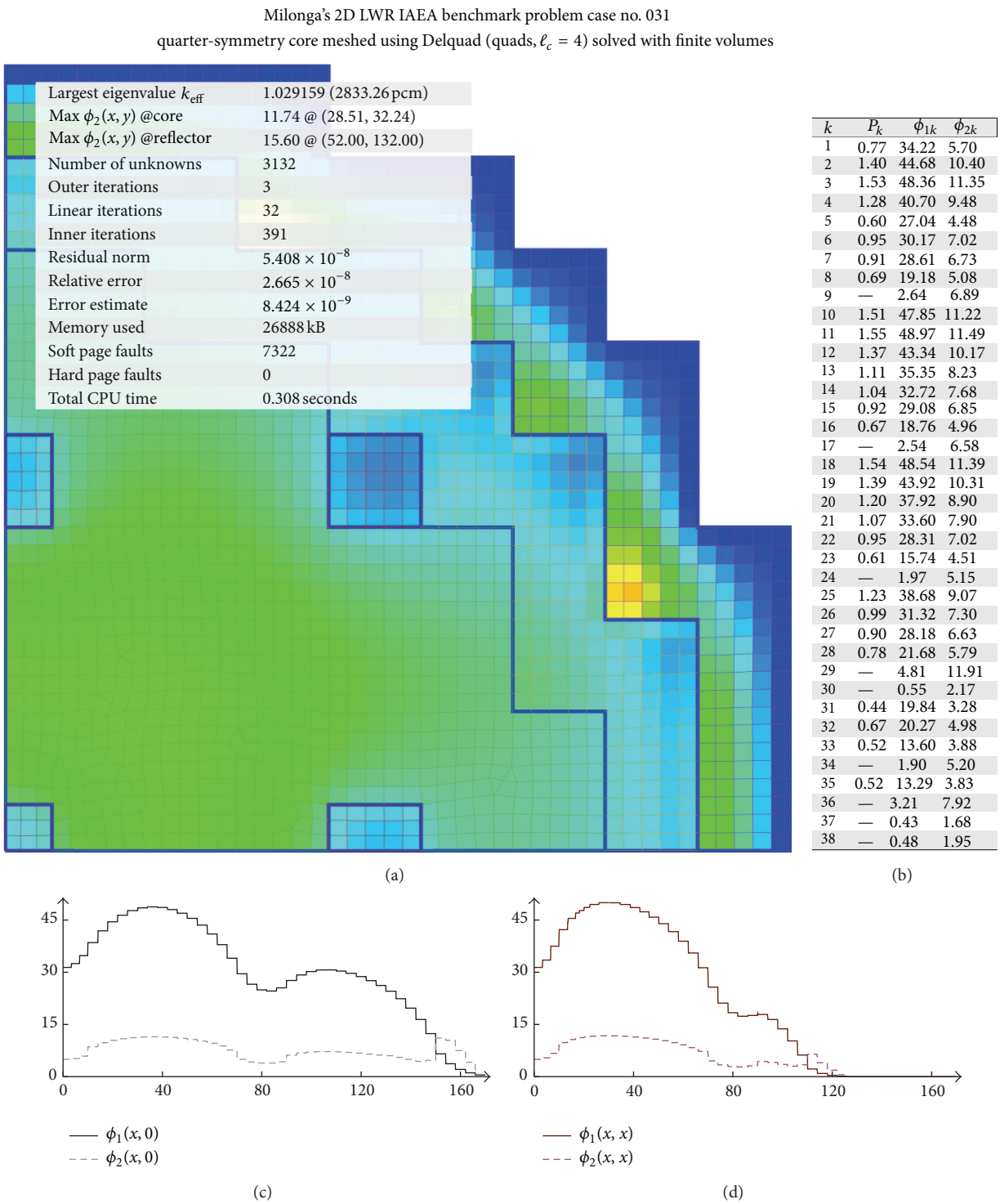


k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.74	32.20	5.49
2	1.30	41.52	9.61
3	1.44	46.51	10.68
4	1.20	38.43	8.90
5	0.61	26.48	4.52
6	0.94	29.96	6.93
7	0.94	29.59	6.96
8	0.72	20.61	5.69
9	—	3.58	8.62
10	1.42	44.95	10.54
11	1.47	46.34	10.88
12	1.31	41.27	9.68
13	1.07	34.09	7.89
14	1.04	32.75	7.68
15	0.96	30.03	7.08
16	0.71	20.14	5.54
17	—	3.42	8.20
18	1.46	46.06	10.81
19	1.34	42.22	9.91
20	1.18	37.11	8.71
21	1.07	33.75	7.94
22	0.98	29.29	7.27
23	0.62	16.92	5.22
24	—	2.58	6.32
25	1.19	37.52	8.80
26	0.96	30.84	7.15
27	0.91	28.58	6.73
28	0.80	22.80	6.33
29	—	6.11	12.77
30	—	0.80	3.18
31	0.47	20.43	3.50
32	0.69	20.88	5.10
33	0.54	14.63	4.50
34	—	2.55	6.48
35	0.51	14.18	4.39
36	—	4.10	8.54
37	—	0.64	2.52
38	—	0.71	2.85

Figure 12: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

volumes and the finite elements methods by solving a two-group homogeneous bare reactor with the Robin boundary conditions over the very same grid, although a rather coarse one, so the differences can be observed directly into the resulting figures. We then solve the classical two-dimensional LWR problem, also known as the 2D IAEA PWR benchmark. Not only do we show again the differences between the finite volumes and elements formulation but also we solve the problem using different combinations of meshing algorithms, basic shapes, and characteristic lengths of the mesh.

To solve the two examples shown below, we employed the milonga code, which was written from scratch by the author



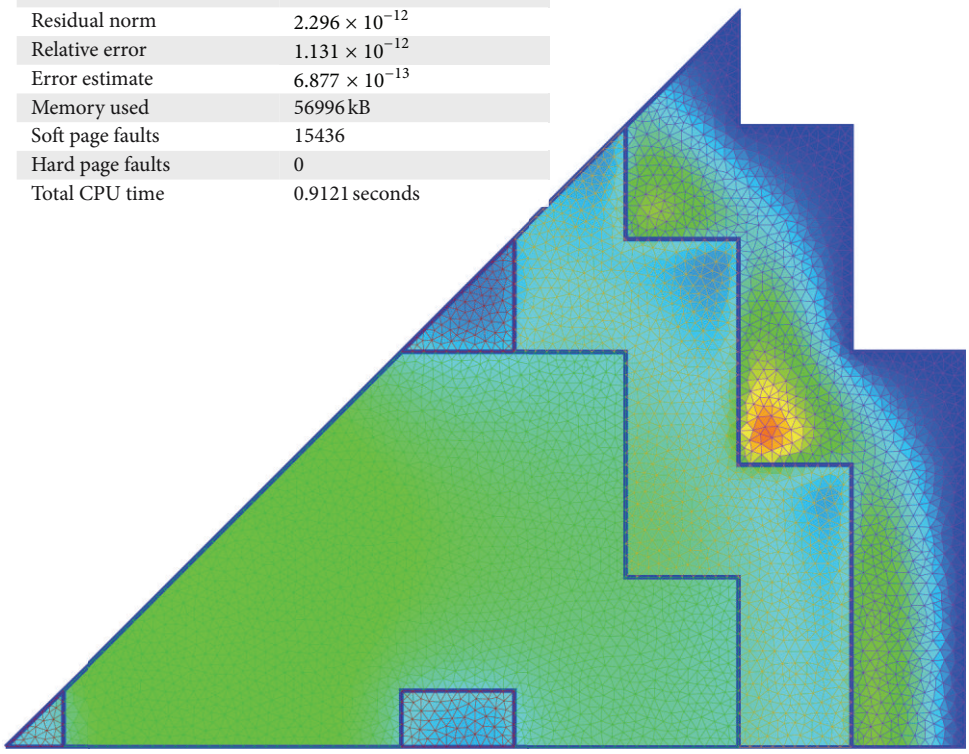
k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.77	34.22	5.70
2	1.40	44.68	10.40
3	1.53	48.36	11.35
4	1.28	40.70	9.48
5	0.60	27.04	4.48
6	0.95	30.17	7.02
7	0.91	28.61	6.73
8	0.69	19.18	5.08
9	—	2.64	6.89
10	1.51	47.85	11.22
11	1.55	48.97	11.49
12	1.37	43.34	10.17
13	1.11	35.35	8.23
14	1.04	32.72	7.68
15	0.92	29.08	6.85
16	0.67	18.76	4.96
17	—	2.54	6.58
18	1.54	48.54	11.39
19	1.39	43.92	10.31
20	1.20	37.92	8.90
21	1.07	33.60	7.90
22	0.95	28.31	7.02
23	0.61	15.74	4.51
24	—	1.97	5.15
25	1.23	38.68	9.07
26	0.99	31.32	7.30
27	0.90	28.18	6.63
28	0.78	21.68	5.79
29	—	4.81	11.91
30	—	0.55	2.17
31	0.44	19.84	3.28
32	0.67	20.27	4.98
33	0.52	13.60	3.88
34	—	1.90	5.20
35	0.52	13.29	3.83
36	—	3.21	7.92
37	—	0.43	1.68
38	—	0.48	1.95

Figure 13: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

of this paper and is currently still under development within his ongoing PhD thesis. There exists a first public release [14] under the terms of the GNU General Public License—that is, it is a free software—that can only handle structured grids. There is a second release being prepared, whose main relevance is that it can work with nonstructured grids as well, which is the main feature of the code. It uses a general mathematical framework—coded from scratch as well—which provides input file parsing, algebraic expressions evaluation, one- and multidimensional interpolation of

Milonga's 2D LWR IAEA benchmark problem case no. 043
eighth-symmetry core meshed using Delaunaya (triangs, $\ell_c = 2$) solved with finite volumes

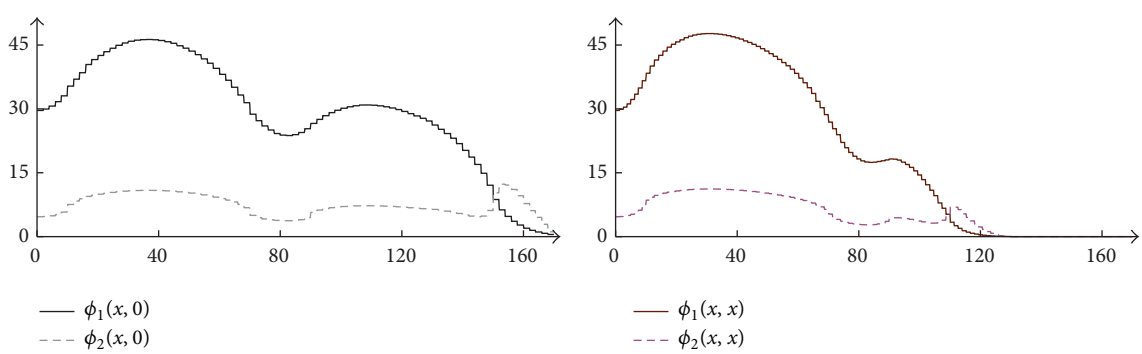
Largest eigenvalue k_{eff}	1.029788 (2892.60 pcm)
Max $\phi_2(x, y)$ @core	11.20 @ (31.49, 30.11)
Max $\phi_2(x, y)$ @reflector	10.91 @ (130.38, 51.14)
Number of unknowns	15776
Outer iterations	3
Linear iterations	32
Inner iterations	1972
Residual norm	2.296×10^{-12}
Relative error	1.131×10^{-12}
Error estimate	6.877×10^{-13}
Memory used	56996 kB
Soft page faults	15436
Hard page faults	0
Total CPU time	0.9121 seconds



(a)

k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.74	32.41	5.48
2	1.31	41.88	9.71
3	1.45	45.82	10.76
4	1.22	38.77	9.00
5	0.60	26.44	4.45
6	0.93	29.86	6.92
7	0.93	29.41	6.92
8	0.75	20.30	5.52
9	—	3.22	7.79
10	1.43	45.23	10.61
11	1.48	46.67	10.95
12	1.31	41.45	9.72
13	1.07	34.23	7.94
14	1.03	32.69	7.67
15	0.95	29.93	7.06
16	0.73	19.88	5.39
17	—	3.12	7.51
18	1.47	46.37	10.88
19	1.34	42.43	9.96
20	1.18	37.21	8.73
21	1.07	33.70	7.93
22	0.98	29.14	7.23
23	0.67	16.51	4.99
24	—	2.35	5.71
25	1.19	37.63	8.82
26	0.96	30.77	7.15
27	0.91	28.47	6.71
28	0.83	22.55	6.17
29	—	5.76	12.25
30	—	0.68	2.66
31	0.46	20.16	3.41
32	0.68	20.58	5.05
33	0.59	14.43	4.35
34	—	2.33	5.89
35	0.57	13.81	4.20
36	—	3.81	8.23
37	—	0.54	2.08
38	—	0.62	2.44

(b)



(c) (d)

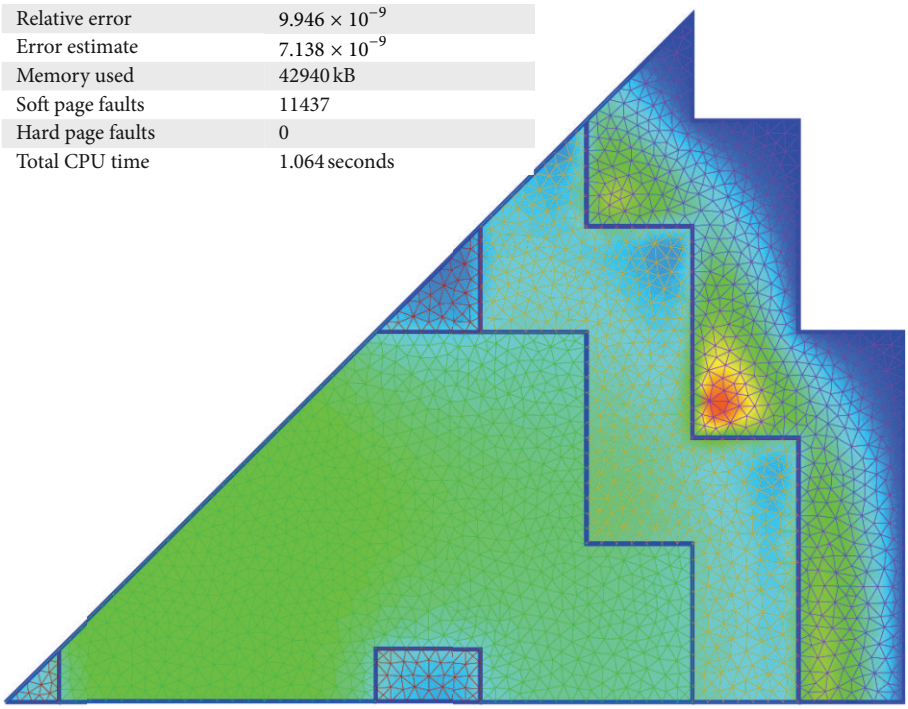
FIGURE 14: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

scattered data, shared-memory access, numerical integration facilities, and so forth. The milonga code was designed with four design basis vectors in mind—which are thoroughly discussed in the documentation [15]—which includes the type of problems it can handle, the code scalability, which features are expected, and what to do with the obtained results.

It works by first reading an input file that, using plain-text English keywords and arguments, defines the number m of spatial dimensions, the number G of group energies,

Milonga's 2D LWR IAEA benchmark problem case no. 047
eighth-symmetry core meshed using Delaunay (triangles, $\ell_c = 3$) solved with finite volumes

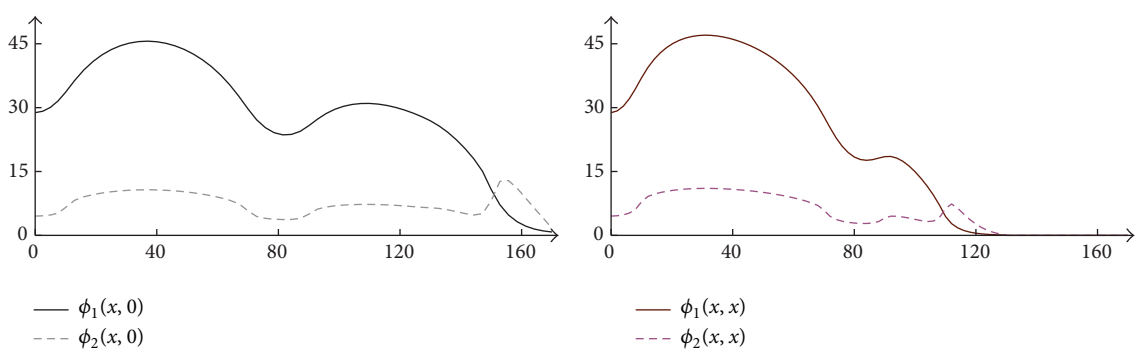
Largest eigenvalue k_{eff}	1.029726 (2886.75 pcm)
Max $\phi_2(x, y)$ @core	11.04 @ (30.69, 30.69)
Max $\phi_2(x, y)$ @reflector	8.64 @ (130.00, 50.00)
Number of unknowns	4040
Outer iterations	2
Linear iterations	24
Inner iterations	505
Residual norm	2.019×10^{-8}
Relative error	9.946×10^{-9}
Error estimate	7.138×10^{-9}
Memory used	42940 kB
Soft page faults	11437
Hard page faults	0
Total CPU time	1.064 seconds



(a)

k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.74	31.94	5.45
2	1.29	41.17	9.53
3	1.43	45.15	10.60
4	1.19	38.14	8.83
5	0.61	26.34	4.50
6	0.93	29.84	6.90
7	0.94	29.54	6.95
8	0.71	20.64	5.71
9	—	3.62	8.65
10	1.41	44.58	10.45
11	1.46	45.99	10.79
12	1.30	40.97	9.61
13	1.06	33.88	7.84
14	1.03	32.63	7.65
15	0.95	29.98	7.07
16	0.68	20.15	5.57
17	—	3.46	8.23
18	1.45	45.72	10.73
19	1.33	41.94	9.84
20	1.17	36.91	8.66
21	1.07	33.63	7.92
22	0.98	29.25	7.26
23	0.59	16.96	5.28
24	—	2.63	6.35
25	1.18	37.28	8.74
26	0.96	30.68	7.11
27	0.91	28.50	6.72
28	0.79	22.80	6.35
29	—	6.19	12.76
30	—	0.81	3.21
31	0.47	20.36	3.49
32	0.68	20.85	5.09
33	0.51	14.65	4.54
34	—	2.58	6.50
35	0.49	14.19	4.43
36	—	4.16	8.54
37	—	0.65	2.54
38	—	0.71	2.88

(b)



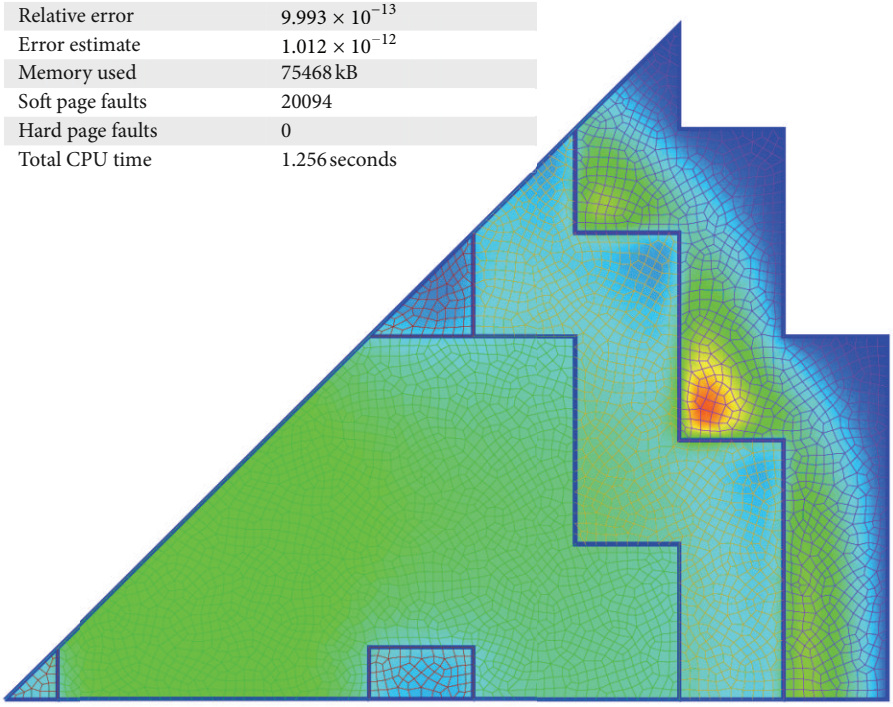
(c) (d)

Figure 15: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

and optionally a mesh file. Currently, only grids generated with the code gmsh [16] are only supported, mainly because it is also a free software and it suits perfectly well milonga's design basis in the sense that the continuous geometry can be defined as a function of a number of parameters. Afterward, the physical entities defined in the grid are mapped into materials with macroscopic cross sections, which may depend on the spatial coordinates $\mathbf{x}$ by means of intermediate functions such as burn-up or temperatures distributions, which in turn may be defined by algebraic expressions, by interpolating data located in files, by reading shared-memory objects or by a combination of them. In the same sense,

Milonga's 2D LWR IAEA benchmark problem case no. 058
eighth-symmetry core meshed using Delaunay (quads, $\ell_c = 2$) solved with finite volumes

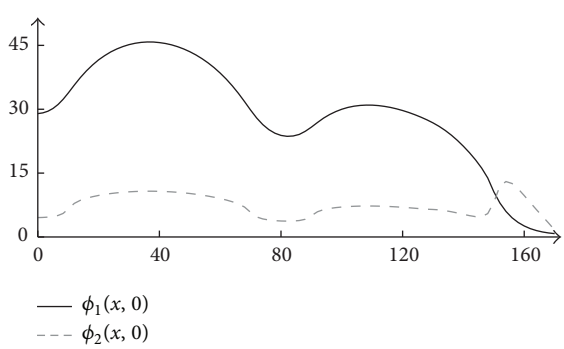
Largest eigenvalue k_{eff}	1.029695 (2883.87 pcm)
Max $\phi_2(x, y)$ @core	11.08 @ (30.45, 30.45)
Max $\phi_2(x, y)$ @reflector	8.33 @ (130.00, 50.00)
Number of unknowns	8234
Outer iterations	3
Linear iterations	32
Inner iterations	1029
Residual norm	2.028×10^{-12}
Relative error	9.993×10^{-13}
Error estimate	1.012×10^{-12}
Memory used	75468 kB
Soft page faults	20094
Hard page faults	0
Total CPU time	1.256 seconds



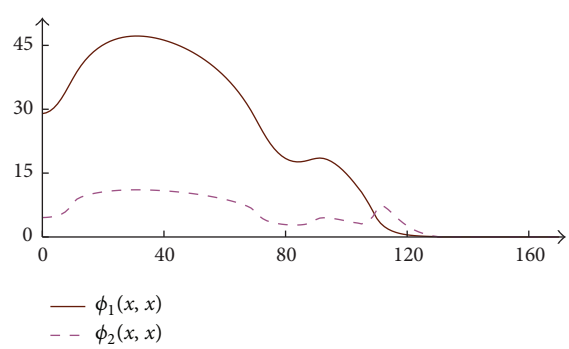
(a)

k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.74	32.07	5.47
2	1.29	41.36	9.57
3	1.44	45.34	10.64
4	1.20	38.28	8.87
5	0.61	26.38	4.50
6	0.93	29.85	6.90
7	0.94	29.48	6.94
8	0.72	20.53	5.66
9	—	3.57	8.59
10	1.42	44.78	10.50
11	1.46	46.17	10.84
12	1.30	41.11	9.64
13	1.06	33.96	7.86
14	1.03	32.63	7.65
15	0.95	29.91	7.06
16	0.71	20.05	5.52
17	—	3.40	8.16
18	1.45	45.88	10.77
19	1.33	42.06	9.87
20	1.17	36.98	8.68
21	1.07	33.62	7.91
22	0.98	29.18	7.25
23	0.62	16.86	5.20
24	—	2.58	6.29
25	1.18	37.37	8.76
26	0.96	30.73	7.12
27	0.91	28.48	6.71
28	0.80	22.72	6.31
29	—	6.09	12.72
30	—	0.80	3.17
31	0.47	20.35	3.48
32	0.69	20.81	5.09
33	0.54	14.58	4.48
34	—	2.55	6.46
35	0.52	14.13	4.37
36	—	4.09	8.51
37	—	0.64	2.51
38	—	0.71	2.84

(b)



(c)



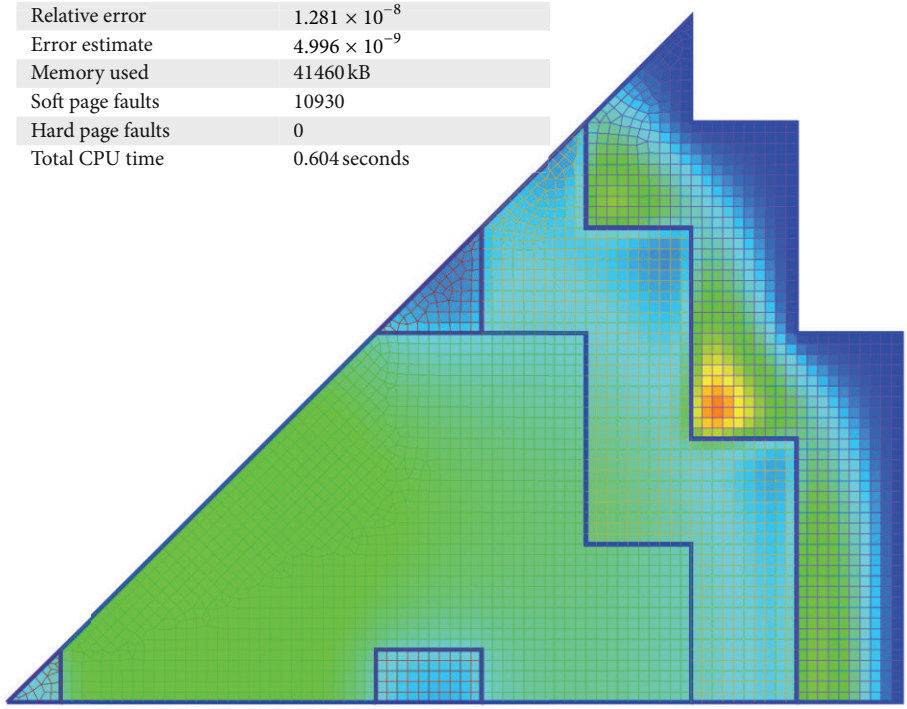
(d)

Figure 16: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

boundary conditions are applied to appropriate physical entities of dimension $m-1$. The problem matrices R and F are then built and stored in an appropriate sparse format using the free PETSc [17, 18] library, and the eigenvalue problem is solved using the free SLEPc [19] library. The results are stored into milonga's variables and functions, which may be evaluated, integrated—usually using the free GNU Scientific Library [20] routines—and of course written into appropriate outputs. Milonga can also solve problems parametrically and be used to solve optimization problems. As stated above, the code is a free software so corrections and contributions are more than welcome.

Milonga's 2D LWR IAEA benchmark problem case no. 073
eighth-symmetry core meshed using Delaunay (quads, $\ell_c = 2$) solved with finite volumes

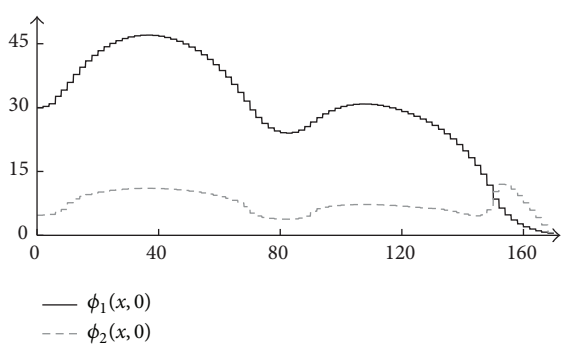
Largest eigenvalue k_{eff}	1.029462 (2861.85 pcm)
Max $\phi_2(x, y)$ @core	11.36 @ (30.93, 29.53)
Max $\phi_2(x, y)$ @reflector	12.33 @ (131.00, 51.00)
Number of unknowns	6228
Outer iterations	2
Linear iterations	24
Inner iterations	778
Residual norm	2.6×10^{-8}
Relative error	1.281×10^{-8}
Error estimate	4.996×10^{-9}
Memory used	41460 kB
Soft page faults	10930
Hard page faults	0
Total CPU time	0.604 seconds



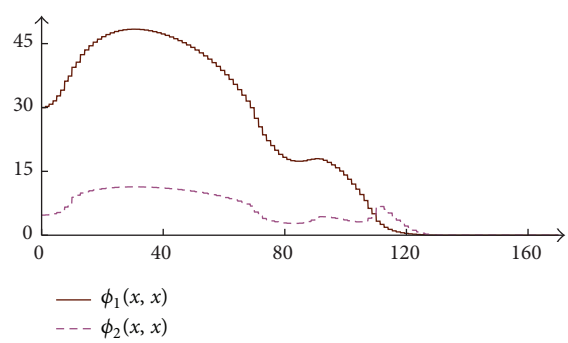
(a)

k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.75	32.97	5.55
2	1.34	42.68	9.92
3	1.48	46.60	10.94
4	1.23	39.26	9.14
5	0.59	26.45	4.40
6	0.94	29.97	6.97
7	0.93	29.14	6.85
8	0.72	20.02	5.34
9	—	3.01	7.73
10	1.46	46.06	10.80
11	1.50	47.35	11.11
12	1.33	42.12	9.88
13	1.08	34.48	8.02
14	1.04	32.71	7.67
15	0.94	29.54	6.96
16	0.70	19.54	5.21
17	—	2.88	7.35
18	1.49	46.91	11.01
19	1.36	42.89	10.07
20	1.19	37.48	8.80
21	1.07	33.59	7.90
22	0.96	28.80	7.14
23	0.64	16.36	4.75
24	—	2.17	5.63
25	1.21	38.10	8.94
26	0.98	31.10	7.24
27	0.90	28.29	6.66
28	0.81	22.27	5.99
29	—	5.36	12.57
30	—	0.62	2.50
31	0.45	20.02	3.36
32	0.68	20.40	5.00
33	0.55	14.14	4.09
34	—	2.15	5.79
35	0.55	13.59	4.08
36	—	3.59	8.34
37	—	0.50	1.99
38	—	0.61	2.36

(b)



(c)



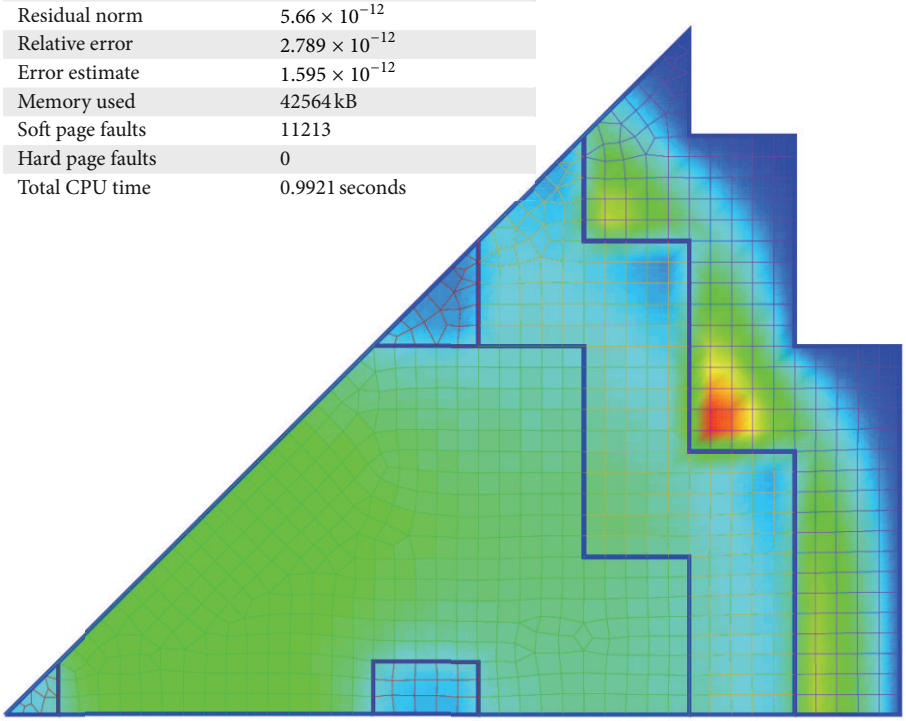
(d)

Figure 17: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

3.1. A Coarse Bare Homogeneous Circle. Figures 5 and 6 show the results of solving a bare two-dimensional circular homogeneous reactor of radius a over an unstructured grid which is deliberately coarse, using the finite volumes method and the finite elements method, respectively. Two group energies were used and a null-flux boundary condition was fixed at the external surface. Figures 5(a) and 6(a) compare the obtained fast flux distributions. In the first case, the numerical solution provides mean values for each neutron flux group in each cell, while in the latter, the solution is computed at the nodes, and

Milonga's 2D LWR IAEA benchmark problem case no. 076
eighth-symmetry core meshed using Delaunay (quads, $\ell_c = 4$) solved with finite volumes

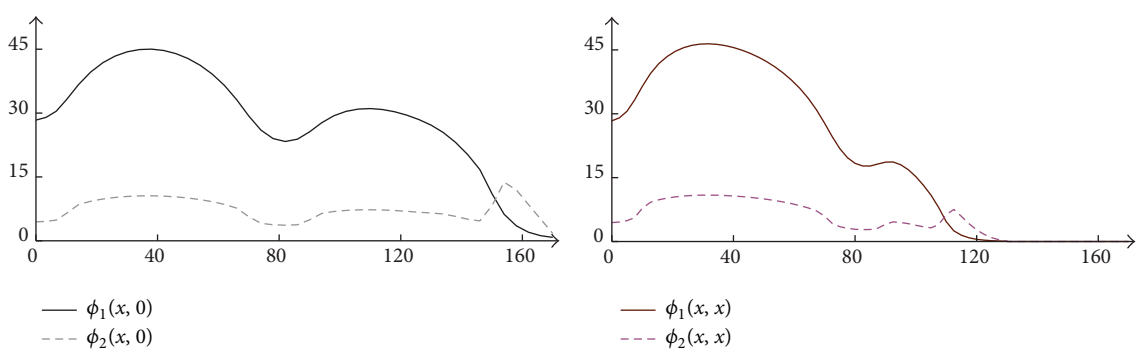
Largest eigenvalue k_{eff}	1.029851 (2898.58 pcm)
Max $\phi_2(x, y)$ @core	10.90 @ (31.82, 31.82)
Max $\phi_2(x, y)$ @reflector	8.51 @ (130.00, 50.00)
Number of unknowns	1752
Outer iterations	3
Linear iterations	32
Inner iterations	219
Residual norm	5.66×10^{-12}
Relative error	2.789×10^{-12}
Error estimate	1.595×10^{-12}
Memory used	42564 kB
Soft page faults	11213
Hard page faults	0
Total CPU time	0.9921 seconds



(a)

k	P_k	ϕ_{1k}	ϕ_{2k}
1	0.73	31.51	5.40
2	1.27	40.59	9.39
3	1.41	44.57	10.46
4	1.18	37.68	8.71
5	0.61	26.15	4.50
6	0.93	29.79	6.88
7	0.94	29.71	6.99
8	0.67	20.92	5.82
9	—	3.75	8.85
10	1.39	43.97	10.31
11	1.44	45.41	10.66
12	1.28	40.52	9.50
13	1.05	33.65	7.78
14	1.03	32.59	7.64
15	0.96	30.16	7.12
16	0.66	20.45	5.68
17	—	3.59	8.41
18	1.43	45.20	10.61
19	1.32	41.55	9.75
20	1.16	36.72	8.62
21	1.07	33.66	7.92
22	0.98	29.51	7.33
23	0.54	17.28	5.43
24	—	2.73	6.51
25	1.17	37.00	8.68
26	0.95	30.57	7.08
27	0.91	28.60	6.74
28	0.74	23.11	6.48
29	—	6.42	13.00
30	—	0.84	3.32
31	0.48	20.41	3.53
32	0.69	21.03	5.13
33	0.46	14.93	4.67
34	—	2.69	6.67
35	0.46	14.44	4.55
36	—	4.32	8.71
37	—	0.67	2.63
38	—	0.74	2.97

(b)



(c) (d)

FIGURE 18: (a) Mesh and thermal flux distribution. (b) Power and fluxes. (c) Flux distribution $\phi_g(x, 0)$ along the x-axis. (d) Flux distribution $\phi_g(x, x)$ along the diagonal.

continuous functions are evaluated by means of the shape functions used in the formulation. Figures 5(b) and 6(b) illustrate the fast fluxes unknowns and its relative position in space. It can be noted that the mesh coarseness gives results that differ substantially in both cases. Finally, the structure of the sparse eigenvalue problem matrices is shown for each case with blue, red, and cyan representing positive, negative, and explicitly inserted zero values. In the finite volume case, there are 184 unknowns (92 cells × 2 groups), while in the second case there are 218 unknowns (109 nodes × 2 groups). The volumes' fission matrix is almost diagonal: it has a bandwidth equal to the number of energy groups, which in this case is

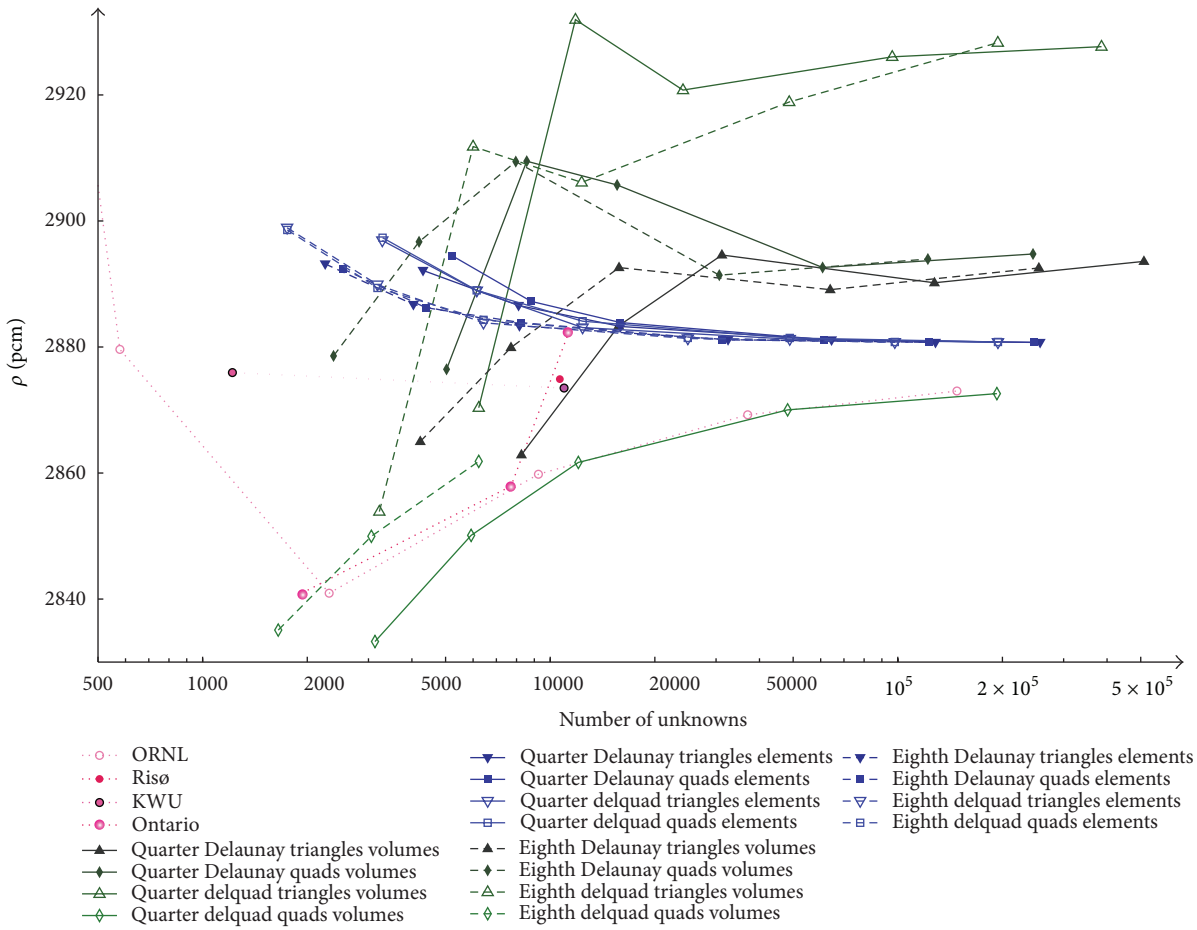


Figure 19: Static reactivity versus number of unknowns. The four original solutions as published in 1977 [10] are included as reference.

two. The off-diagonal values appear right next to the diagonal elements because of the chosen ordering of the unknowns in the flux vector $\phi \in \mathbb{R}^{184}$. Other orderings may be used, but the rate of convergence of the eigenvalue problem can be deteriorated. On the other hand, the elements' fission matrix has a nontrivial structure, because the grid is unstructured, and the net fission rate at each element depends on the fluxes at the nodes whose location inside the matrix depends in turn on how the grid was generated. This effect is similar to the one that appears in structural analysis where mass matrices need to be lumped in order to simplify transient computations [21]. The diagonal block of element's R matrix and the null block in F correspond to the discrete equations that set the null-flux boundary conditions on the nodes located at the external surface. Other types of boundary conditions lead to different kinds of structures within the problem matrices.

In case a part of the domain contained a nonmultiplicative material such as a reflector, then there would appear sections of the fission matrix with null values in both methods, rendering F singular in both methods. Care should be taken when dealing with the numerical schemes for the eigenvalue problem solution. It can be seen in the elements' matrices a particular structure that implements the boundary conditions at the external surfaces. This structure does not appear in the volumes' matrices because the boundary conditions appear as flux terms which are summed up over all the surfaces of each cell, so they are masked inside the volumetric discretization of the divergence and gradient operators.

As the two-group neutron diffusion equation with uniform cross sections over a circle subject to null-flux boundary conditions has an analytical solution, it is adequate to compare how the two proposed numerical schemes relate to it. In the studied problem, we ignored upscattering and fast fissions. Then, the analytical effective multiplication factor is

$$k_{\text{eff}} = \frac{\nu\Sigma_{f2} \cdot \Sigma_{s1\to 2}}{\left[\Sigma_{a1} + \Sigma_{s1\to 2} + D_1(\nu_0/a)^2\right]\left[\Sigma_{a2} + D_2(\nu_0/a)^2\right]} \quad (7)$$

being, $\nu_0 = 2.4048\ldots$, the smallest root of Bessel's first-kind function of order zero $J_0(r)$.

Figure 7 shows how the two numerical effective multiplication factors compare to the analytical solution given by (7) as a function of the mesh refinement, indicated by the quotient a/ℓ_c between the radius of the circle and the characteristic length of the cell/elements. We can see that the k_{eff} computed by the finite volumes (elements) method is always greater (less) than the analytical solution. Indeed, it can be proven that for a bare one-dimensional slab this is always the case [13]. However, this result does not hold

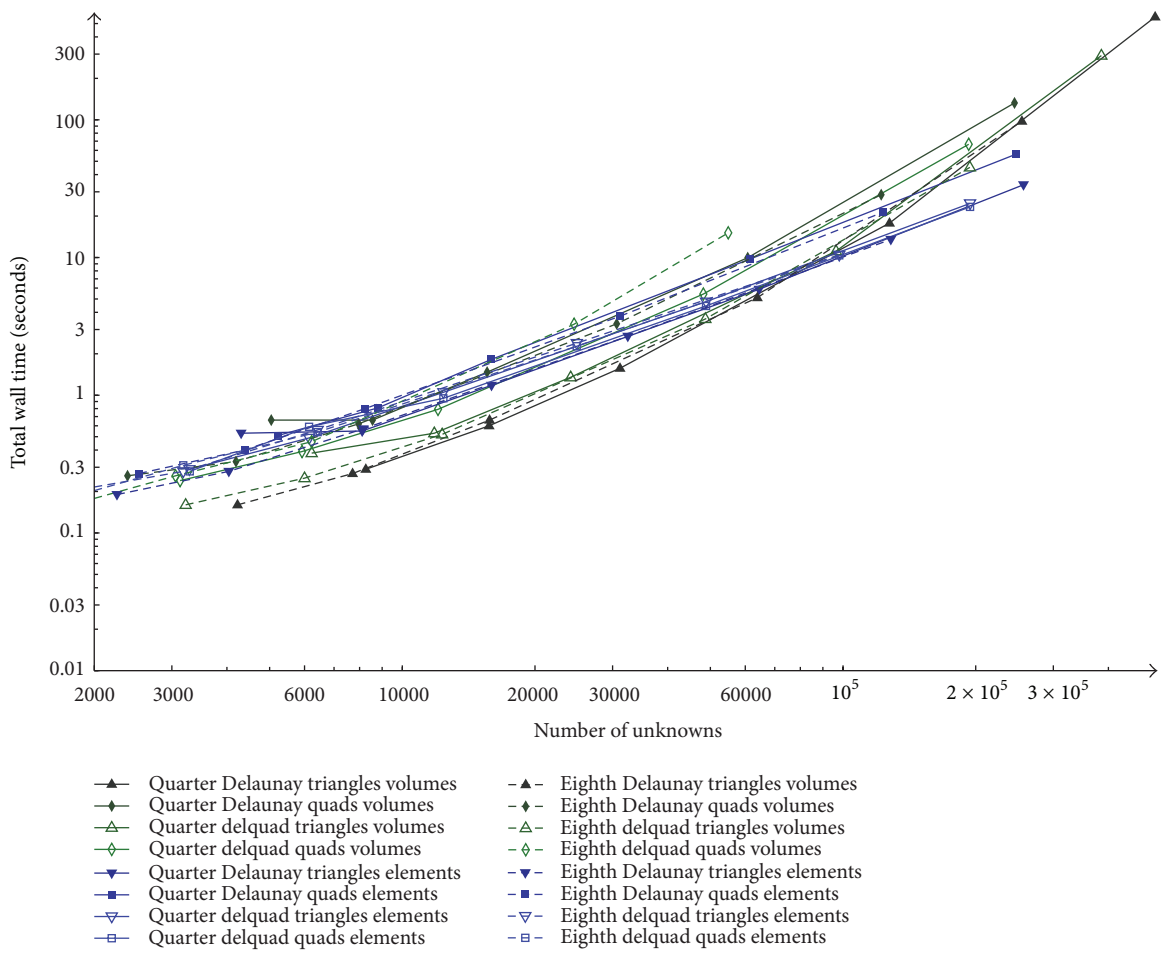


Figure 20: Total wall time versus number of unknowns.

for problems with nonuniform cross sections, even in simple one-dimensional reflected reactors.

We may draw two other conclusions from Figure 7. First, that the finite element method seems to provide a better solution than the volumes-based scheme and, second, that even though the error committed tends to decrease with finer grids, its behavior is not monotonic for finite volumes. In fact, Figure 8 shows the difference between the numerical and analytical solutions using a logarithmic vertical scale, where both conclusions are even more evident. We defer the discussion of such differences between the discretization scheme until the next section. It is worth to note, however, that the fact that a finite-element-based scheme throws better results for the particular bare homogeneous circular reactor under study than the finite volumes does not imply that finite volumes ought to be discarded as a valid tool for solving the neutron diffusion equation in general cases. The combination of lattice and core-level computations is usually performed using cell-based results which when fed into node-based methods to solve the few-group neutron diffusion equation may introduce errors which potentially can lead to unacceptable solutions. Nevertheless, this analysis is far beyond the scope of this paper which focuses on solving a mathematical equation over unstructured grids.

3.2. The 2D IAEA PWR Benchmark Problem. This is a classical two-group neutron diffusion problem, first designed in the early 1970s and taken as a reference benchmark for computational codes. A number of codes were used to solve either this problem or its three-dimensional formulation [22, 23], including milonga using a structured grid [24]. The original formulation can be found in the reference [10], and there is a reproduction that may be easily found online in reference [24]. The geometry consists of a one-quarter of a PWR core depicted in Figure 9, and the homogeneous macroscopic cross sections are listed in Table 1. An axial buckling term $B_{z,g}^2 = 0.8 \times 10^{-4}$ should be taken into account. The external surface should be subject to a zero incoming current condition, which may be written as

$$\frac{\partial \phi_g}{\partial n} = -\frac{0.4692}{D_g} \cdot \phi_g. \tag{8}$$

The expected results are as follows.

(1) Maximum eigenvalue.

(2) Fundamental flux distributions:

(a) Radial flux traverses $\phi_g(x, 0)$ and $\phi_g(x, x)$.

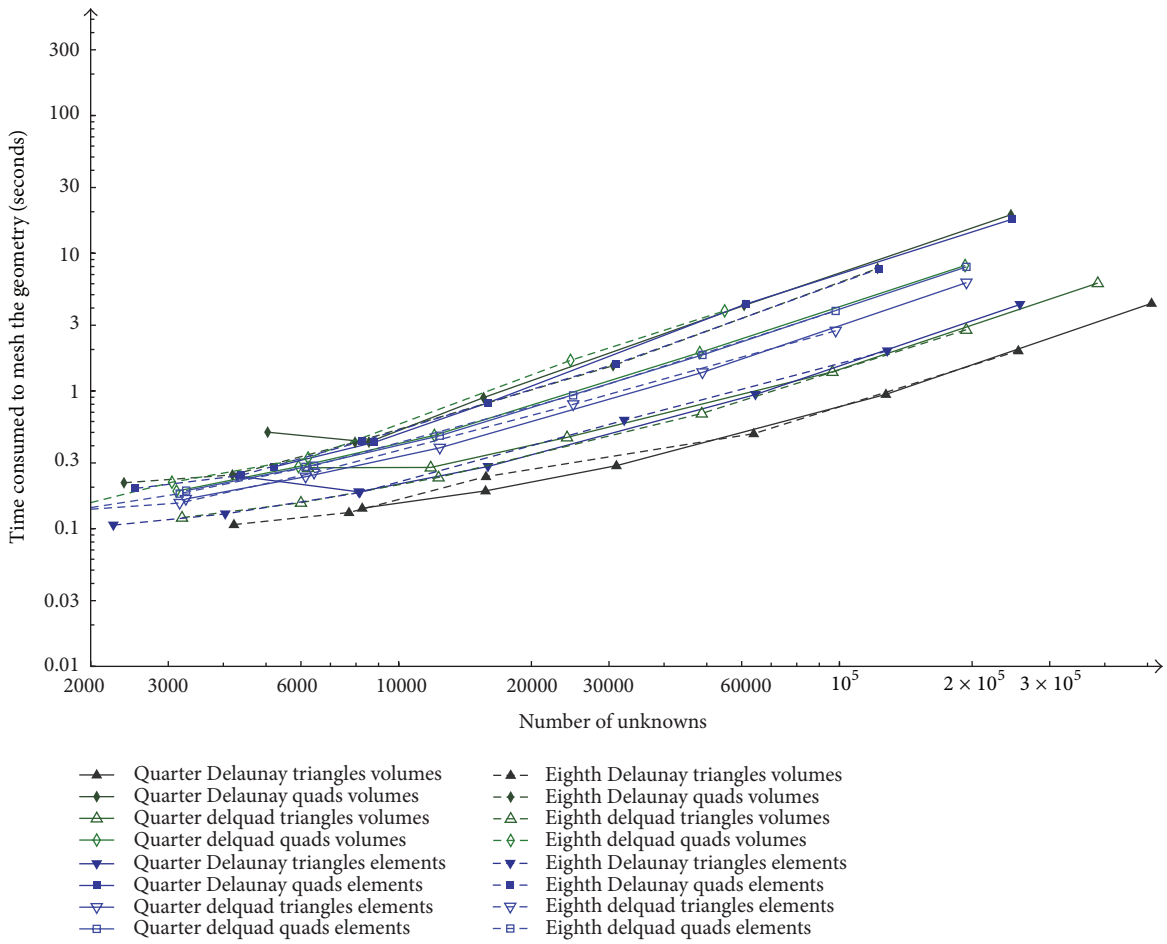


Figure 21: Time needed to mesh the geometry versus number of unknowns.

Table 1: Macroscopic cross-sections (units are not stated in the original reference, but they are assumed to be in cm^{-1} or cm as appropriate).

	D_1	D_2	$\Sigma_{1\to 2}$	Σ_{a1}	Σ_{a2}	$\nu\Sigma_{f2}$	Material
1	1.5	0.4	0.02	0.01	0.080	0.135	Fuel 1
2	1.5	0.4	0.02	0.01	0.085	0.135	Fuel 2
3	1.5	0.4	0.02	0.01	0.130	0.135	Fuel 2 + rod
4	2.0	0.3	0.04	0	0.010	0	Reflector

Note: the fluxes shall be normalized such that

$$\frac{1}{V_{\text{core}}}\int_{V_{\text{core}}}\sum_g \nu\Sigma_{fg}\cdot\phi_g dV = 1. \tag{9}$$

(b) Value and location of maximum power density. This corresponds to maximum of ϕ_2 in the core. It is recommended that the maximum values of ϕ_2 both in the inner core and at the core/reflector interface be given.

(3) Average subassembly powers P_k

$$P_k = \frac{1}{V_k}\int_{V_k}\sum_g \nu\Sigma_{fg}\cdot\phi_g dV, \tag{10}$$

where V_k is the volume of the kth subassembly and k designates the fuel subassemblies as shown in Figure 9.

(4) Number of unknowns in the problem, number of iterations, and total and outer.

(5) Total computing time, iteration time, IO-time, and computer used.

(6) Type and numerical values of convergence criteria.

(7) Table of average group fluxes for a square mesh grid of 20 × 20 cm.

(8) Dependence of results on mesh spacing.

Even though the original problem is based on a quarter-core situation, the problem has an eighth-core symmetry

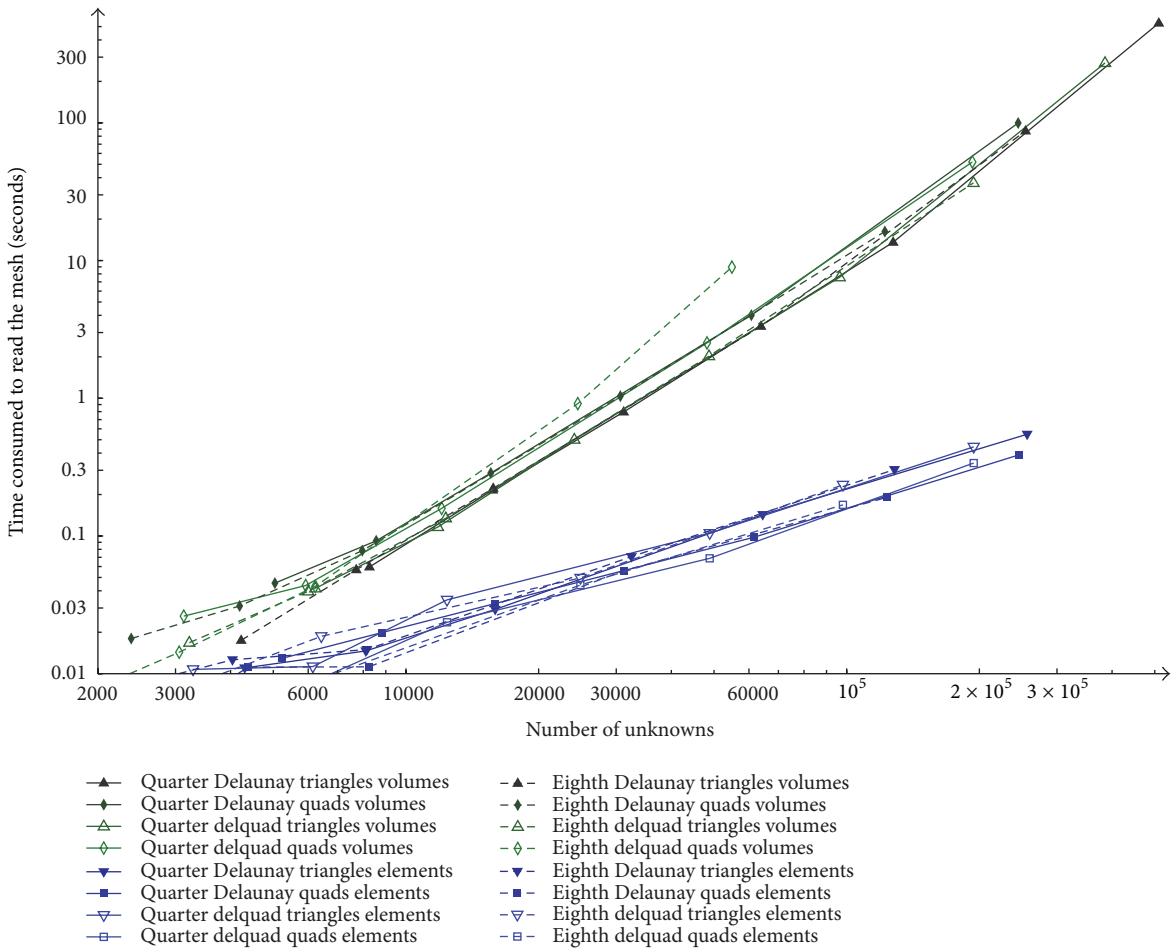


Figure 22: Time needed to read the mesh versus number of unknowns.

which cannot be taken into account by structured grids which are the main target of the benchmark. However, nonstructured grids can take into consideration any kind of symmetry almost without loss of accuracy and at the same time reducing roughly the number of unknowns by a half and the associated computational effort needed to solve the problem by a factor of four. Answers to items (1)–(7) can be given completely by milonga using a single input file (see Supplementary data in Supplementary Material available online at http://dx.doi.org/10.1155/2013/641863). As asked in item (8), how results depend not only on the mesh spacing but also on the meshing algorithm, on the grid's elementary geometric shape, and on the discretization scheme may shed lights on the subject which may be even more interesting than the numerical results themselves.

Taking advantage of milonga's capability of reading and parsing command-line arguments, the selection of the core geometry (quarter or eighth), the meshing algorithm (delaunay [16] or delquad [25]), the shape of the elementary entities (triangles or quadrangles), the discretization scheme (volumes or elements), and the characteristic length ℓ_c of the mesh can be provided at run time. Fixing five values for $\ell_c = 4, 3, 2, 1, 0.5$ gives $2 \times 2 \times 2 \times 2 \times 5 = 80$ possible combinations, which we solve with successive invocations to milonga with the same input file but with different arguments from a simple script. Figures 10, 11, 12, 13, 14, 15, 16, 17, and 18 show the results corresponding to items (1)–(7) for some illustrative cases. The complete set of figures and the code used to generate them may be provided upon request. Table 2 compiles the answers to the problem for every case studied.

As the milonga code is still under development, its numerical routines are not yet fully optimized nor designed for parallel computation. Therefore, the reported times are only rough estimates and should be taken with care. The solution comprises five steps:

(1) generate the grid with the requested geometry, meshing algorithm, basic shape, and characteristic length by calling to gmsh;

(2) read the generated mesh;

(3) build the matrices;

(4) solve the eigenvalue problem;

(5) compute the requested results.

The CPU time reported in Figures 10–18 is thus the sum of all of these five steps, but not the time needed to generate the figures—which in some cases with coarse meshes

Table 2: Results of the 2D IAEA PWR Benchmark problem obtained with the milonga code for the eighty proposed combinations of symmetry, meshing algorithm, basic shape, discretization scheme, and characteristic element/cell length. The reference solution is $\rho = 2874.9 \times 10^{-5}$, which corresponds to $k_{\text{eff}} = 1.0296$.

Case no.	Symm-etry	Mesh algorithm	Basic shape	Solution method	ℓ_c (cm)	$\Delta\rho$ (pcm)	$\max \phi_2$ (—)	Total unknowns	Outer iter.	Linear iter.	Inner iter.	Residual norm	Relative error	Error estimate	Memory (Mb)	Page faults
1	1/4	Delaunay	△	Volumes	4.0	−12.0	11.45	8264	3	32	1033	6e − 09	3e − 09	1e − 09	34	9470
2	1/4	Delaunay	△	Volumes	3.0	8.5	11.35	15740	3	32	1967	2e − 08	1e − 08	4e − 09	48	13401
3	1/4	Delaunay	△	Volumes	2.0	19.7	11.24	31188	3	32	3898	1e − 08	6e − 09	3e − 09	76	21932
4	1/4	Delaunay	△	Volumes	1.0	15.3	11.23	127476	3	32	15934	7e − 09	3e − 09	3e − 09	273	81886
5	1/4	Delaunay	△	Volumes	0.5	18.7	11.19	510676	3	32	63834	2e − 09	9e − 10	1e − 09	1148	352261
6	1/4	Delaunay	△	Elements	4.0	17.3	11.00	4308	3	32	538	2e − 08	8e − 09	4e − 09	31	8794
7	1/4	Delaunay	△	Elements	3.0	11.7	11.07	8108	3	32	1013	5e − 09	2e − 09	2e − 09	47	12946
8	1/4	Delaunay	△	Elements	2.0	8.4	11.12	15936	3	32	1992	6e − 09	3e − 09	2e − 09	73	21347
9	1/4	Delaunay	△	Elements	1.0	6.2	11.16	64478	3	32	8059	6e − 09	3e − 09	4e − 09	262	77105
10	1/4	Delaunay	△	Elements	0.5	5.8	11.17	256700	3	32	32087	1e − 09	5e − 10	1e − 09	1091	331969
11	1/4	Delaunay	□	Volumes	4.0	5.3	11.49	5042	3	32	630	2e − 08	1e − 08	5e − 09	28	8007
12	1/4	Delaunay	□	Volumes	3.0	34.6	11.33	8560	3	32	1070	7e − 09	3e − 09	2e − 09	39	10895
13	1/4	Delaunay	□	Volumes	2.0	30.8	11.31	15576	3	32	1947	1e − 08	7e − 09	3e − 09	54	15645
14	1/4	Delaunay	□	Volumes	1.0	17.7	11.22	60774	3	32	7596	8e − 09	4e − 09	3e − 09	167	50115
15	1/4	Delaunay	□	Volumes	0.5	19.9	11.20	244936	3	32	30617	4e − 09	2e − 09	2e − 09	698	215678
16	1/4	Delaunay	□	Elements	4.0	19.6	11.00	5222	3	32	652	3e − 08	1e − 08	8e − 09	47	13098
17	1/4	Delaunay	□	Elements	3.0	12.3	11.07	8810	3	32	1101	2e − 08	9e − 09	7e − 09	69	18598
18	1/4	Delaunay	□	Elements	2.0	9.0	11.12	15922	3	32	1990	1e − 08	5e − 09	5e − 09	107	30591
19	1/4	Delaunay	□	Elements	1.0	6.3	11.16	61456	3	32	7682	5e − 09	2e − 09	4e − 09	389	113237
20	1/4	Delaunay	□	Elements	0.5	5.8	11.17	246332	3	32	30791	8e − 10	4e − 10	1e − 09	1676	498192
21	1/4	Delquad	△	Volumes	4.0	−4.5	11.44	6228	3	32	778	4e − 08	2e − 08	7e − 09	30	8495
22	1/4	Delquad	△	Volumes	3.0	57.0	11.53	11820	3	32	1477	3e − 08	1e − 08	4e − 09	40	10879
23	1/4	Delquad	△	Volumes	2.0	45.9	11.03	24100	3	32	3012	2e − 08	1e − 08	5e − 09	62	17715
24	1/4	Delquad	△	Volumes	1.0	51.2	11.04	96400	3	32	12050	2e − 08	1e − 08	9e − 09	207	61714
25	1/4	Delquad	△	Volumes	0.5	52.8	11.06	385600	3	32	48200	2e − 09	1e − 09	2e − 09	838	255735
26	1/4	Delquad	△	Elements	4.0	22.0	10.93	3288	3	32	411	3e − 08	2e − 08	6e − 09	30	8495
27	1/4	Delquad	△	Elements	3.0	14.0	11.04	6148	3	32	768	4e − 08	2e − 08	8e − 09	39	11000
28	1/4	Delquad	△	Elements	2.0	8.2	11.10	12392	3	32	1549	2e − 08	9e − 09	6e − 09	61	17067
29	1/4	Delquad	△	Elements	1.0	6.3	11.15	48882	3	32	6110	2e − 09	1e − 09	1e − 09	197	58046
30	1/4	Delquad	△	Elements	0.5	5.8	11.16	194162	3	32	24270	2e − 09	9e − 10	2e − 09	790	238769
31	1/4	Delquad	□	Volumes	4.0	−41.6	11.74	3132	3	32	391	5e − 08	3e − 08	8e − 09	26	7322
32	1/4	Delquad	□	Volumes	3.0	−24.8	11.51	5922	3	32	740	9e − 09	4e − 09	2e − 09	31	8779
33	1/4	Delquad	□	Volumes	2.0	−13.2	11.39	12050	3	32	1506	1e − 08	7e − 09	4e − 09	44	12239
34	1/4	Delquad	□	Volumes	1.0	−4.9	11.26	48200	3	32	6025	6e − 09	3e − 09	2e − 09	122	36094
35	1/4	Delquad	□	Volumes	0.5	−2.3	11.23	192800	3	32	24100	5e − 09	2e − 09	3e − 09	464	140228
36	1/4	Delquad	□	Elements	4.0	22.4	10.93	3294	3	32	411	3e − 08	1e − 08	7e − 09	35	9852
37	1/4	Delquad	□	Elements	3.0	14.1	11.04	6144	3	32	768	3e − 08	2e − 08	9e − 09	51	14018
38	1/4	Delquad	□	Elements	2.0	9.2	11.11	12392	3	32	1549	1e − 08	6e − 09	5e − 09	81	22526
39	1/4	Delquad	□	Elements	1.0	6.5	11.15	48882	3	32	6110	5e − 09	3e − 09	4e − 09	276	78431

Table 2: Continued.

Case no.	Symm-etry	Mesh algorithm	Basic shape	Solution method	ℓ_c (cm)	$\Delta\rho$ (pcm)	$\max\phi_2$ (—)	Total unknowns	Outer iter.	Linear iter.	Inner iter.	Residual norm	Relative error	Error estimate	Memory (Mb)	Page faults
40	1/4	Delquad	□	Elements	0.5	5.9	11.17	194162	3	32	24270	$1e-09$	$5e-10$	$1e-09$	1104	318967
41	1/8	Delaunay	△	Volumes	4.0	−9.9	11.39	4228	3	32	528	$4e-12$	$2e-12$	$1e-12$	35	9630
42	1/8	Delaunay	△	Volumes	3.0	5.0	11.32	7712	3	32	964	$1e-11$	$6e-12$	$2e-12$	42	11578
43	1/8	Delaunay	△	Volumes	2.0	17.7	11.20	15776	3	32	1972	$2e-12$	$1e-12$	$7e-13$	55	15436
44	1/8	Delaunay	△	Volumes	1.0	14.2	11.18	63902	3	32	7987	$3e-12$	$1e-12$	$1e-12$	160	46691
45	1/8	Delaunay	△	Volumes	0.5	17.6	11.15	254612	3	32	31826	$2e-12$	$8e-13$	$1e-12$	555	168526
46	1/8	Delaunay	△	Elements	4.0	18.3	10.96	2252	3	32	281	$6e-12$	$3e-12$	$2e-12$	35	9589
47	1/8	Delaunay	△	Elements	3.0	11.9	11.04	4040	2	24	505	$2e-08$	$1e-08$	$7e-09$	41	11437
48	1/8	Delaunay	△	Elements	2.0	8.5	11.08	8156	3	32	1019	$2e-12$	$8e-13$	$6e-13$	55	15053
49	1/8	Delaunay	△	Elements	1.0	6.3	11.12	32480	3	32	4060	$1e-12$	$6e-13$	$8e-13$	151	43671
50	1/8	Delaunay	△	Elements	0.5	5.8	11.13	128358	3	32	16044	$7e-13$	$3e-13$	$7e-13$	527	157581
51	1/8	Delaunay	□	Volumes	4.0	3.7	11.44	2380	2	24	297	$5e-08$	$2e-08$	$1e-08$	53	14173
52	1/8	Delaunay	□	Volumes	3.0	21.8	11.20	4194	3	32	524	$7e-12$	$4e-12$	$2e-12$	36	10034
53	1/8	Delaunay	□	Volumes	2.0	34.5	11.10	7960	3	32	995	$5e-12$	$3e-12$	$1e-12$	47	12879
54	1/8	Delaunay	□	Volumes	1.0	16.5	11.17	30652	3	32	3831	$2e-12$	$1e-12$	$7e-13$	100	28851
55	1/8	Delaunay	□	Volumes	0.5	19.1	11.17	122066	2	24	15258	$2e-08$	$8e-09$	$9e-09$	343	103825
56	1/8	Delaunay	□	Elements	4.0	17.5	10.97	2526	3	32	315	$4e-12$	$2e-12$	$1e-12$	60	16159
57	1/8	Delaunay	□	Elements	3.0	11.4	11.04	4386	3	32	548	$4e-12$	$2e-12$	$2e-12$	50	13768
58	1/8	Delaunay	□	Elements	2.0	9.0	11.08	8234	3	32	1029	$2e-12$	$1e-12$	$1e-12$	73	20094
59	1/8	Delaunay	□	Elements	1.0	6.2	11.12	31186	2	24	3898	$2e-08$	$7e-09$	$9e-09$	208	59589
60	1/8	Delaunay	□	Elements	0.5	5.8	11.13	123122	2	24	15390	$9e-09$	$5e-09$	$1e-08$	807	236670
61	1/8	Delquad	△	Volumes	4.0	−21.0	11.68	3224	3	32	403	$1e-11$	$5e-12$	$2e-12$	43	11784
62	1/8	Delquad	△	Volumes	3.0	36.9	11.50	6000	3	32	750	$1e-11$	$6e-12$	$3e-12$	50	13502
63	1/8	Delquad	△	Volumes	2.0	31.2	11.04	12316	3	32	1539	$2e-11$	$8e-12$	$5e-12$	60	16395
64	1/8	Delquad	△	Volumes	1.0	44.0	10.94	48714	3	32	6089	$9e-12$	$4e-12$	$4e-12$	118	34441
65	1/8	Delquad	△	Volumes	0.5	53.4	10.81	193980	3	32	24247	$5e-12$	$2e-12$	$4e-12$	423	126768
66	1/8	Delquad	△	Elements	4.0	24.0	10.90	1750	3	32	218	$9e-12$	$4e-12$	$2e-12$	43	11786
67	1/8	Delquad	△	Elements	3.0	15.0	11.00	3184	3	32	398	$1e-11$	$5e-12$	$2e-12$	50	13509
68	1/8	Delquad	△	Elements	2.0	8.9	11.06	6426	3	32	803	$5e-12$	$3e-12$	$2e-12$	59	16047
69	1/8	Delquad	△	Elements	1.0	6.4	11.11	24886	3	32	3110	$2e-12$	$1e-12$	$1e-12$	113	32788
70	1/8	Delquad	△	Elements	0.5	5.9	11.13	98052	2	24	12256	$1e-08$	$5e-09$	$8e-09$	398	118153
71	1/8	Delquad	□	Volumes	4.0	−39.8	11.72	1650	3	32	206	$1e-11$	$5e-12$	$2e-12$	36	9907
72	1/8	Delquad	□	Volumes	3.0	−24.9	11.46	3058	3	32	382	$4e-12$	$2e-12$	$1e-12$	34	9349
73	1/8	Delquad	□	Volumes	2.0	−13.1	11.36	6228	2	24	778	$3e-08$	$1e-08$	$5e-09$	40	10930
74	1/8	Delquad	□	Volumes	1.0	−5.1	11.23	24536	2	24	3067	$3e-08$	$2e-08$	$1e-08$	78	22716
76	1/8	Delquad	□	Elements	4.0	23.7	10.90	1752	3	32	219	$6e-12$	$3e-12$	$2e-12$	41	11213
77	1/8	Delquad	□	Elements	3.0	14.5	11.01	3184	3	32	398	$6e-12$	$3e-12$	$2e-12$	43	11811
78	1/8	Delquad	□	Elements	2.0	9.4	11.07	6426	3	32	803	$3e-12$	$1e-12$	$1e-12$	60	16220
79	1/8	Delquad	□	Elements	1.0	6.5	11.11	24874	3	32	3109	$1e-12$	$6e-13$	$8e-13$	156	44000
80	1/8	Delquad	□	Elements	0.5	5.9	11.13	98042	2	24	12255	$8e-09$	$4e-09$	$8e-09$	567	162168

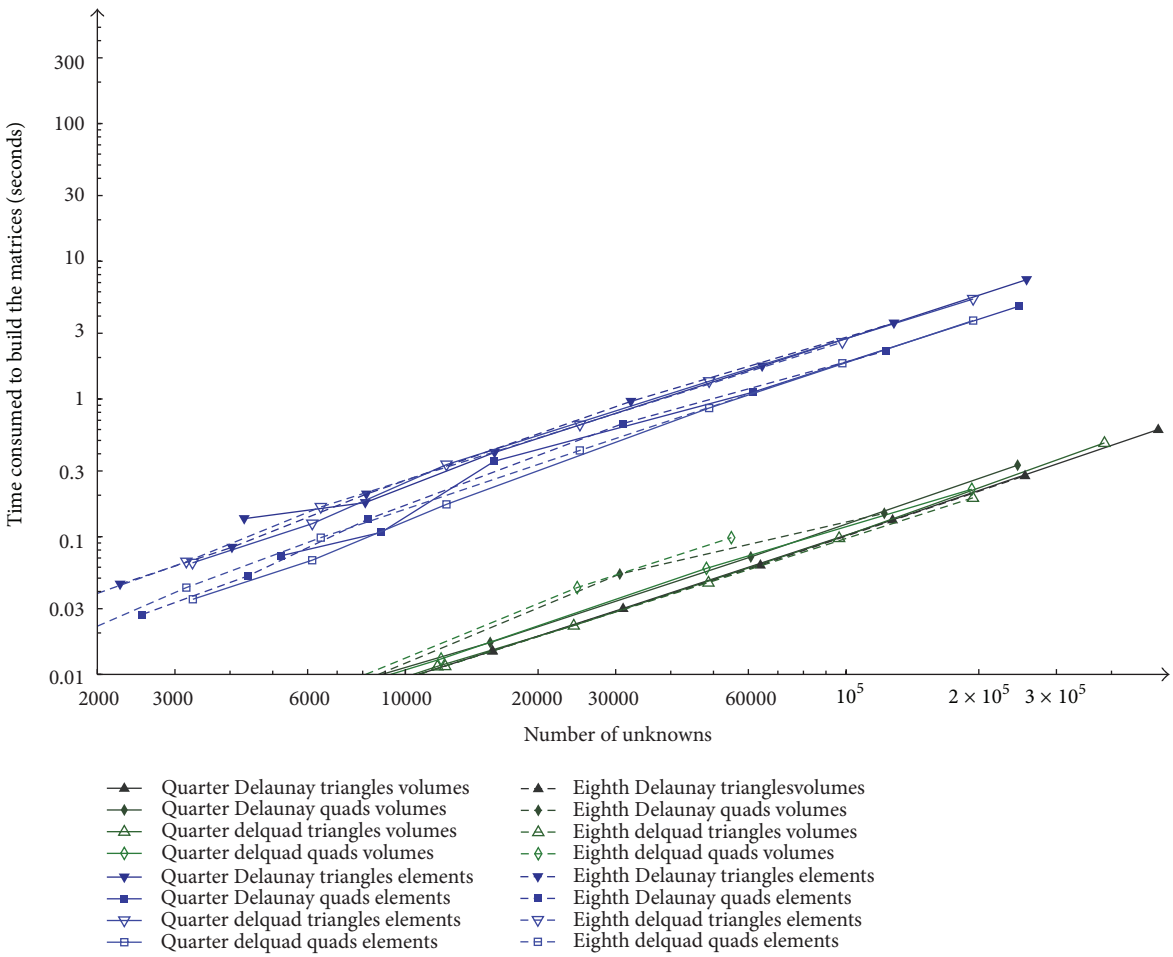


Figure 23: Time needed to build the matrices versus number of unknowns.

was considerably larger than the solution time itself. The eigenvalue problem was solved using a multilayer iterative Krylov-Schur method [26] with a tolerance relative to the matrices norm

$$\frac{\left\|R \cdot \phi - \left(1/k_{\text{eff}}\right) \cdot F \cdot \phi\right\|}{\|R\| + \left(1/k_{\text{eff}}\right) \cdot \|F\|} < 10^{-8}. \tag{11}$$

The reported residual norm and relative error are

$$\left\|R \cdot \phi - \frac{1}{k_{\text{eff}}} \cdot F \cdot \phi\right\|, \qquad \frac{\left\|R \cdot \phi - \left(1/k_{\text{eff}}\right) \cdot F \cdot \phi\right\|}{\left\|\left(1/k_{\text{eff}}\right) \cdot \phi\right\|}, \tag{12}$$

respectively. The fields marked as outer, linear, and inner iterations refer to the number of steps needed to attain the requested tolerance in each layer of the Krylov-Schur algorithm. The computer used to solve the problem has an Intel i7 920 @ 2.67 GHz processor with 4 Gb of RAM running Debian GNU/Linux Wheezy.

When using a finite volumes-based scheme over an unstructured mesh, the solver has to gather information about which cells are neighbors and which are not. Currently gmsh does not write this kind of lists in its output files, so milonga has to explicitly solve the neighbors problem. Performing a linear search is an $O(N^2)$ task, which is unacceptable for problem sizes N of interest. The code uses a search based on a k-dimensional tree [27], which can in principle be performed in $O(N)$ steps. Still, for large values of N, the time needed to read and parse the mesh (number two previously mentioned) is the bottleneck of the solution. This step is not needed in finite elements, although the construction of the elementary matrices involves the computation of the multidimensional Jacobians and integrals, which then have to be assembled. Again, for large N, this step (number three) takes up most of the time needed to solve the problem.

The Delaunay algorithm is a standard method for generating two-dimensional grids [16] by tessellating a plane with triangles. If the mesh needs to be based on quadrangles instead, a recombination algorithm can be used to transform two adjacent triangles in one quadrangle, whenever is possible. However, for geometries which are based on rectangular shapes there exist other algorithms [25] both for meshing and for recombining the triangles that give rise to elementary entities with right angles almost everywhere, which may be a

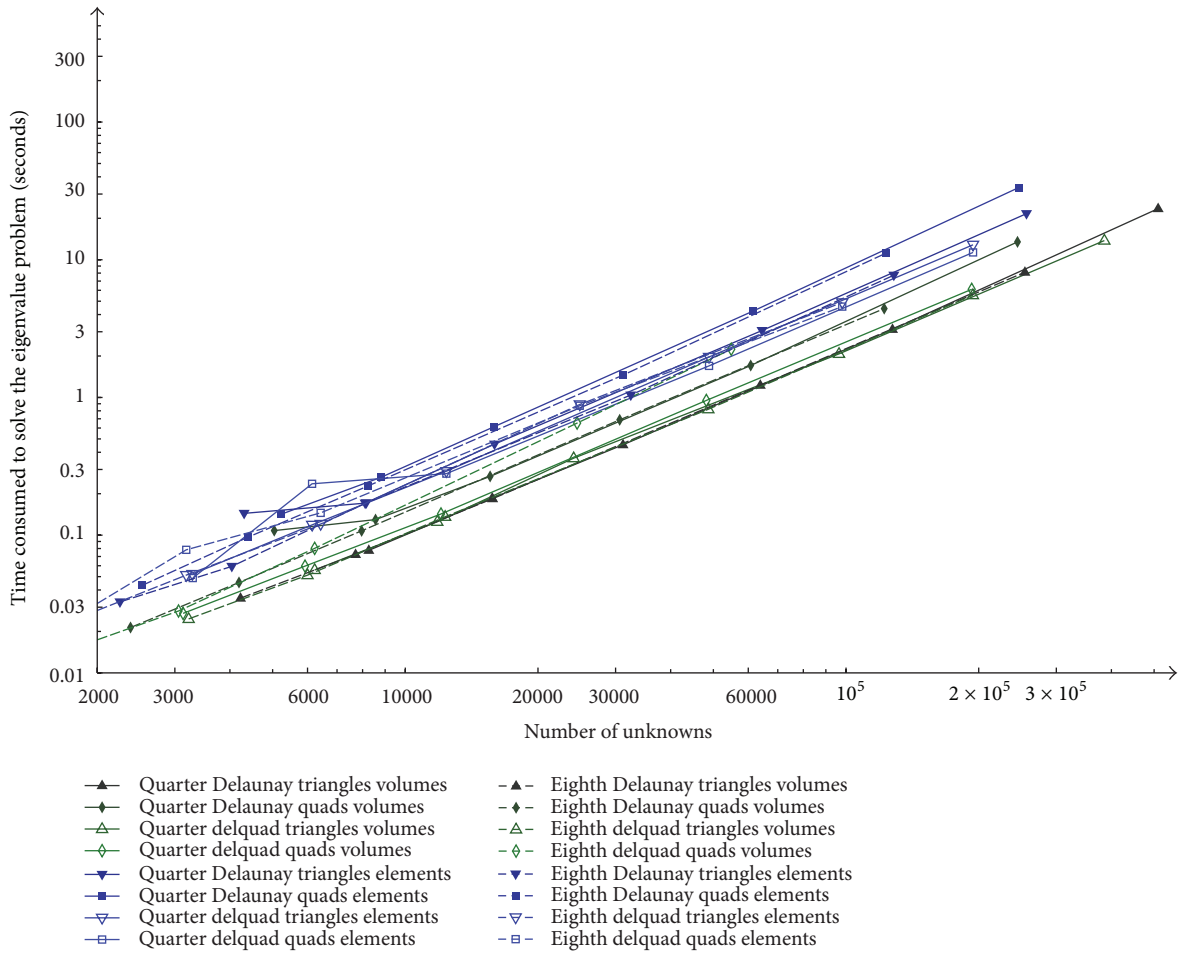


Figure 24: Time needed to solve the eigenvalue problem versus number of unknowns.

desired property of the resulting grid. For finite elements, the steps needed to build the matrices depend on the selection of triangles or quadrangles as the basic elementary geometry because the shape functions change. However, the number of unknowns is the same as the number of nodes that does not change after a recombination procedure. On the other hand, the number of unknowns in the finite volumes schemes with triangles is roughly twice as the number of unknowns with quadrangles for the same grid.

Figure 19 shows how the computed static reactivity (i.e., $1 - 1/k_{\text{eff}}$) depends on the number of unknowns for each of the sixteen combinations of geometry algorithm shape scheme. The four accepted results published in the original reference [10] are included for reference, although it should be taken into account that said reactivities were computed almost forty years ago. Figure 20 shows the total wall time needed to solve the problem as a function of NG, while Figures 21, 22, 23, and 24 show the times needed for each of the first four steps involved in the solution, maintaining the same logarithmic scale for both the abscissas and the ordinates. Green data represent finite volumes, whilst blue bullets correspond to finite elements. Solid lines indicate quarter-core and dashed lines eighth-symmetry. Fillet bullets are results obtained by the Delaunay triangulation, and empty bullets were computed with the delquad algorithm. Finally, triangle-shaped data correspond to triangles and squares, and diamonds denote quadrangles as the basic geometry of the grid.

It can be seen that finite elements produce a much smaller dispersion of eigenvalues k_{eff} than finite volumes with the refinement of the mesh. This can be explained because finite volumes methods rely on a geometric condition of the mesh which may change abruptly if the meshing algorithm decides to allocate the cells in a rather different form for small changes in ℓ_c. Finite elements methods are less influenced by these discontinuous lay-out changes of the elements. As expected, eighth-core symmetries give almost the same results as the quarter-core geometries with roughly half the unknowns. For small problems, finite volumes run faster that finite elements because the time needed to solve the neighbor problem is negligible. When the problem size grows, this time increases and exceeds the overhead implied in the construction and assembly of the finite elements matrices. Also, at least for this configuration, it is seen that the eigenvalue problem is solved faster for finite volumes than for finite elements.

4. Conclusions

Unstructured grids provide the cognizant engineer with a wide variety of possibilities to deal with the design or analysis of nuclear reactor cores. These kinds of grids can successfully reproduce continuous geometries commonly found in reactor cores such as cylinders, and therefore, not only can the diffusion equation be better approximated inside the domain of definition but also the fulfillment of boundary conditions is improved. A free computer code was written from scratch that is able to completely solve the 2D IAEA PWR Benchmark using unstructured grids for sixteen combinations of geometry, meshing algorithm, basic shape, and discretization scheme plus any value of the grid's characteristic length by using a single input file. The complete set of input files and code—executable and source—is available either online or upon request, with comments, experiences, suggestions, and corrections being more than welcome. Further development should include tackling full three-dimensional geometries with complete thermal hydraulic feedback in order to analyze how the solutions of the coupled neutronic-thermal problem depend on the spatial discretization scheme of the neutron leakage term. Parallelization of the computation and assembly of the matrices and of the solution of the eigenvalue problem and its implementation using GPUs are also desired features to implement. A problem with direct applications that the future versions of milonga ought to solve is the analysis of how the geometry of the absorbing materials should be taken into account in structured coarse grids in order to mitigate effects such as the rod-cusp problem.

Suitable schemes for approximating the continuous differential operators by discrete matrix expressions include finite volumes and finite elements families. Finite volumes methods compute cell mean values, whilst finite elements give functional values at the grid's nodes. In general, finite elements are less sensitive to changes in the mesh so the results they provide do not change significantly for different meshing algorithms or elementary shapes. Small problems are best solved by finite volumes as the neighbor-finding problem is faster than the process of building and assembling the eigenvalue-problem matrices. For a large number of unknowns, the process of finding which cell is neighbor of which—even based on a k-dimensional tree—overwhelms the computation and assembly of elementary matrices, and finite-element methods perform better.

References

[1] G. Glasstone and S. Bell, *Nuclear Reactor Theory*, Krieger Publishing Company, 1970.

[2] J. J. Duderstadt and L. J. Hamilton, *Nuclear Reactor Analysis*, John Wiley & Sons, New York, NY, USA, 1976.

[3] C. Gho, *Reactor Physics Undergraduate Course Lecture Notes*, Instituto Balseiro, 2006.

[4] G. G. Theler and F. J. Bonetto, "On the stability of the point reactor kinetics equations," *Nuclear Engineering and Design*, vol. 240, no. 6, pp. 1443–1449, 2010.

[5] E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, John Wiley & Sons, 1984.

[6] E. Masiello, "Analytical stability analysis of Coarse-Mesh finite difference method," in *Proceedings of the International Conference on the Physics of Reactors (PHYSOR '08)*, Nuclear Power: A Sustainable Resource, pp. 456–464, September 2008.

[7] M. L. Zerkle, *Development of a polynomial nodal method with flux and current discontinuity factors [Ph.D. thesis]*, Massachusets Institute of Technology, 1992.

[8] J. I. Yoon and H. G. Joo, "Two-level coarse mesh finite difference formulation with multigroup source expansion nodal kernels," *Journal of Nuclear Science and Technology*, vol. 45, no. 7, pp. 668–682, 2008.

[9] O. Mazzantini, M. Schivo, J. Di Césare, R. Garbero, M. Rivero, and G. Theler, "A coupled calculation suite for Atucha II operational transients analysis," *Science and Technology of Nuclear Installations*, vol. 2011, Article ID 785304, 12 pages, 2011.

[10] Computational Benchmark Problem Comitee for the Mathematics and Computation Division of the American Nuclear Society, "Argonne Code Center: Benchmark problem book," Tech. Rep. ANL-7416, Supplement 2, Argonne National Laboratory, 1977.

[11] A. F. Henry, *Nuclear Reactor Analysis.*, MIT, Cambridge, UK, 1975.

[12] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, vol. 1, Elsevier, 6th edition, 2005.

[13] G. Theler, *Difusión de neutrones en mallas no estructuradas: comparación entre volúmenes y elementos finitos*, Academic Monograph, Universidad de Buenos Aires, 2013.

[14] G. Theler, "Milonga: a free nuclear reactor core analysis code," Available online, 2011.

[15] G. Theler, F. J. Bonetto, and A. Clausse, "Optimización de parametros en reactores de potencia: base de diseño del codigo neutrónico milonga," in *Reunion Anual de la Asociacion Argentina de Tecnologia Nuclear, XXXVII*, 2010.

[16] C. Geuzaine and J.-F. Remacle, "Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[17] S. Balay, J. Brown, K. Buschelman et al., "PETSc users manual," Tech. Rep. ANL-95/11-Revision 3.4, Argonne National Laboratory, 2013.

[18] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, Eds., pp. 163–202, Birkhäuser, 1997.

[19] V. Hernandez, J. E. Roman, and V. Vidal, "SLEPC: a scalable and flexible toolkit for the solution of eigenvalue problems," *ACM Transactions on Mathematical Software*, vol. 31, no. 3, pp. 351–362, 2005.

[20] M. Galassi, J. Davies, J. Theiler et al., *GNU Scientific Library Reference Manual*, 3rd edition, 2009.

[21] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method For Solid and Structural Mechanics*, vol. 3, Elsevier, 6th edition, 2005.

[22] A. Imelda and K. Doddy, Evaluation of the IAEA 3-D PWR benchmark problem using NESTLE code, 2004.

[23] R. Mosteller, "Static benchmarking of the NESTLE advanced nodal code," in *Proceedings of the Joint International Conference*

on Mathematical Methods and Supercomputing for Nuclear Applications, vol. 2, pp. 1596–1605, 1997.

[24] G. Theler, F. J. Bonetto, and A. Clausse, "Solution of the 2D IAEA PWR Benchmark with the neutronic code milonga," in *Actas de la Reunión Anual de la Asociación Argentina de Tecnología Nuclear, XXXVIII*, 2011.

[25] J. F. Remacle, F. Henrotte, T. Carrier-Baudouin et al., "A frontal delaunay quad mesh generator using the L^{∞} norm," *International Journal For Numerical Methods in Engineering*, vol. 94, no. 5, pp. 494–512, 2013.

[26] G. W. Stewart, "A Krylov-Schur algorithm for large eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 601–614, 2002.

[27] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.