M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

Reactor physics analysis using Isogeometric Analysis and GeoPDEs

W.F.G. van Rooijen, R. Horita

Research Institute of Nuclear Engineering, University of Fukui, Tsuruga, Japan
rooijen@u-fukui.ac.jp, fp130532@u-fukui.ac.jp

Abstract - *The reactor physical properties of a nuclear reactor depend on the geometry of the reactor, and this geometry in turn depends on the neutron flux, power distribution, etc. With (most) existing reactor physics codes only perfect geometries can be treated. Even the conventional Monte Carlo method, which has sufficient precision to be considered as a numerical experiment, can only treat exact geometries, i.e. perfect cylinders, spheres, etc. To increase the power of reactor analysis, a calculation method which can treat truly arbitrary geometry could be very beneficial. The recently developed Iso-Geometric Analysis (IGA) method provides a general framework to solve PDEs with the Finite Element Method on arbitrary geometry. In the present research the GeoPDEs package was investigated for applications to neutronic calculations in multi-group, heterogeneous diffusion theory and transport theory. Results shown in this manuscript show the powerful features of the IGA method, and in general very good results were obtained. The biggest drawback of GeoPDEs is the calculation time, which is prohibitively long for practical neutronic calculations.*

I. INTRODUCTION

The reactor physical properties of any nuclear reactor are more or less sensitive to the geometrical configuration of the reactor. Especially in fast reactors, the reactivity effect due to changes of the core geometry can be very important. One example is the passive safety behavior of the EBR-II reactor, in which feedback from thermal expansion was so strong that the reactor was capable of safely sustaining an Unprotected Loss Of Flow (ULOF) from full power, as was verified experimentally [1]. Another example is the prototype fast reactor Monju: the pressure head of the primary pumps causes the core support plate to bend, leading to a deformation of the core as a function of the pump flow rate, which has a definite (and quite strong) feedback on the reactivity; a measurement of this effect was succesfully performed in 2010 [2].

It would be very attractive if it were possible, somehow, to calculate the reactivity effects of the geometrical changes directly. Thermal-hydraulics codes can be used to determine the temperature distribution with great detail - in fact, with sub-channel codes or CFD calculations it is nowadays possible to resolve the transient temperature in wire-wrapped fuel bundles directly [1]. Thermo-mechanics codes can be used to determine stress fields and the geometrical deformation. Unfortunately, there exist, at present, no neutronics codes which are capable of treating truly arbitrary geometry. As detailed in the next section, codes are commonly limited to idealized meshes (RZ, XYZ, Hex-Z), while ray-tracing methods are based on ideal surfaces represented by simple mathematical equations.

The ultimate goal of the work described in this manuscript is a method to solve neutron transport in truly arbitrary geometry, focusing on applications to (fast) nuclear reactors. Applications of such a method include direct numerical evaluation of feedback effects due to (small) geometrical changes, e.g. due to thermal expansion or other mechanical causes, and integration of multi-physics analysis, including thermo-mechanical analysis, deformation analysis and direct coupling with neutronic calculations. While this manuscript focuses on nuclear reactors, there are many other fields of particle transport theory where arbitrary shapes are important; medical physics would be a good example of such a field.

In this manuscript it is proposed to use Iso-Geometric Analysis (IGA) as a method to solve PDEs. The most important feature of IGA is that it can, in principle, treat truly arbitrary geometry without any approximations, and the description of the geometrical domain is directly linked to the numerical calcution on the domain. IGA can be categorized as a Finite Element Method [3].

This manuscript is organized as follows: in Section II. some background is given about the treatment of geometry in numerical analysis in reactor phsyics codes. In Section III. an introduction of NURBS (Non-Uniform Rational B-Splines) geometry is given, and in Section IV. the Iso-Geometric Analysis (IGA) is introduced. In Section V. are given the results of some trial calculations using the IGA-method, implemented with the free software package GeoPDEs, and the paper finished with a conclusion and discussion in Section VI..

II. TREATMENT OF GEOMETRY IN NUMERICAL SIMULATIONS

This section focuses on the description of the geometrical domain in neutronics calculations. The objective is to give an indication where the weak points lie in the conventional methods, and how IGA can improve the situation. A distinction is made between, on the one hand, *ray tracing methods*, such as the Method of Characteristics or Monte Carlo (and, in some cases, calculations based on Collision Probability), and on the other hand *mesh-based methods*.

In calculations based on ray tracing, the trajectory of the particle is represented by a straight line:

$$l : \boldsymbol{p} + t\hat{\boldsymbol{e}}, -\infty \leq t \leq \infty$$

where $\boldsymbol{p}$ is an arbitrary point on the trajectory and $\hat{\boldsymbol{e}}$ is the unit vector giving the direction of particle movement. In most codes, two types of surfaces can be represented, namely plane surfaces:

$$p : \hat{\boldsymbol{n}}^T \boldsymbol{r} + d = 0$$

with $\hat{\boldsymbol{n}}$ the column unit vector perpendicular to the surface, $\boldsymbol{r}$ the location vector (i.e. $\boldsymbol{r} = [x, y, z]^T$), and d a measure for the translation of the plane from the origin along $\hat{\boldsymbol{n}}$; and quadratic surfaces:

$$s : \boldsymbol{r}^T \boldsymbol{A} \boldsymbol{r} + \boldsymbol{b}^T \boldsymbol{r} + d = 0$$

where the elements of the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ determine the shape of the surface (cylinder, sphere, etc), d determines the scale of the surface (e.g. radius of the sphere).

Having represented the surfaces as equations, the distance t from the current point $\boldsymbol{p}$ to the point $\boldsymbol{i}$ where a particle crosses the surface is found by solving the roots of a first- or second-order equation. This task can be performed quickly and with good accuracy. The drawback is that surfaces which cannot be represented as planes or quadratic surfaces cannot be used in these ray-tracing codes[1]. Thus, conventional Monte Carlo codes can represent only a limited set of shapes: perfect planes, perfect cylinders and perfect spheres[2].

In *mesh-based methods* the actual geometrical domain is subdivided into (small) parts (call them *nodes*) and approximations are used for derivative operators so that, in general, only neighboring nodes interact with each other. In general, these solution methods yield a matrix-vector solution, where the matrix must be inverted. In the simplest of these methods, the geometrical domain is limited to XYZ, RZ, or Hex-Z. Modeling of any real reactor with such codes necessarily implies some level of a-priori geometrical simplication. In more advanced FEM-based methods, the geometrical domain can be more or less “complex" but the discretization into “elements" still implies an approximation. Most importantly, the FEM mesh in general does not necessarily conserve the actual volumes of the geometrical entities; this is a problem when investigating the effects of geometrical changes. For example, suppose two FEM meshes are created: one mesh is for a reference geometry, and the other mesh corresponds to a case with higher temperature, where the fuel region is slightly larger, and the fuel has a lower density. Since the actual volume of fuel in the FEM mesh is not necessarily conserved, any reactivity difference may be due to modeling error rather than a real physical effect.

There are many fields of engineering where complex geometrical shapes must be represented numerically - fields such as automotive or aerospace engineering would be unimaginable without such flexible descriptions of the geometry! One common method of representing arbitrary surfaces in CAD systems is through so-called NURBS ([4], see Section III.). Stated simply, in NURBS theory physical space represented in a parametric space of simple geometry. For example, a complicated surface in 3D space is represented as a 2D plane in parametric space; denote the variables as (u, v). Technically, it is possible to use such surfaces directly for ray tracing. Formulating the problem of ray-tracing in this way results in a minimization problem to find the coordinates (u_0, v_0) corresponding to the intersection point between the ray and the surface. In earlier research, such a method was investigated but the results were discouraging: the method was slow, inaccurate, and numerically unstable [5].

In recent years, several Monte Carlo codes have been extended to treat arbitrary surfaces with some sort of CAD-based geometry. In most (if not all) cases, the parametric surfaces are not used directly, but tesselation is used: one defines a set of points $(u, v)_i$, then calculates the corresponding vertices $(x, y, z)_i$ and the surface is represented by triangles interpolating the vertices. The main drawbacks of this approach are twofold, i.e. since each triangle represents a part of an infinite plane, the number of planes in the geometry increases considerably, resulting in a longer calculation time for the ray tracing[3]. Another problem is that the tesselation does not necessarily conserve the volume of the underlying body. This may be a problem in cases where one is calculating small differences due to small geometrical changes.

III. A CRASH COURSE IN NURBS THEORY

In the limited space of this manuscript only a very basic overview of the theory behind NURBS (Non-Uniform Rational B-Splines) can be given. The interested reader is referred to the standard work by Piegl & Tiller for more detailed explanations [4]. For brevity, explanations are given here for a one-dimensional case. A NURBS is defined by a *knot-vector* $\boldsymbol{k}$ and a set of *control points* $\boldsymbol{P}$; the control points form the *control polygon*.

The knot vector is a vector defined on a continuous variable u, so that $\boldsymbol{k} = [u_0, u_1, \ldots, u_m]$; each u_k is called a *knot*. Without loss of generality, it is assumed that $u_0 = 0$ and $u_m = 1$. The knots are ordered, i.e. $u_{k+1} \geq u_k$, and the interval $(u_k, u_{k+1}]$ is called a *knot span*. The knots need not be unique. The knot vector determines a set of basis functions of $N_{i,p}(u)$; the degree p of these basis functions, their continuity, and the existence and continuity of the derivatives is determined by the multiplicity of the knots. For NURBS, the basis functions $N_{p,i}(u)$ are B-spline curves.

A control point $\boldsymbol{P}_i$ is a point in three-dimensional space, and has associated with it a weight w_i. The set of control points forms the *control polygon*. A NURBS curve is now defined as:

$$\boldsymbol{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{1}$$

Basically, the shape of the curve is determined by the the

[1] To be complete, in MCNP one can also use a toroidal surface, represented by a fourth-order equation. As indicated in the MCNP manual, this surface is challenging, requiring at least double precision arithmatic to have sufficient accuracy when determining the intersections.

[2] In defense of conventional Monte Carlo codes: complex geometries can be represented by collections of rectangles, so-called *voxels* for *volume elements*. This kind of representation is very common in medical physics and linked to calculations based on measured 3D CAT data.

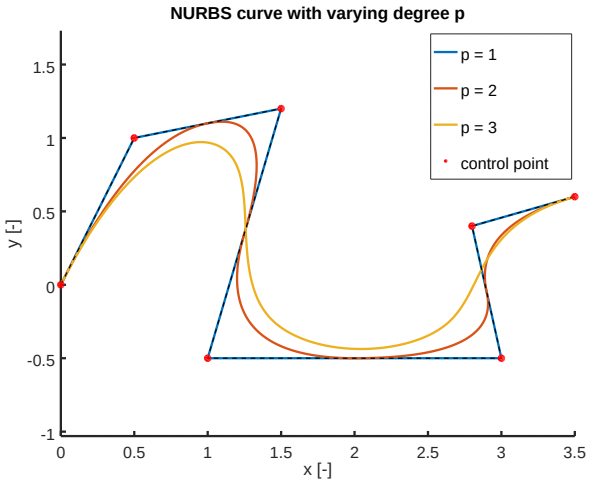
[3] To be complete, it should be noted that ray tracing on tesselated surfaces is fundamental to the field of computer graphics and fast and accurate algorithms are available.

control polygon. The NURBS curve “interpolates" the control points, and the “smoothness" of the curve depends on the degree p of the basis functions and the weights of the control points. If linear basis functions are used, the NURBS curve interpolates the control points exactly, so that the NURBS curve coincides with the control polygon. With increasing degree of the basis functions, the NURBS curve is an increasingly “smoothed" approximation of the control polygon. This is illustrated in Figure 1. The powerful feature of NURBS curves is that the shape is completely arbitrary and can be customized by changing one or more control points. Thus, NURBS curves provide a way to mathematically represent arbitrary geometric shapes, and perform mathematical operations and manipulations on this geometry. Finally, NURBS can exactly represent several common shapes such as straight lines, circular arcs, etc.

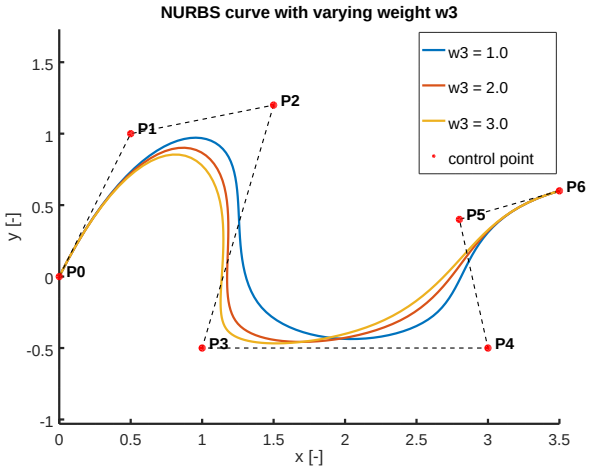
IV. ISO-GEOMETRIC ANALYIS, IGA, AND GEOPDES

The NURBS basis functions $N_{i,p}(u)$ have several properties which make them candidates as trial functions in a Galerkin-style finite element method: partition of unity, finite support domain, continuity, existence and continuity of derivatives, etc. A 3D surface in physical space can be constructed from a tensor surface from two NURBS curves in parametric space; similarly, a 3D volume can be constructed from the tensor volume from three NURBS curves in parametric space. For a surface, call the parametric variables u and v, then there are two knot vectors $\boldsymbol{k}_u$ and $\boldsymbol{k}_v$, a control network having $P \times Q$ control points, and a set of 2D basis functions $N_{i,p}(u)N_{j,q}(v)$. The basic premise of the Iso-Geometric Analysis (IGA) is to solve the PDE on the parametric (square) (u, v) domain, where the knot vectors $\boldsymbol{k}_u$ and $\boldsymbol{k}_v$ define a rectangular mesh. Then one uses the information from the control network to “transform" the parametric solution to the physical domain. One immediately identifies the two most important requirements of the transformation: the transformation from the parametric domain to the physical must exist and be unique; the reverse transform, i.e. from the physical domain to the parametric domain, must exist and be unique, so that boundary conditions and physical properties may be transformed to the parametric domain. The resulting solution method is called *Iso-Geometric Analysis* (IGA, [3]) because the basis functions to describe the physical domain of the problem are also the trial functions in the FEM-based solution. The most powerful aspect of IGA is that the NURBS can represent exactly any arbitrary geometrical shape, in other words, the shape of the physical domain can be truly arbitrary, and still be represented in a mathematically exact way.

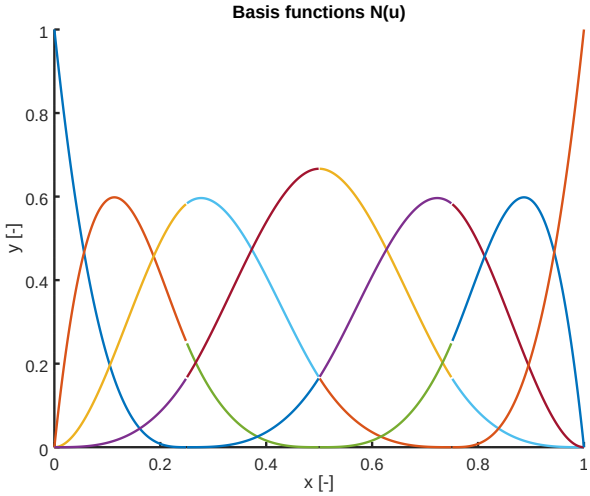
The basic approach of NURBS geometry is on the following (based on the introduction given in the manual of the GeoPDEs software [6]). Consider that one has an *n-dimensional parametric domain* $\hat{\Omega} = (0, 1)^n$, and a *physical domain* $\Omega = \boldsymbol{F}(\hat{\Omega}) \subset \mathbb{R}^r$, of dimension r, $n \leq r$, defined by a *parametrization* $\boldsymbol{F}$. The Jacobian of $\boldsymbol{F}$ is denoted as $J_{\boldsymbol{F}}$ and is an $r \times n$ matrix. Let V be a Hilbert space, to which the continuous solution of the problem belongs, and $V_h \subset V$ a discrete subspace of V in which we seek an approximate solution. Given a bilinear form $a : V \times V \to \mathbb{R}$ and a func-



(a) NURBS curve $\boldsymbol{C}(u)$ for linear, quadratic and cubic basis functions. As the degree p increases the curve becomes smoother.



(b) NURBS curve $\boldsymbol{C}(u)$, cubic basis functions, with a variation of the weight of control point $\boldsymbol{P}_3$. As the weight w_3 increases the curve is more strongly pulled towards $\boldsymbol{P}_3$.



(c) The cubic basis functions $N_{i,p}(u)$ used in this NURBS curve.

Fig. 1: Example of a NURBS curve.

tion $f \in L^2(\Omega)$, the variational formulation of the differential equation becomes: *Find* $u_h \in V_h$ *such that*

$$a(u_h, v_h) = (f, v_h), \quad \forall v_h \in V_h \tag{2}$$

where $(\cdot,\cdot)$ represents the L^2-inner product (i.e., integrals over the problem domain). Based on the assumption of a parametric domain, the approximation domain V_h becomes:

$$V_h : \left\{ v_h = p_{\boldsymbol{F}}(\hat{v}_h), \hat{v}_h \in \hat{V}_h \right\} \tag{3}$$

where $p_{\boldsymbol{F}}$ is a *push-forward* defined from the parametrization $\boldsymbol{F}$. If $\boldsymbol{F}$ is sufficiently regular, then one can define a *basis* $\hat{B} = \left\{ \hat{v}_j, j = 1, \ldots, N_h \right\}$ for the space $\hat{V}_h$ and one can derive a basis for the space V_h by applying the push-forward operator to the $\hat{v}_j$, i.e., $B = \left\{ p_{\boldsymbol{F}}(\hat{v}_j), j = 1, \ldots, N_h \right\}$. Thus, the solutions u_h can be written as:

$$u_h = \sum_{j=1}^{N_h} \alpha_j v_j = \sum_{j=1}^{N_h} \alpha_j p_{\boldsymbol{F}}(\hat{v}_j)$$

If one replaces in Eq. (2) the expansion of u_h, and testing against each basis function $v_i, i = 1, \ldots, N_h$, a linear system of equations follows with the α_j as unknowns and a matrix $A_{ij} = a(v_j, v_i)$ and a right hand side $f_i = (f, v_i)$.

In general the integrals appearing in A_{ij} and f_i cannot calculated exactly and some numerical approximation is used. For the present, the integrals are replaced by quadrature rules. The parametric domain $\hat{\Omega}$ is divided into N_{el} non-overlapping subregions $\hat{\boldsymbol{K}}_k := \left\{ \hat{K}_k \right\}_{k=1}^{N_{\text{el}}}$; the subregions are loosely referred to as *elements*. If the parametrization $\boldsymbol{F}$ is not singular, the image of the elements in the phsyical space is a non-overlapping partition of Ω denoted as $\boldsymbol{K}_k := \{K_k\}_{k=1}^{N_{\text{el}}}$, with $K_k = \boldsymbol{F}(\hat{K}_k)$. Assume that a quadrature rule is defined on each element $\hat{K}_k$, defining on each element a set of n_k *quadrature nodes*:

$$\hat{\boldsymbol{x}}_{l,k} \in \hat{K}_k \quad l = 1, \ldots, n_k, \quad k = 1, \ldots, N_{\text{el}}$$

and *quadrature weights*:

$$\hat{w}_{l,k} \in \mathbb{R}, \quad l = 1, \ldots, n_k, \quad k = 1, \ldots, N_{\text{el}}$$

Thus, an integral of some function $\phi(\boldsymbol{x})$ in the physical domain can be written as:

$$\begin{aligned} \int_{K_k} \phi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} &= \int_{\hat{K}_k} \phi(\boldsymbol{F}(\hat{\boldsymbol{x}})) |J_{\boldsymbol{F}}(\hat{\boldsymbol{x}})| \mathrm{d}\hat{\boldsymbol{x}} \\ &\approx \sum_{l=1}^{n_k} \phi(\boldsymbol{F}(\hat{\boldsymbol{x}}_{l,k})) |J_{\boldsymbol{F}}(\hat{\boldsymbol{x}}_{l,k})| \hat{w}_{l,k} = \sum_{l=1}^{n_k} \phi(\hat{\boldsymbol{x}}_{l,k}) w_{l,k} \end{aligned}$$

where $\boldsymbol{x}_{l,k} = \boldsymbol{F}(\hat{\boldsymbol{x}}_{l,k})$ are the images of the quadrature nodes in the physical domain and $w_{l,k} = \hat{w}_{l,k} J_{\boldsymbol{F}}(\hat{\boldsymbol{x}}_{l,k})$, with $|J_{\boldsymbol{F}}|$ the measure evaluated at the quadrature points. Stated simply: the integrals in the physical domain are transformed to integrals in the parametric domain with a coordinate transformation; the integrals are subsequently approximated by a quadrature rule in the parametric domain; and the quadrature rule in the parametric domain can be interpreted as a quadrature rule in the physical domain. In Figure 2 is given an illustration of the relations between an element in the physical domain Ω_e, the corresponding element in the parametric domain $\hat{\Omega}_e$ and the image of the element in the domain where the Gaussian integration is performed $\tilde{\Omega}_e$.

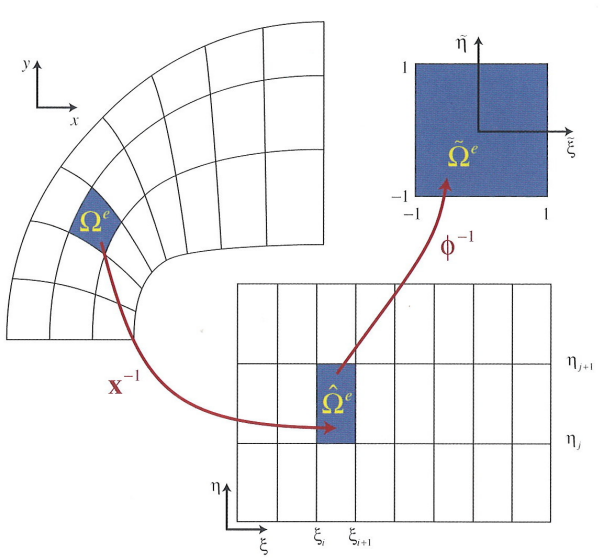


Fig. 2: The relations between an element in the physical domain Ω_e, the corresponding element in the parametric domain $\hat{\Omega}_e$ and the image of the element in the domain where the Gaussian integration is performed $\tilde{\Omega}_e$.

From Figure 2 some observations can be made about the calculation settings for IGA. The knot vectors create a rectangular mesh in parametric space; each square is referred to as an *element*. A first option for refinement is to choose a finer spacing of the knots, increasing the number of elements (in NURBS parlance: *knot refinement*). The second option is *degree elevation* of the basis functions. Increasing the degree of the basis functions has various side effects so care must be taken. Finally, one can increase the number of quadrature nodes for the Gaussian integrations in each element.

Essential differences between IGA and conventional FEM

The difference between conventional FEM and IGA is illustrated in Figure 3. In conventional FEM, the problem domain is first meshed with a number of elements. The number of elements and their basic shape is selected by the user. In IGA, the problem domain is subdivided into so-called patches (this is for convenience only, the theory remains valid if the entire domain is one single patch), and on each patch, the equation is solved directly based on the NURBS description of the geometry. IGA is sometimes grouped as a "mesh-less" FEM. With NURBS-based geometry, the parametrization $\boldsymbol{F}$ can represent any arbitrary, smooth domain.

The IGA method as described in [3] is a conventional Galerkin Finite Element Method, where the trial functions (basis functions) are chosen to be the basis functions of NURBS geometry. There is thus an intimate relation between the de-

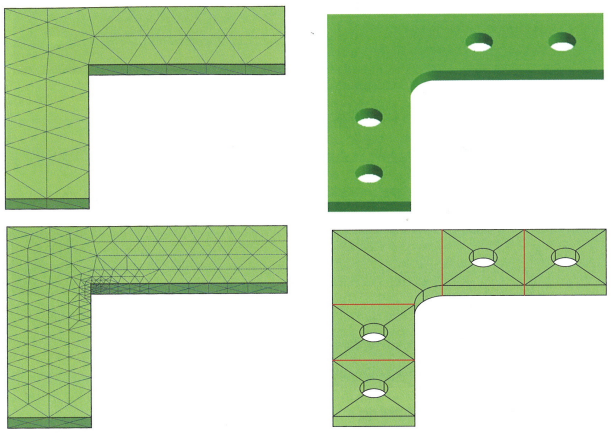


Fig. 3: Conventional FEM meshing versus NURBS-based geometry in IGA. Top right: the physical domain. Top left: a potential FEM-mesh on the (simplified) domain. Bottom left: a refined mesh to capture the stress concentration in the corner section. Bottom right: NURBS-based IGA mesh.

scription of the problem geometry and the trial functions used in the numerical solution. In conventional FEM, one describes the geometry of the problem domain followed by some kind of meshing algorithm to create the FEM mesh. The FEM mesh is thus always *an approximation* of the actual physical domain, even if one uses higher-order (curved) FEM meshes. In IGA the actual physical domain is used for the solution. Another difference is that IGA, due to the nature of the NURBS geometry, has the possibility to use knot refinement and degree elevation to refine the solution space while maintaining the *exact same* physical domain Ω. In conventional FEM meshes, the shape of the meshes is often related to the trial functions, and higher-order trial functions may require a different FEM mesh. Finally, as a matter of practical concern, the IGA method can use data from CAD software directly, i.e. numerical analysis can be done on the actual geometry rather than relying on an intermediate meshing. As illustrated in [3] the creation of FEM meshes is one of the bottlenecks for large scale FEM calculations.

V. IGA WITH GEOPDES FOR REACTOR PHSYICS

The GeoPDEs software [6] provides a complete framework to do IGA-based calculations in Octave or Matlab[4]. GeoPDEs provides some basic routines to create NURBS geometries, and provides routines to set up common PDEs. In the present work, we have approached the problems as follows: first, calculations were done based on homogeneous geometry, using diffusion theory with 1 energy group. Then, GeoPDEs was used for multi-group diffusion calculations in multi-patch geometries. Finally, a first approach to transport theory was implemented.

[4] Available for download from `http://rafavzqz.github.io/geopdes/`

TABLE I: Comparison of GeoPDEs results with analytical calculations for diffusion theory.

	Cuboid $30\times45\times50$ cm	Cylinder $R=20$ cm $Z=40$ cm	Sphere $R=45$ cm
k_{eff}	1.126019	1.116927	1.316095
k error [%]	2.29×10^{-4}	1.84×10^{-3}	3.96×10^{-2}
ϕ error [%]	3.16×10^{-5}	1.06×10^{-4}	2.03×10^{-3}
time [s]	191	141	159
# elements	$5\times5\times5$	←	←
# quad points	$3\times3\times3$	←	←
degree	$2\times2\times2$	$3\times3\times3$	←

1. 1-group homogeneous diffusion theory

Calculations were done based on homogeneous geometry, using diffusion theory with 1 energy group. For the eigenvalue calculation, the conventional power method was applied. Solutions from GeoPDEs were compared with analytical calculations. The results are shown in Table I.

From the table, one can immediately identify a problem with GeoPDEs: the very long calculation time, which is unacceptable for large-scale problems. Another, much more practical problem, was found with the GeoPDEs software: it is not easy to create the NURBS geometries. For example, a circular surface can be created by first creating a curve (circular arc) of radius r_1, then creating a second curve (circular arc) of radius r_2, and then defining a ruled surface between the two curves, thus creating an annulus. r_1 must be positive in GeoPDEs, i.e. $r_1 = 0$ is not acceptable. This was checked with the developers, it is a practical problem of GeoPDEs, not a theoretical problem of IGA. In our current work, r_1 was set to a small value and reflective boundary conditions were applied on the boundary. The same problem affects 3D bodies, for example, a sphere can be created by first creating a half-annulus and then revolving over 2π; however, GeoPDEs cannot handle this geometry so some approximations are necessary. This is clearly something to be addressed if IGA is applied to nuclear reactor geometries. In Figure 4 we show an exaggerated image of the spherical body used in our work.

2. Multi-group, heterogeneous, diffusion theory calculations on multi-patch geometry.

In multi-group diffusion theory, the neutron diffusion equation is written as:

$$-\nabla\cdot D^g\nabla\phi^g+\Sigma_t^g\phi^g=\sum_{g'=1}^{G}\Sigma_s^{g'\to g}\phi^{g'}+\frac{\chi^g}{k}\sum_{g'=1}^{G}\nu\Sigma_f^{g'}\phi^{g'} \quad (4)$$

It is usual to write the RHS of this equation as two source terms, i.e. a scatter source S^g and a fission source F^g:

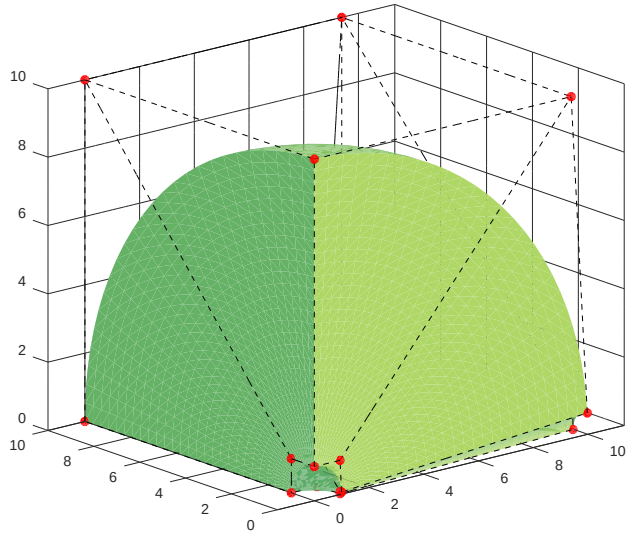


Fig. 4: NURBS spherical geometry. In our calculations, the radii of the "open spaces" are set to very small values and reflective boundary conditions are applied.

$$S^g = \sum_{g'=1}^{G} \Sigma_s^{g' \to g} \phi^{g'}$$

$$F^g = \frac{\chi^g}{k} \sum_{g'=1}^{G} \nu \Sigma_f^{g'} \phi^{g'}$$

$$-\nabla \cdot D^g \nabla \phi^g + \Sigma_t^g \phi^g = S^g + F^g$$

The last equation contains a divergence operator and a multiplication operator. GeoPDEs has routines to handle these operators. The scatter and fission term require knowledge of the flux. In our present implementation, we have used the standard GeoPDEs routines to construct the physical solution from the Finite Element solution. This step proved to be a rather time-consuming step in the algorithms and for dedicated reactor physics calculations the calculation of the source terms should be optimized.

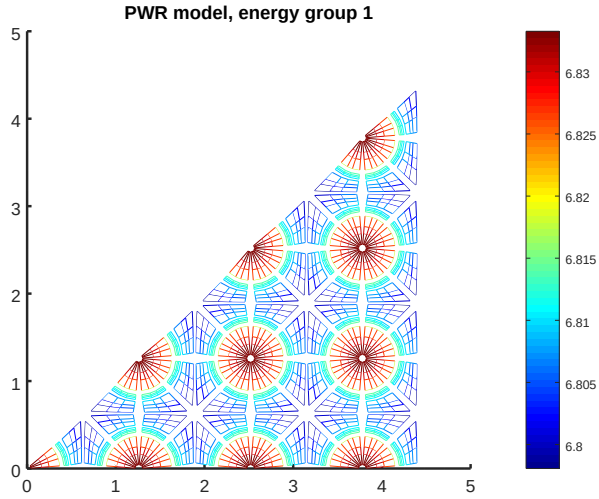
There are two ways to describe a heterogeneous geometry in GeoPDEs: one option is to use space-dependent material properties to set different cross sections for fuel, cladding, moderator, etc. In GeoPDEs the material properties can be continuous (user-supplied) functions, but the accuracy of such an approach is doubtful and the programming for the space-dependent cross sections can become complicated and error-prone. Thus, it was decided to use piecewise homogeneous regions, so that the geometry is made up of several NURBS-based parts; in NURBS parlance, this is called a *multi-patch* geometry. GeoPDEs has some support for multi-patch geometries but we have had to add quite a bit of programming for our purposes. The basic approach is to solve the equations on individual patches and then use continuity conditions to link the solutions between the patches.

A 2D PWR lattice was investigated. A 7×7 lattice of fuel pins was selected with reflective boundary conditions. Using symmetry properties, the model is reduced to a 1/8 sector.

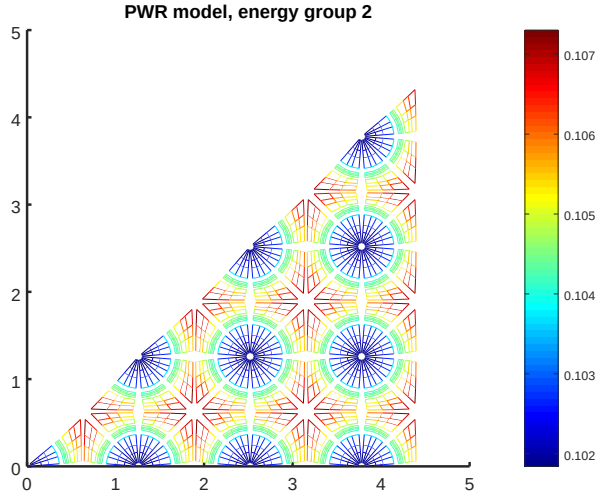
TABLE II: Heterogeneous, 4-group diffusion calculation on PWR model with GeoPDEs. Note: GeoPDEs calculation time includes time to render figures (approx. 130 s).

	GeoPDEs	T-NEWT	Δ
k_{eff}	1.30936	1.30714	0.17%
time [s]	2114	0.1	

The multipatch model was created using 45° patches, i.e. one complete PWR cell (fuel, clad, and coolant) is composed of 24 patches. The total model thus comprises 147 patches. Cross sections were prepared using the T-NEWT module of SCALE6 and collapsed to 4 energy groups. The flux maps are shown in Figure 5 and a comparison with T-NEWT is given in Table II. It is to be noted that T-NEWT uses transport theory rather than diffusion theory so a comparison is perhaps not completely jusitified.

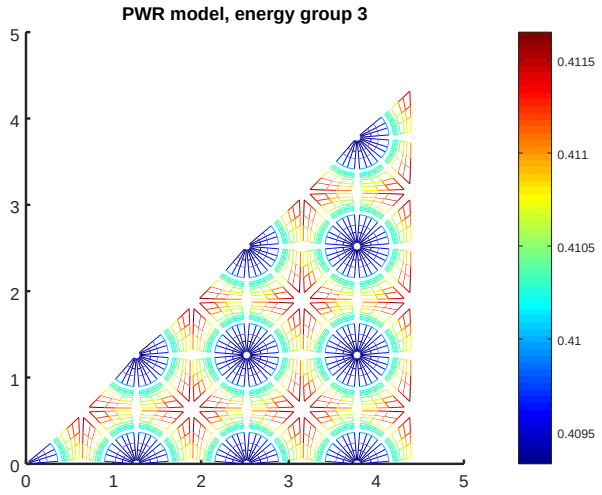


(a) Energy group 1.

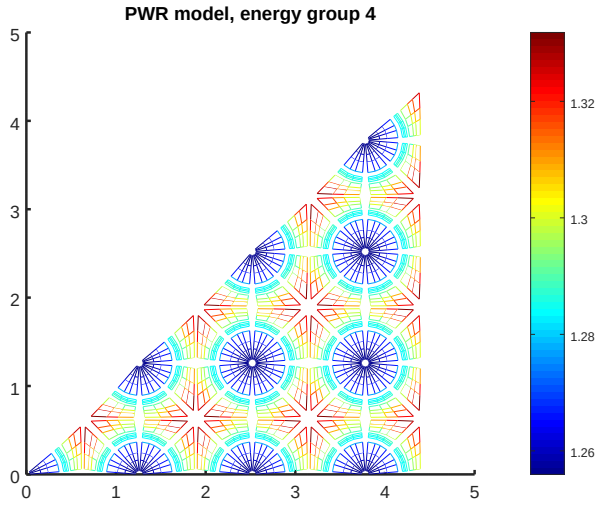


(b) Energy group 2.

Fig. 5: 1/8 sector of a PWR model, heterogeneous, multigroup diffusion calculation with GeoPDEs, flux map.



(c) Energy group 3.



(d) Energy group 4.

Fig. 5: 1/8 sector of a PWR model, heterogeneous, multigroup diffusion calculation with GeoPDEs, flux map.

3. Approach to transport theory

If one uses the conventional, multi-group S_N-method to solve the neutron transport equation, the basic equation to solve in the inner iterations is the following:

$$\hat{\boldsymbol{\Omega}}_m \cdot \boldsymbol{\nabla} \psi_m^g(\boldsymbol{r}) + \Sigma_t^g(\boldsymbol{r}) \psi_m^g(\boldsymbol{r}) = q_m^g(\boldsymbol{r}) \tag{5}$$

with $\psi_m^g(\boldsymbol{r}) \equiv \psi^g\left(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}_m\right)$ the angular neutron flux in direction $\hat{\boldsymbol{\Omega}}_m$ in energy group g, Σ_t^g the total cross section in group g, and $q_m^g \equiv q^g\left(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}_m\right)$ a general neutron source in group g injecting neutrons in direction $\hat{\boldsymbol{\Omega}}_m$. This source can be due to fissions, scattering, external, etc. This equation has the same form as the convection term which appears in the classic convection-dominated diffusion problem, where a term $\boldsymbol{v} \cdot \boldsymbol{\nabla} c$ appears. GeoPDEs has a routine `op_vel_dot_gradu_v` to calculate this type of equation; if the velocity function `vel` is chosen to be $\hat{\boldsymbol{\Omega}}_m$, GeoPDEs can be used to solve S_N-type equations.

With transport theory, the choice of the problem domain is limited. In diffusion theory it was possible to use annuli instead of cylinders, but in the case of transport theory the use of a reflective boundary condition on the inner surface of the annulus does not make physical sense. For this reason, trial calculations were performed on two simple 2D geometries as a proof of concept.

The first geometry is a rectangular domain, composed of a four rectangular patches. An angular source is present in two of the four patches. The resulting flux is shown in Figure 6. For these figures, $\hat{\boldsymbol{\Omega}}_m = \pi/6$. Important aspects are the calculation time (25.5 s) and the presence of “ripples" in the solution. As explained in the GeoPDEs manual, these are due to the implementation of the boundary conditions, an area which we will continue to study.

A second trial calculation was done on a hexagonal geometry. There are 7 hexagons in total, and each hexagon is built up of 3 quadrilateral patches, for a total of 21 patches. Some patches contain sources, injecting neutrons in a direction $\hat{\boldsymbol{\Omega}} = \pi/7$. The resulting flux map is shown in Figure 6.

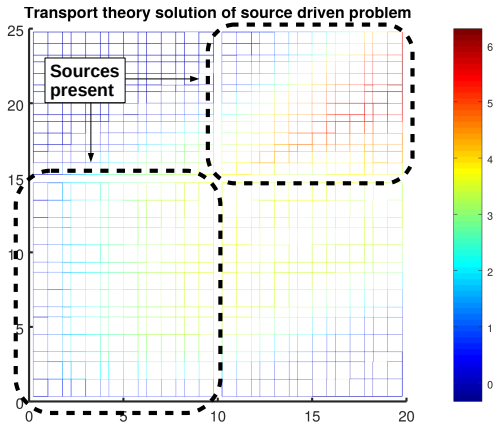
From the aforegoing results, GeoPDEs can be used for calculations with the S_N-method, but the calculation time is prohibitive. For example, the calculation in the hexagonal geometry required 121.8 s, for one energy group, in one direction.

VI. CONCLUSION AND DISCUSSION

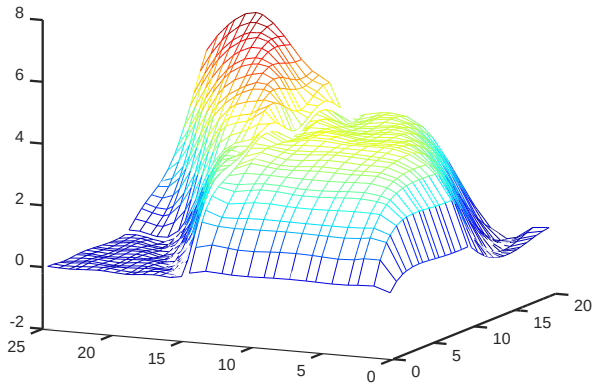
In the present work we have used the GeoPDEs software to perform calculations using the IGA-method for neutron diffusion theory and neutron transport theory in multiplying systems. Our conclusion based on the work is that the GeoPDEs software is capable of simulating all the important aspects of conventional multi-group diffusion and transport theory, but the calculation time is unacceptable. We did not investigate the cause of the long calculation time. Codes like Octave are well-known for being user-friendly, but slow. It may be possible to accelerate the GeoPDEs software but this was not attempted in the current work.

One powerful feature of GeoPDEs is that the physical properties of the domain are treated as continuous functions, opening the possibility to perform detailed multi-physics calculations. This feature also causes GeoPDEs to be slow for reactor physics calculations: to determine the sources due to fission and scatter, the solution has to be transformed from the parametric domain to the physical domain to obtain the physical neutron flux, then perform the multiplications with the cross sections, and subsequently the updated neutron source must be transformed back to the parametric domain.

In the present work, the biggest practical problem was the creation of the geometry based on NURBS. GeoPDEs offers some functions to create elementary patches, such as circular arcs, rectangles, etc, but the creation of multi-patch geometries proved to be a rather time-consuming problem. Another issue is that the NURBS geometries in GeoPDEs must retain the unique transform between parametric and physical domain, which makes it impossible to make for example true circular domains. For future development of the IGA method, it would be attractive if dedicated CAD software can be used, such as

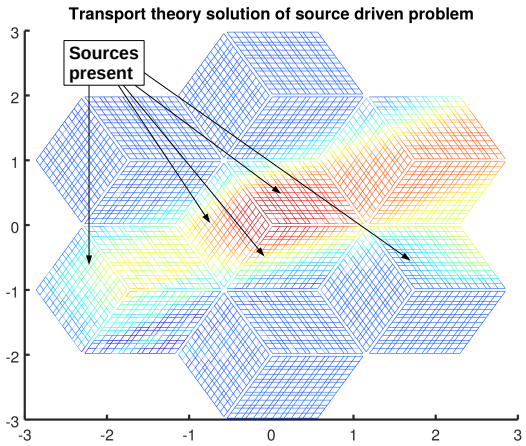


(a) S_N calculation, square geometry.

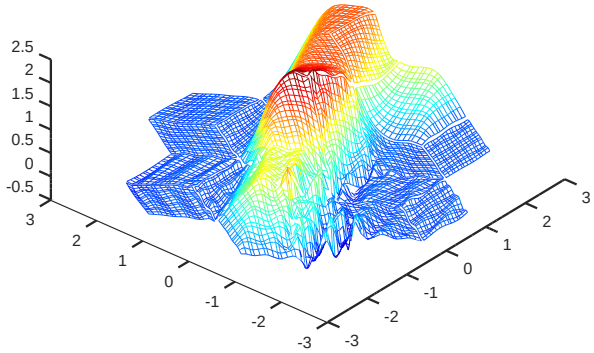


(b) S_N calculation, square geometry, 3D view.

Fig. 6: S_N calculation in 2D square geometry.



(c) S_N calculation, hexagonal geometry.



(d) S_N calculation, hexagonal geometry, 3D view.

Fig. 6: S_N calculation in 2D hexagonal geometry.

FreeCAD[5], to create, check, and validate the definition of the problem domain.

GeoPDEs (and the NURBS toolbox included in GeoPDEs) are useful to study the IGA method, and useful as examples of how to implement the IGA method in practice. In the future, we aim to create a code based on the IGA-method to perform multi-group neutron transport theory calculations with the S_N-method (similar to work presented by other researchers [7]). We also target the extension to multi-physics calculations.

REFERENCES

1. "Benchmark Analyses of EBR-II Shutdown Heat Removal Tests," Tech. Rep. TECDOC, International Atomic Energy Agency (IAEA) (2017 (under review)).
2. T. TECHNICAL COMMITTEE ON MONJU RESEARCH UTILIZATION, "External evaluation on Monju core confirmation test in FY2010," Tech. Rep. JAEA-Evaluation 2011-002, Japan Atomic Energy Agency, FBR Plant Engineering Center, Fast Breeder Reactor Research and Development Center, Tsuruga Head Office (2011).
3. J. A. COTTRELL, T. J. HUGHES, and Y. BAZILEVS, *Isogeometric Analysis*, Wiley (2009).
4. L. PIEGL and W. TILLER, *The NURBS Book*, Springer (1995).
5. W. VAN ROOIJEN, "On the use of NURBS for particle transport calculations," in "Fall meeting of the Atomic Energy Society of Japan," AESJ (September 2014), H12.
6. R. VÁZQUEZ, "A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0," Tech. Rep. IMATI REPORT Series Nr. 16-02, Instituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche (April 2016).
7. A. OWENS, J. WELCH, J. KÓPHÁZI, and M. EATON, "Discontinuous isogeometric analysis methods for the first-order form of the neutron transport equation with discrete ordinates (S_N) angular discretization," *J. Comp. Phys.*, **315**, 501 - 535 (2016).

[5] Available at `https://www.freecadweb.org/`