Three-dimensional $h$-adaptivity for the multigroup neutron diffusion equations

Yaqi Wang$^1$, Wolfgang Bangerth$^{2, 3}$, Jean Ragusa$^1$

$^1$Department of Nuclear Engineering, Texas A&M University
College Station, TX 77843, USA (yaqiwang@tamu.edu, ragusa@ne.tamu.edu)

$^2$Department of Mathematics, Texas A&M University
College Station, TX 77843, USA (bangerth@math.tamu.edu)

Abstract

Adaptive mesh refinement (AMR) has been shown to allow solving partial differential equations to significantly higher accuracy at reduced numerical cost. This paper presents a state-of-the-art AMR algorithm applied to the multigroup neutron diffusion equation for reactor applications. In order to follow the physics closely, energy group-dependent meshes are employed. The method is formulated using finite elements of any order, for any number of energy groups. The spatial error distribution is assessed with a generalization of an error estimator originally derived for the Poisson equation.

Our implementation of this algorithm is based on the widely used Open Source adaptive finite element library deal.II and is made available as part of this library’s extensively documented tutorial. We illustrate our methods with results for 2-D and 3-D reactor simulations using 2 and 7 energy groups, and using finite elements of polynomial degree up to 6.

Key words: Finite elements, adaptive mesh refinement, multigroup diffusion approximation, reactor simulation.

1 Introduction

In many engineering disciplines, from structural mechanics to fluid dynamics, automated Adaptive Mesh Refinement (AMR) is now an established technique for obtaining numerical solutions with higher accuracy and resolution, while requiring less memory and shorter CPU times. Adaptivity, which can be traced back to the late 1970s [3], is based on the idea that in order to achieve
high accuracy, a uniformly fine mesh is not necessarily required; rather, the computational grid only needs to be fine in regions where the solution is rough and can be coarse in areas where the solution is smooth and, therefore, well resolved even on large cells. The challenge is that, in general, it is not known a priori where the solution will require the mesh to be fine. Consequently, the computation of local error or smoothness indicators from a numerical solution, previously obtained on a coarser mesh, lies at the heart of all adaptive mesh refinement algorithms, and a significant number of successful approaches have been developed for this problem in the last decade [1,4,9,12]. Using these methods, it has been shown for many problems that the computational effort needed to reach a certain accuracy can often be reduced by one or several orders of magnitude compared to uniform meshes, frequently enabling the solution of entire new classes of problems that were previously considered too computationally expensive to solve to the required accuracy.

For a variety of reasons, AMR has not yet been widely used in nuclear science and engineering, with only a few references available in the literature (see our review below). Among the main causes is that the correct description of many processes in neutronics applications in nuclear engineering is the transport equation [10,20], an equation that is extraordinarily complicated to solve numerically, even in the absence of advanced numerical methods like adaptivity. These complications arise from its hyperbolic character that allows discontinuous solutions and in particular its dependence on seven variables (three space dimensions, the two-dimensions of direction, time, and energy).

Consequently, a significant number of approximations have been developed to make it tractable for particular cases. For example, the multigroup diffusion approximation is often employed in 3-D reactor analysis and design [16,29]. This approximation leads to a collection of partial differential equations for individual energy groups that need to be solved concurrently. While adaptive mesh refinement techniques have been investigated for a wide variety of engineering problems in the past, the multigroup approximation that arises in the treatment of the transport or diffusion equations has properties that require a re-analysis of straightforward approaches. Among its particularities is that, unlike most other coupled systems the authors are aware of, individual solution components are not necessarily smooth or rough in the same parts of the domain (see the numerical examples in Section 4). This is easy to understand: fast particles typically travel relatively unimpeded through fine-scale features of the medium and their spatial distribution is therefore a relatively smooth function; on the other hand, thermal (slow) particles typically interact more locally and strongly with materials, and the resulting large variations of cross sections values and mean-free-path lengths then lead to less smooth solutions. Thus, it is not a priori clear that it is worthwhile to discretize different energy groups with a single mesh.
In this contribution, we propose and evaluate an automated mesh adaptation technique based on the use of arbitrary-order finite elements that is applied to the multigroup diffusion approximation. The main feature of this method is the use of separate meshes for individual energy groups. While in general the use different meshes can lead to expensive algorithms, we will show how to efficiently use our approach by using meshes that are hierarchically refined from a common coarse grid. The spatial integration of terms that involve shape functions from different meshes can therefore be performed efficiently by considering only prolongations of shape functions from parent to child cell; this will be needed for group coupling terms such as scattering and fission events. We will illustrate this approach using $k$-eigenvalue benchmark problems taken from reactor analysis applications. We will present our algorithms and demonstrate their advantages over uniformly refined meshes using two 2-D 2-group examples as well as a 3-D 7-group example of significant complexity.

Our implementation of these algorithms are freely available as part of the widely used Open Source finite element library deal.II [6, 7]. The tutorial step-28 of deal.II contains the code developed in this present work; this tutorial is extensively documented and we explicitly encourage the use of this program as the basis for further experiments by other researchers. The results presented here use a slightly optimized but otherwise unchanged version of tutorial step-28. Since deal.II already provides most of the functionalities required for a project like this (including, for example, the handling of adaptive meshes, a variety of finite elements, all components of linear algebra, and input/output features), step-28 is in fact a relatively small code, of about 1,200 lines of code, with several times that amount in documentation. Together with the extensive documentation of all aspects of deal.II (more than 5,000 pages of API documentation along with more than 30 tutorial programs), we believe that step-28 is a good starting point for future research in this area.

We conclude this section by a brief review of the literature on adaptive mesh refinement for transport problems. For instance, in radiative transfer, where a transport (or diffusion) solver is frequently coupled to AMR hydrodynamics codes, there has been a natural inclination to develop AMR capabilities for the transport/diffusion solvers as well. In [23], the gradient of the solution is employed to drive adaptive mesh refinement for the radiation transport component in 2-D Cartesian geometries for one-group (one-frequency) equations. In [2], local refinement techniques based on the value of the neutron mean-free-path in a given cell are used for the one-group transport equation in 3-D Cartesian geometries. While this approach takes into account the size of potential internal layers at a given location in the domain, it does not account for the actual smoothness of the solution at these locations and is therefore far from optimal; for example, in optically thick areas, the solution may well be approximated by a smooth spatial representation on coarse meshes despite the smallness of the mean-free-path. Klar et al. [27] consider a coupled
radiation-temperature model and apply adaptive methods to it that can resolve the boundary layer of a hot, homogenous body that is in thermal contact with a cooler exterior. In neutronics applications, Lewis et al. [38] presented a $p$-refinement technique (adaptive selection of the polynomial approximation) for the inter-element approximation in a primal hybrid finite element technique. Results were given for a 2-D 1-group diffusion problem; later they expanded the technique to the 2-D 1-group $P_N$ equations [39]. Finally, the authors of [32, 33] employed an error indicator based on estimating the second derivatives of the numerical solution and applied the resulting method to 1-group and 2-group diffusion applications. The error indicator used there is based on the interpolation error of linear finite elements [22] and is therefore limited to these elements. For higher polynomial approximations (say of order $p$), this method can be generalized by using the tensor of $p + 1$-st derivatives of the numerical solution (see the general discussion in [17] and application in [31] to the adaptive finite element simulation of aerodynamic flows). More recently, Wang and Ragusa applied the $hp$-adaptation concept to the multigroup diffusion equations, where both the local polynomial degree of shape functions and the local cell size are selected adaptively [36, 37]. Error estimators for the combined $hp$ adaptation are less mature and, as a consequence, these authors used the difference between a finer mesh solution $\Phi_{\text{fine}}$ and a coarser mesh solution $\Phi_{\text{coarse}}$ to drive the refinement [18], with an overhead due to the computation of a fine solution at each step of the adaptation. Their results included mesh adaptation for 1-D multigroup and 2-D 1-group diffusion problems. Finally, Kanschat used fully adaptive finite approximations to the stationary, monochromatic radiative transfer problem [24, 34].

An outline of the remainder of this contribution is as follows: In Section 2, we briefly introduce the mathematical formulation used for the multigroup diffusion approximation eigenproblem. Section 3 presents the discrete formulation upon which we base our finite element scheme, the error indicator used to select mesh cells for refinement, and the mesh refinement scheme. In particular we also discuss the specificities of having group-dependent meshes and how an implementation can be made efficient. Finally, in Section 4, 2-D and 3-D results are presented using 2-group and 7-group diffusion problems. We conclude in Section 5 and present an outlook on open problems and possible next steps.
2 Mathematical formulation

2.1 The steady state multigroup diffusion equations

This goal of this paper is the development of automated spatial adaptivity for the numerical solution of the multigroup neutron diffusion equations. These equations can be derived as the diffusion dominated limit of the neutron transport equation and by binning neutron energies into a finite number $G$ of groups [10, 19]. Such problems in nuclear engineering typically involve either a criticality search eigenproblem or a source-driven problem. Here, we will only present the eigenproblem case; the source-driven case is generally simpler and can be dealt with using the same techniques.

For conciseness, let us consider the neutron flux $\phi_g(r)$, $1 \leq g \leq G$ in the $G$ different energy groups. In the absence of external neutron source, these satisfy the balance equations

$$- \nabla \cdot (D_g(r) \nabla \phi_g(r)) - \Sigma_{r,g}(r) \phi_g(r) = \sum_{g' \neq g} \Sigma_{s,g'\rightarrow g}(r) \phi_{g'}(r) + \frac{\lambda_g}{k_{\text{eff}}} \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(r) \phi_{g'}(r), \quad r \in \Omega, 1 \leq g \leq G.$$  

(1)

The various coefficients in these equations have the following meaning:

- $D_g(r)$ is the diffusion coefficient in group $g$;
- $\Sigma_{r,g}(r)$ is the removal cross section in group $g$;
- $\Sigma_{f,g}(r)$ is fission cross section in group $g$, $\nu$ is the number of neutrons emitted per fission event, and $\lambda_g$ is the fission spectrum. The quantity $S_f(r) = \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(r) \phi_{g'}(r)$ is often referred to as the fission integral;
- $\Sigma_{s,g'\rightarrow g}$ is the scattering cross section from group $g'$ to group $g$.

The equations above are augmented by Dirichlet boundary conditions $\phi_g = 0$ on the exterior boundaries of the domains we consider in Section 4 and boundary conditions $n \cdot \nabla \phi_g = 0$ on planes of symmetry since we only consider one quadrant of a full reactor in our examples.

2.2 The eigenvalue problem

Considering all energy groups at once, we can re-write equation (1) in the following operator form:
where $L$, $X$, and $F$ are the loss, scattering, and fission operators, respectively. $L$ here includes both the leakage and removal terms. Note that $L$ is symmetric, whereas $F$ and $X$ are not. The multiplicative factor $k_{\text{eff}}$ is the largest (and unique) eigenvalue of the operator $(L - X)^{-1}F$ (Krein-Rutman theorem, [28]).

The goal of this paper is to solve the above $k$-eigenvalue problem using spatial adaptivity. Our numeric implementation uses the power method with Chebyshev acceleration [21]. The basic algorithm for this is as follows:

1. Initialize the multigroup fluxes and the eigenvalue with $\phi^{(0)}_g, k_{\text{eff}}^{(0)}$; compute the initial the fission integral $S_f^{(0)} = \sum_{g=1}^{G} \nu \Sigma_{f,g}(r)\phi^{(0)}_g(r)$ and set the iteration index to $n = 1$.
2. Solve for the multigroup fluxes $\phi^{(n)}_g, g = 1, \ldots, G$ using
   \[
   -\nabla \cdot D_g \nabla \phi^{(n)}_g + \Sigma_{r,g} \phi^{(n)}_g = \sum_{g' \neq g} \Sigma_{s,g'-g} \phi^{(n)}_g + \frac{\lambda_g}{k_{\text{eff}}^{(n-1)}} S_f^{(n-1)}. \]
3. Without upscattering of particles from lower energy to higher energy groups the above system is lower triangular in energy groups. When upscattering is present, the linear system is no longer triangular and an additional iterative loop is needed to solve for the thermal fluxes.
4. Update the fission integral: $S_f^{(n)}(r) = \sum_{g'=1}^{G} \nu \Sigma_{f,g'}(r)\phi^{(n)}_{g'}(r)$.
5. Update the eigenvalue: $k_{\text{eff}}^{(n)} = \frac{\int_{\Omega} S_f^{(n)}(r)}{\int_{\Omega} S_f^{(n-1)}(r)} k_{\text{eff}}^{(n-1)}$.
6. Compare $k_{\text{eff}}^{(n)}$ with $k_{\text{eff}}^{(n-1)}$ and $\phi^{(n)}$ with $\phi^{(n-1)}$. If the changes are greater than a prescribed tolerance, then set $n = n + 1$ and repeat the iteration starting at step 2; otherwise end the iteration.

We implement this algorithm to solve for the multiplication factor $k_{\text{eff}}$ and the associated fundamental mode using the adaptive finite element scheme described in the following sections.

3  $h$-Adaptivity for the multigroup diffusion equations

Before describing our spatially adaptive algorithm, let us first consider the question whether it is adequate to solve for the fluxes $\phi_g$ of all energy groups $g = 1, \ldots, G$ on the same mesh $\mathbb{T}$, or whether it would be more appropriate to discretize each flux $\phi_g$ on its own mesh $\mathbb{T}_g$. To this end, it is important to realize that a single mesh $\mathbb{T}$ can only be suitable for resolving all solution components $\phi_g$ simultaneously to an adequate degree if it is fine wherever at
least one of these functions is non-smooth. If all components are non-smooth to the same degree in the same locations, then a single mesh is adequate. On the other hand, if the components $\phi_g$ differ in their degree of smoothness or if they are non-smooth in disjoint areas, then a single mesh would be too fine in some areas for some of the components; consequently the mesh would be wasteful and lead to unnecessarily many degrees of freedom.

As will become obvious by looking at images of the individual components of solutions in Section 4, multigroup diffusion problems fall in the second category. In particular, it is physically obvious that fast neutrons tend to travel farther (due to smaller cross sections) and are, therefore, less affected by local variations in the material properties, whereas thermal neutrons “see” more localized details. Hence, smoother solutions are usually expected for higher energy neutrons and more rapidly varying solutions for lower energy neutrons. We can therefore expect that meshes that are adjusted to each energy group solution are significantly more efficient than using a single mesh for all groups.

The use of separate meshes for different energy groups has two principal consequences for their efficient implementation. First, we need to find a way to refine the meshes individually. Secondly, when assembling the cross-group source terms (fission and scattering) in the power iteration, we have to integrate the solution $\phi^{(n)}_g$ defined on mesh $T_g$ against the shape functions defined on mesh $T_g$. We will discuss efficient algorithms for these tasks below, following a concise statement of the finite element discretization we use.

### 3.1 Discrete formulation

In order to discretize the $n$th power iteration of the eigenvalue problem of Eq. 1, we approximate $\phi^{(n)}_{g,h}(r) = \sum_{i=1}^{N_g} \Phi^{(n)}_{g,i} \phi^{(n)}_i(r)$ with coefficients $\Phi^{(n)}_{g,i}$ and finite element shape functions $\phi^{(n)}_i(r)$ defined on meshes $T_g$, $g = 1, \ldots, G$, see [11, 13, 14]. The coefficients $\Phi^{(n)}_{g,i}$ are the unknowns to be solved for. For lowest-order (linear) continuous elements, shape functions are associated with the vertices of the mesh $T_g$. However, below we will also use higher order shape functions up to polynomial degree 6 that are associated with the faces, the edges, and the interior of cells. In the following sections, we will only consider meshes that consist of quadrilaterals (in 2-D) and hexahedra (in 3-D); however, all algorithms are readily applicable to triangular or tetrahedral elements as well.

Using a Galerkin procedure to determine the coefficients $\Phi^{(n)}_{g,i}$, we arrive at the following weak discrete form of (3), which we obtain by multiplying that equation by test functions $\varphi^{(n)}_i$, integrating over the spatial domain $\Omega$, and
integrating by parts all terms involving second derivatives:

\[
(D_g \nabla \phi_{g,h}^{(n)}, \nabla \varphi_g^{j}) + (\sum_{r,g'} \phi_{g',h}^{(n)} \varphi_g^{j}) = \sum_{g' \neq g} (\sum_{s,g'} \phi_{g',h}^{(n)} \varphi_g^{j}) + (\chi_g \mathbb{S}_g^{(n-1)}, \varphi_g^{j}).
\]

(4)

Here, \((u, v) = \int_\Omega u(r)v(r) \, d^3r\). Since this has to hold for each of the test functions \(\varphi_g^{j}, j = 1, \ldots, N_g, g = 1, \ldots, G\), we obtain a system of linear equations for the coefficients \(\Phi_g^{(n)}\) that can be solved using a preconditioned conjugate gradient solver, owing to its symmetry and positive definiteness.

Instead of going into the technical details of the other steps of the iterative procedure to compute \(k_{\text{eff}}\), which can be found in [35] for instance, we will focus on the implications of automatically choosing adaptive meshes in the remainder of this section.

3.2 Outer mesh iteration

In adaptive mesh refinement algorithms, one usually starts with a coarse mesh in order to compute a rough approximate solution. This solution is then used to compute a refinement indicator that states on which cells the error is likely to be the largest. The cells with the largest error are then refined, and the solution procedure starts over on this new mesh. The mesh adaptation iteration proceeds until we are satisfied with the accuracy of the solution. Since each mesh has more cells than the previous one, the total compute time is usually dominated by computations on the last mesh. The additional effort necessary to achieve a well-chosen mesh for a given accuracy (i.e., computing solutions and refinement indicators on coarser meshes) is, therefore, not a large factor in adaptive mesh calculations.

For the present problem, we solve the entire eigenvalue problem on one set of meshes \(T_g^m, g = 1, \ldots, G\), where \(m\) denotes the mesh iteration number. We then compute refinement indicators \(\eta_{g,K}\) for all cells \(K \in T_g^m\). With these indicators, we refine the mesh \(T_g^m\) to obtain the next mesh \(T_g^{m+1}\). The final solution \(\phi_{g,h}^{(n)}\) on the old mesh \(T_g^m\) is interpolated to the new mesh to obtain the initial guess \(\phi_{g,h}^{(0)}\) for the power iteration procedure on the next set of meshes, \(T_g^{m+1}\).

Because all following discussions will only concern a single set of meshes, we will drop the mesh iteration index \(m\) in the remainder of this paper, except where ambiguous.
3.3 Mesh refinement

The central component of the mesh iteration is the estimation of the error on each cell $K$ of each of the meshes $T_{g}^{m}$, $g = 1, \ldots, G$, in order to obtain refinement and coarsening criteria. Such error indicators are ideally derived using a posteriori error estimators using strict upper bounds on the total error [1]. For the current nonlinear eigenvalue problem (2), strict estimates or error approximations could be derived using the techniques discussed in [9] but would be complicated to implement and expensive to evaluate.

On the other hand, experience has shown that oftentimes it is sufficient to have error indicators to drive mesh refinement. Such indicators typically determine the smoothness of a solution, by computing estimates of higher order derivatives. In our calculations, we use an indicator that uses the formula derived by Kelly et al. in [22]. In that paper, an a posteriori error estimator is derived for the Poisson equation; it approximates the error per cell by integrating the jump of the gradient of the solution along the faces of each cell, i.e., by computing

\[
\rho_{g,K} = \sqrt{h_{K}} \left\| \mathbf{n} \cdot \nabla \phi_{g,h}^{(n)} \right\|_{\partial K},
\]

where $h_{K}$ is the diameter of cell $K \in T_{g}$, $g = 1, \ldots, G$, $\partial K$ represents the edges in 2-D or the faces in 3-D of cell $K$, and $\mathbf{n}$ is the normal unit vector on $\partial K$. This quantity is equivalent to the norm of the second derivatives of a function, up to a power of $h_{K}$. While this is an error estimator for the Poisson equation, it is only an error indicator for equations containing a Laplace operator. Nonetheless, this indicator is widely used as a heuristic refinement indicator in many applications and is considered a good choice in the absence of actual estimators for a particular equation (see, for example, [5, 8, 25]) and is implemented in a number of standard finite element software packages [6, 7, 26]. The obvious generalization of the indicator above to cases where the diffusion operators contains a non-constant diffusion coefficient is to use the jump of net current,

\[
\rho_{g,K} = \sqrt{h_{K}} \left\| D_{g} \mathbf{n} \cdot \nabla \phi_{g,h}^{(n)} \right\|_{\partial K}.
\]

This is the form that we will use in our numerical experiments below.

The next question is how to use these indicators to refine the cells of all the triangulations $T_{g}$. In general, it needs to be understood that the quantities $\rho_{g,K}$ only indicate the magnitude of the second derivative of the solution component $\phi_{g}$ at the location of cell $K$, and thereby presumably the magnitude of the local error $\phi_{g} - \phi_{g,h}$. However, $\rho_{g,K}$ does not contain any information regarding the quantity of interest (for example the eigenvalue $k_{\text{eff}}$) that drives the numerical
solution, or, in other words, why it is important to have a small local error on cell $K$. For example, in two-group diffusion problems, the fast flux is about one order of magnitude larger than the thermal flux; if the solutions would otherwise have similar qualitative behavior, then the indicators $\rho_{g,K}$ would also be ten times larger for the fast neutron flux. However, this does not mean that the fast group mesh should be preferentially refined: suppose, for example, that the quantity we are interested in is the power distribution – a quantity mostly driven by thermal neutrons in light water reactors due to the significantly higher thermal cross sections – then refinement of the thermal group mesh is clearly of importance as well.

A good error indicator would therefore multiply the smoothness indicator $\rho_{g,K}$ by an importance factor $z_{g,K}$ to obtain refinement indicators $\eta_{g,K} = \rho_{g,K} z_{g,K}$. Such importance factors can be computed using duality techniques [9], however at the price of a significant mathematical and computational overhead. In this contribution, we do not intend to describe such a theory and instead will choose the importance factors heuristically as $z_{g,K} = (\|\phi^{(n)}_{g,h}\|_{L^\infty(\Omega)})^{-1}$, thereby ensuring equal weight for all energy groups. An alternative choice we have investigated is $z_{g,K} = \|\Sigma_{f,g}\|_{L^\infty(K)}$ (i.e., the magnitude of the fission cross section on each cell), a choice suggested by the importance with which $\phi_g$ appears in the fission integral that is employed in the computation of $k_{\text{eff}}$; however, despite the fact that it makes intuitive sense, this choice does not lead to an efficient refinement indicator and performs significantly worse than the simple choice above in numerical experiments, mostly due to the fact that regions containing no fissile materials, such as reflector zones, are not considered for refinement with this alternate choice.

Using above definition for our refinement indicators $\eta_{g,K}$, we refine a mesh cell $K$ if

$$\eta_{g,K} > \alpha_1 \max_{1 \leq g \leq G, K \in T_g} \eta_{g,K}$$

and coarsen it if

$$\eta_{g,K} < \alpha_2 \max_{1 \leq g \leq G, K \in T_g} \eta_{g,K}.$$

In the numerical experiments shown in Section 4, we use $\alpha_1 = 0.3$ and $\alpha_2 = 0.01$.

An interpretation of this strategy is that if for a given energy group $g$ there are many cells $K \in T_g$ for which the refinement indicators are large, due to, for example, roughness in the solution of large importance factor, then many cells will be above the refinement threshold and will be selected for refinement. On the other hand, if there are a few cells with large errors and many cells with small errors, for example because the solution is overall rather smooth except at a few places, then only the few cells with large refinement indicators will be refined. Finally, if the solution for a given energy group is smooth everywhere, then none of its cells will be refined until the refinement indicators for cells
on the meshes of other energy groups have been sufficiently reduced by mesh refinement. Consequently, the strategy allows for meshes that track the global smoothness properties of the solutions of all energy groups equally well.

3.4 Assembling terms on different meshes

A consequence of using different meshes for different energy groups is that in the discrete counterpart of step 3 of the eigenvalue iteration, i.e., in equation (4), we have to compute a right hand side vector that contains terms of the following form:

\[ F_i = \int_\Omega f(\mathbf{r}) \phi_g^{(n)}(\mathbf{r}) \varphi_g^i(\mathbf{r}) \, d^3r, \]  

where \( f(\mathbf{r}) \) is one of the coefficient functions \( \Sigma_{s,g'} \rightarrow g \) or \( \chi_{g'} \Sigma_{f,g'} \) used in the right hand side of the eigenvalue equation. \( \phi_g \) can be expanded as \( \phi_{g'}(\mathbf{r}) = \sum_j \Phi_j^{g'} \varphi_j^{g'}(\mathbf{r}) \), with basis functions \( \varphi_j^{g'}(\mathbf{r}) \). Consequently, the contribution to the right hand side can be written as

\[ F_i = \sum_{j=1}^{N_{g'}} \left\{ \int_\Omega f(\mathbf{r}) \varphi_g^i(\mathbf{r}) \varphi_j^{g'}(\mathbf{r}) \, d^3r \right\} \Phi_j^{g'}. \]  

Note that the test functions \( \varphi_g(\mathbf{r}) \) are defined on mesh \( \mathbb{T}_g \), whereas the basis functions \( \varphi_j^{g'}(\mathbf{r}) \) are defined on mesh \( \mathbb{T}_{g'} \). Hence, it is not possible to simply split the integral over \( \Omega \) into integrals over the cells of either mesh \( g \) or \( g' \).

In general, integrating terms that involve functions defined on two entirely different meshes is an expensive procedure, since it involves finding the cell of one mesh in which a quadrature point defined on the other mesh lies. However, this problem can be avoided if we use hierarchical meshes that all result from regular refinement of the same initial coarse mesh. In that case, we assume that all energy groups start out with the same coarse mesh, i.e., \( \mathbb{T}_0^g = \mathbb{T}_0^{g'} \), and that we refine a mesh by regular bisection of cells (i.e., quadrilaterals in 2-D are subdivided into four child-cells whereas hexahedra in 3-D yield eight child-cells). The cells on each of the two grids will then remain related and we can always find a set of cells, which we denote by \( \mathbb{T}_g \cap \mathbb{T}_{g'} \), that satisfy the following conditions:

- the union of the cells covers the entire domain, and
- a cell \( K \in \mathbb{T}_g \cap \mathbb{T}_{g'} \) is active on at least one of the two meshes.

Here, we denote a cell as active if it is not refined any further, i.e., it is a cell on which shape functions are defined. A cell becomes inactive after it is refined.
A way to construct this set of cells is to take each cell of the coarse mesh and do the following steps: (i) if the cell is active on either $T_g$ or $T_{g'}$, then add this cell to the set; (ii) otherwise, i.e., if this cell has children on both meshes, then do step (i) for each of the children of this cell. With this, we can write (8) as follows:

$$F_i = \sum_{K \in T_g \cap T_{g'}} F_i|_K, \quad F_i|_K = \sum_j \left\{ \int_{K} f(r) \, \varphi^i_g(r) \, \varphi^j_{g'}(r) \, d^3r \right\} \Phi^j_{g'}. \quad (9)$$

By construction, there are now three cases to be considered:

(i) The cell $K$ is active on both meshes, i.e., both the basis functions $\varphi^i_g$ as well as $\varphi^j_{g'}$ are defined on $K$.
(ii) The cell $K$ is active on mesh $g$, but not on mesh $g'$, i.e., the $\varphi^i_g$ are defined on $K$, whereas the $\varphi^j_{g'}$ are defined on children of $K$.
(iii) The opposite case, i.e. the cell $K$ is active on mesh $g'$, but not on mesh $g$.

To compute the right hand side contributions above, we need to consider each of these three cases separately:

(i) If the cell $K$ is active on both meshes, then we can directly evaluate the integral since both sets of shape functions are defined on this cell.
(ii) If the cell $K$ is active on mesh $g$, but not on mesh $g'$, then the basis functions $\varphi^j_{g'}$ are only defined either on the children $K_c$, with $0 \leq c < 2^{\dim}$, or on children of these children if cell $K$ is refined more than once on mesh $g'$.

Let us assume that $K$ is only refined once more on mesh $g'$ than on mesh $g$. Using the fact that we use embedded finite element spaces where each basis function on one mesh can be written as a linear combination of basis functions on the next refined mesh, we can expand the restriction of $\varphi^i_g$ to child cell $K_c$ into the basis functions defined on that child cell (i.e., on cells on which the basis functions $\varphi^j_{g'}$ are defined):

$$\varphi^i_{g,h}|_{K_c} = \sum_l B^i_{c,l} \varphi^j_{g'}|_{K_c}. \quad (10)$$

$B_c$ is the matrix that interpolates data from a cell to its $c$-th child cell. We then write the contribution of cell $K$ to the right hand side component $F_i$ as

$$F_i|_K = \sum_j \left\{ \int_{K} f(r) \, \varphi^i_g(r) \, \varphi^j_{g'}(r) \, dx \right\} \Phi^j_{g'}$$

$$= \sum_j \left\{ \sum_{c=1}^{2^{\dim}} \sum_l B^i_{c,l} \int_{K_c} f(r) \, \varphi^j_{g'}(r) \, \varphi^j_{g'}(r) \, d^3r \right\} \Phi^j_{g'}.$$
In matrix notation, this can be written as
\[
F_i|_K = \sum_{c=1}^{2^{\dim K}} F_i|_{K_c}, \quad F_i|_{K_c} = \sum_j \sum_l B^l_c M^{ij}_{K_c} \Phi^j_{g'}, = \sum_j (B_c M_{K_c})^{ij} \Phi^j_{g'},
\]
where \( M^{ij}_{K_c} = \int_{K_c} f(r) \varphi^i_g(r) \varphi^j_{g'}(r) \, d^3r \) is the weighted mass matrix on child \( c \) of \( K \).

On the other hand, if a child \( K_c \) of \( K \) is not active, then we have to apply the process recursively, i.e., we have to interpolate the basis functions \( \varphi^i_g \) onto child \( K_c \) of \( K \), then onto child \( K_{cc'} \) of that cell, onto child \( K_{cc'c''} \) of that one, etc, until we find an active cell. We then have to sum all the contributions from all the children, grandchildren, etc, of cell \( K \), with contributions of the form
\[
F_i|_{K_{cc'}} = \sum_j (B_c B_{c'} M_{K_{cc'}})^{ij} \Phi^j_{g'},
\]
(11)
or
\[
F_i|_{K_{cc'c''}} = \sum_j (B_c B_{c'} B_{c''} M_{K_{cc'c''}})^{ij} \Phi^j_{g'},
\]
(12)
etc. The matrix on the right can be computed by recursing into the hierarchy of cells generated by regular subdivision of cells.

(iii) The last case is where \( K \) is active on mesh \( g' \) but not mesh \( g \). In that case, we have to express basis function \( \varphi^j_{g'} \) in terms of the basis functions defined on the children of cell \( K \), rather than \( \varphi^j_g \) as before. This of course works in exactly the same way. If the children of \( K \) are active on mesh \( g \), then a similar procedure leads to the following expression:
\[
F_i|_K = \sum_j \left\{ \int_K f(r) \varphi^i_g(r) \varphi^j_{g'}(r) \, dx \right\} \Phi^j_{g'},
\]
\[
= \sum_j \left\{ \sum_{c=1}^{2^{\dim K}} \int_{K_c} f(r) \varphi^i_g(r) B^j_c \varphi^j_{g'}(r) \, d^3r \right\} \Phi^j_{g'},
\]
In matrix notation, this expression now reads as
\[
F_i|_K = \sum_{c=1}^{2^{\dim K}} F_i|_{K_c}, \quad F_i|_{K_c} = \sum_j \sum_l M^{ij}_{K_c} B^j_c \Phi^j_{g'}, = \sum_j (M_{K_c} B^T_c)^{ij} \Phi^j_{g'},
\]
and, correspondingly, for cases where cell \( K \) is refined more than once on mesh \( g \):
\[
F_i|_{K_{cc'}} = \sum_j (M_{K_{cc'}} B^T_c B^T_c)^{ij} \Phi^j_{g'},
\]
(13)
or
\[
F_i|_{K_{cc'c'}} = \sum_j (M_{K_{cc'c''}} B^T_c B^T_c B^T_c)^{ij} \Phi^j_{g'},
\]
(14)
etc. In other words, the process works in exactly the same way as before,
except that we have to take the transpose of the prolongation matrices and need to multiply it to the mass matrix from the other side.

The expressions for cases (ii) and (iii) can be understood as repeatedly interpolating either the left or right basis functions in the scalar product $\langle f \varphi^i_g, \varphi^j_g \rangle_K$ onto child cells, and then finally forming the inner product (the mass matrix) on the final cell. To make the symmetry in these cases more obvious, we can write them as follows: for case (ii), we have

$$F_i|_{K_{c^d \cdots c^{(k)}}} = \sum_j [B_cB_{c^d} \cdots B_{c^{(k)}}M_{K_{c^d \cdots c^{(k)}}}]^{ij}\Phi^j_{g'},$$  

(15)

whereas for case (iii) we get

$$F_i|_{K_{c^d \cdots c^{(k)}}} = \sum_j [(B_cB_{c^d} \cdots B_{c^{(k)}}M_{K_{c^d \cdots c^{(k)}}})^T]^{ij}\Phi^j_{g'},$$  

(16)

Although this process of computing terms that involve quantities defined on different meshes may appear cumbersome, it is easily implemented in a rather straightforward way using recursion and using the hierarchical structure of the meshes we use.

4 Numerical results

In this section we present a number of numerical experiments that illustrate the approach we have described above. All computations were performed with a program based on the deal.II finite element library [6, 7], an Open Source library that supports fully adaptive 1-D, 2-D, and 3-D meshes, a variety of different finite element types, and a large collection of classes supporting general finite element computations. An earlier version of the program used here has been extensively documented and will be distributed with next release of deal.II as step #28 of the tutorial.\footnote{Use of step-28 as the starting point for numerical experiments by others is possible and encouraged under the Open Source license of deal.II. We kindly request that all publications resulting from its use acknowledge the original authors, Yaqi Wang and Wolfgang Bangerth, as well as the creators of the deal.II library [6]. The step-28 program differs from the one here only in that it does not use the Chebychev acceleration and it builds the cross-grid terms discussed in Section 3.4 on the fly in each iteration, whereas our final version caches some information for performance reasons.}

The test cases we will discuss below are: (1) the 2-D version of the 2-group IAEA PWR problem [15], where data is given as piece-wise constant for each
fuel assembly, (2) the C4 configuration of the 2-D 2-group pin-by-pin design of the OECD mixed oxide fuel benchmark [30], where the data is piece-wise constant at the pin cell level, and (3) a 3-D 7-group problem with cross sections representative of UOX and MOX fuels and partial rod insertion.

In the following sections, we will compare results obtained on uniform and adaptively refined meshes with respect to (i) the accuracy of the eigenvalue $k_{\text{eff}}$, and (ii) the accuracy in the core power peaking factor. Here, we define the core power peaking factor as the maximum power produced in each initial mesh cell $K_i$, i.e.,

$$PPF = \max_{K_i} \frac{1}{|\Omega|} \int_{\Omega} \sum_{g=1}^{G} \Sigma_{f,g} \phi_g.$$ 

(17)

It may seem that this definition, based on the initial cells favors uniform coarse meshes, in which errors may partially cancel out in through averaging. However, as we will show, this is in fact not the case, and adaptivity yields significantly more accurate results. Note that the initial meshes we will consider are based on the problem geometry (e.g., the core layout of fuel assemblies) and are typically used to output computed data such as power per assembly or quarter assembly; therefore, the power peaking factor above is not defined on an arbitrary set of cells but rather the fixed set of reactor blocks we are actually interested in.

4.1 Example 1: 2-D, two energy groups IAEA PWR benchmark

In this test, we use the 2-D version of the IAEA-PWR benchmark [15]. The 2-D geometry consist of one quarter of a cross-sectional cut of the 3-D geometry at a height of $z = 200$ cm, in which four control rods are inserted. The initial mesh consists of square cells of 10 cm width (i.e., four cells per fuel assembly). We will present results obtained with uniform and adaptive refinement, for Lagrange elements of polynomial order 1 through 4. Fig. 1 shows the solutions for the two energy groups. It clearly shows that the solution for the fast group is significantly smoother than that for the thermal group, as well as the effect of the reflector layer on the outer rim of the reactor.

Fig. 2 shows the error in $k_{\text{eff}}$ as a function of the CPU time required to compute it to a certain accuracy. For practical purposes, the eigenvalue must be obtained with at least 5 digits of accuracy (1 pcm) though for this particular case it is easy to approximate it to much higher accuracy. We computed the error in $k_{\text{eff}}$ using a reference value obtained by extrapolating from a sequence of globally refined meshes using cubic polynomials. Comparing linear finite elements ($Q_1$) results, we observe that adaptive mesh refinement becomes competitive against uniform meshes in the 1-pcm range; for higher accuracy, e.g. 0.1 pcm, adaptive refinement with $Q_1$ elements is clearly superior than
Fig. 1. *Example 1: Solutions after 9 adaptive refinement iterations, fast group (left), thermal group (right), using quadratic finite elements.*

Fig. 2. *Example 1: Accuracy in the eigenvalue $k_{\text{eff}}$ as a function of CPU time for uniform and adaptive mesh refinement, for linear, quadratic, cubic, and quartic elements.*

uniform mesh refinement. For higher order elements, adaptive refinement becomes competitive in the 0.1-pcm range as well.

Since the eigenvalue is a global quantity, it may not be too surprising to see that the benefits of adaptation are modest. Consequently, we also compare adaptive and uniform refinements for the core power peaking factor introduced above, a local quantity. Fig. 3 (left) shows the convergence in the peaking factor between the various refinement techniques as a function of CPU time. It is obvious that adaptive linear finite elements provide 1% accuracy five times faster than uniform refinement, and a 0.01%-accuracy 20 times faster; for an accuracy of 0.01%, adaptive $Q_1$ elements are comparable to uniform $Q_2$ elements. Similarly, for higher finite elements, there is always a significant gain in accuracy of the power peaking factor when using mesh adaptation over uniform refinement. Similar gains can also be seen in Fig. 3 (right), where the convergence in the core power peaking factor is shown a function of the number of unknowns.
Fig. 3. Example 1: Accuracy in the core power peaking factor as a function of CPU time (left) and the number of unknowns (right) for uniform and adaptive mesh refinement, for linear, quadratic, and cubic elements.

Meshes for $Q_1$ elements after no, 2, 4 and 6 adaptive refinements are shown in Fig. 4. From these images, it is clear that adaptive mesh refinement provides better results than uniform meshes: Refinement occurs in regions of steep flux variations leading to a better numerical approximation, whereas regions of smooth solution do not require the same level of refinement. It is also worth mentioning that the same accuracy as obtained with $Q_1$ elements after six adaptive refinements is already reached with two refinements and $Q_2$ elements (see Fig. 3).

4.2 Example 2: 2-D, two energy groups, OECD-L-336 fuel assembly benchmark

As a second example, we consider a 2-D 2-group C4 configuration (semi-reflected MOX checkerboard) of the OECD/NEACRP L-336 mixed oxide fuel benchmark problem issued by the Organization for Economic Cooperation and Development/Nuclear Energy Agency Committee on Reactor Physics.
Fig. 5. Example 2: Solutions of the 2-D 2-group multigroup neutron diffusion equations (top row; left: fast energy group $g = 1$, right: thermal energy group $g = 2$) and meshes after 2 and 4 adaptive refinement cycles (bottom row, left two panels: fast group; bottom row, right two panels: thermal group).

We compare solutions using $Q_p$ finite elements of polynomial degrees $p = 1$ through 6. In the initial mesh, each initial computational cell is equal to the homogenous square fuel pin cell (i.e., an array of 34 $\times$ 34 cells). The meshes used for the fast and thermal energy groups are then refined according to the criteria discussed in Section 3.

We show the solutions of both groups as well as a succession of meshes used for the computations using $Q_1$ (i.e., bilinear) elements in Fig. 5. Even more so than in the first example, it is apparent that the solutions have entirely different smoothness properties: The fast group is relatively smooth, with only little dimples at the location of the guide tubes, while the thermal group shows large variations in neutron fluxes depending on the local neighborhood. On the other hand, the fast group flux is larger in magnitude by almost one order of magnitude. Thus, normalizing the group error estimators by the flux of each group proved to be important as it allows for an appropriate selection of the spatial zones and energy groups which require refinement, as can be seen by the level of refinement performed on the thermal flux in Fig. 5.

In order to investigate if adaptive mesh refinement is indeed better than uniform meshes, the left panel of Fig. 6 shows a comparison of the convergence
Fig. 6. Example 2: Comparison of accuracy reached with a given number of unknowns for uniform and adaptive mesh refinement, for linear, quadratic, and cubic elements (left). Comparison of accuracy achieved with adaptive mesh refinement for elements of degree 1 through 6 (right).

history for Q₁, Q₂, and Q₃ elements using both adaptive and uniform refinement. As can be seen from the figure, depending on the desired accuracy, adaptive meshes can achieve the same accuracy with up to 2–3 times fewer degrees of freedom as required for uniformly refined meshes. An alternative viewpoint is that adaptivity produces meshes on which the error is smaller by a factor of up to 10 for the same number of unknowns as on uniformly refined meshes.

The right panel of Figure 6 compares convergence of $k_{\text{eff}}$ on adaptive meshes for elements $Q_p$ of order $p = 1, 2, \ldots, 6$. As can be seen, the accuracy in the result can be increased significantly by choosing higher order finite elements up to $p = 4$. Beyond this, yet higher orders do not pay off any more. This should not be too unexpected given that the solution is rather rough at least for the thermal group.

4.3 Example 3: Three space dimensions, seven energy groups

To conclude our series of tests, we consider a 3-D core, 7-group calculation. The core layout, composed of homogenized fuel assemblies, is given in Fig. 7. We have chosen to consider two distinct types of fuel, representative of a UOX fuel (labeled Material 1, see Fig. 7) and MOX fuel (Material 2). This results in different and rapidly varying spatial flux distributions in the lower energy range. Additionally, a control rod (Material 3) is partially inserted in the UOX fuel. A reflector (Material 4) surrounds the fissile core radially and axially. For simplicity, a unique reflector composition is employed. The square fuel assemblies have a width of 21.504 cm, the active core height is 3.6 m, with an additional 20 cm of reflector at both top and bottom extremities, for a total height of 4 m. The control rods are inserted from the top in 120 cm of five fuel assemblies (the 240 cm below them are made of Material 1). The initial mesh is composed of right prisms with a square base of width 10.752 cm
and height of 20 cm.

Fig. 8 presents the group-7 solutions and adapted meshes for $Q_1$ and $Q_2$ finite elements. The plots are provided at refinement cycle 7 for $Q_1$ and cycle 2 for $Q_2$, for which the answers have approximately reached the same level of accuracy (see the discussion of Fig.s 10 and 11 below). Fig. 9 shows cross sections of the solution and adaptive meshes for groups 1 (fastest neutron group), 5, and 7 (slowest group), obtained using $Q_1$ and $Q_2$ elements. Again, the $Q_1$ results are shown at mesh iteration 7 whereas the $Q_2$ results are for mesh iteration 2, both computations yielding about similar accuracies on these meshes. As in the previous examples, the solution shows significant differences in smoothness between the energy groups and has appreciable variation in particular for the slower energy groups with steep gradients in MOX fuel assemblies (higher absorption) and in the reflector.

Fig.s 10 and 11 show convergence of $k_{\text{eff}}$ and the core power peaking factor under mesh refinement. This test case, being three-dimensional and with a significant number of energy groups, is complicated enough that we were not able to compute $k_{\text{eff}}$ or the core power peaking factor to enough digits to allow us to compute convergence rates; all plots therefore only show actual values of these quantities (rather than errors) though convergence is apparent visually and a qualitative comparison of different methods is obviously possible. As in previous experiments, linear finite elements performed modestly better with adaptive refinement regarding the accuracy in the eigenvalue (significant gains in memory but comparable CPU times); for a local quantity, such as the core power peaking factor, linear finite elements on adapted meshes are clearly superior than on uniform meshes, with memory and CPU time savings of about one order of magnitude for a given accuracy. For quadratic finite elements, the results are even more favorable: Within a couple of mesh adaptations
the power peaking factor appears converged, whereas the results obtained from $Q_2$ elements with one level of uniform refinement are still inaccurate. By extrapolating the results from the $Q_2$ initial and once-uniformly refined meshes, we can estimate that the twice-uniformly mesh would give results of the same accuracy level as the solution obtained from two mesh adaptations, but at a memory and CPU cost about 100 hundred times larger; we have not been able to verify this in actual computations due to the excessive memory requirements of a twice globally refined mesh. As shown by the figure, $Q_3$ elements yield even higher accuracy at lower cost.

Finally, Fig. 12 shows the profile of the group-7 solution along the line $y = 39$ cm, $z = 200$ cm for several refinement cycles. The reference result utilizes $Q_2$ elements with 3 adaptive refinement cycles. From the left pane we infer that for $Q_2$ elements, after 2 adaptive cycles agreement with the reference solution is excellent. On the other hand, from the right pane we can conclude that for $Q_1$ elements even after 6 adaptive cycle some discrepancies remain.

5 Conclusions and outlook

In this contribution, we have presented an adaptive finite element algorithm for the multigroup diffusion approximation to the radiative transfer equation. It uses different meshes for each of the energy groups to resolve the smoothness properties of the solutions in each group. This approach is efficient because for this set of equations the smoothness of solution components various greatly
between energy groups, due to the significant variation in material parameters with energy.

Through a sequence of numerical experiments we have shown that the proposed algorithm performs better than uniformly refined meshes – in the most complicated three-dimensional case by orders of magnitude – and is therefore able to solve problems to accuracies previously impossible or to solve problems to the required accuracy for the first time at all.

Although we believe that the methods proposed here are a significant step
Fig. 10. *Example 3: Comparison of the eigenvalue $k_{\text{eff}}$ as a function of the number of unknowns (left) and CPU time (right) for uniform and adaptive mesh refinement, for linear, quadratic, and cubic elements.*

Fig. 11. *Example 3: Comparison of the core power peaking factor as a function of the number of unknowns (left) and CPU time (right) for uniform and adaptive mesh refinement, for linear, quadratic, and cubic elements.*

Fig. 12. *Example 3: Flux profile $\phi_7(x, y = 39, z = 200)$. Left: Solution in adaptive refinement cycles 0, 1, 2, and 3 using $Q_2$ elements. Right: Solution in adaptive refinement cycles 2, 4, 6, using $Q_1$ elements and cycle 3 using $Q_2$ elements.*

forward, we are also of the opinion that it can be significantly extended and improved. Among these possibilities are:

- When implemented on a parallel computer with distributed memory, meshes have to be replicated on each machine anyway. This removes the additional
(modest) memory requirements we currently impose on our computations by keeping $G$ instead of only one mesh. The algorithms discussed in the current paper may therefore be even more efficient if implemented on distributed memory machines. The drawback of parallel implementations is, of course, that the eigensolver currently loops over energy groups in a sequential fashion, solving each one in turn.

- The choice of importance factor $z_{a,K}$ in Section 3.3 is heuristic. A proper importance factor can be derived by solving a dual problem [9] that computes the influence of a given cell $K$ for a given group $g$ on a target functional, for example the accuracy of the eigenvalue $k_{\text{eff}}$. For nonlinear problems like the current eigenvalue problem, the solution of such a dual problem only amounts to one additional power iteration step; nevertheless, the implementational burden would be significant. We believe that a properly computed weight factor should significantly increase the accuracy and reliability of adaptive approaches.
- Our approach of using several different grids can of course be applied also to some of the other approximations to the radiative transfer equation that result in a set of coupled partial differential equations. This includes, in particular, approaches like the $P_N$ and $SP_N$ methods.

Acknowledgements. Part of this research was funded through DOE grant DE-FG07-07ID14767. This publication is partly based on work supported by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

6 Appendix

The following table lists the material parameters (cross sections) used for the seven-group example discussed in Section 4.3. Cross sections for the other two examples can be found in the references cited in the respective sections.

| $g$ | $D^0$ | $\Sigma_t^0$ | $\Sigma_t^{-1}$ | $\Sigma_t^{-2}$ | $\Sigma_t^{-3}$ | $\Sigma_t^{-4}$ | $\Sigma_t^{-5}$ | $\Sigma_t^{-6}$ | $\Sigma_t^{-7}$ | $\Sigma_t^0$ | $\chi^0$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9645058</td>
<td>0.0676314</td>
<td>0.1020466</td>
<td>0.0620793</td>
<td>0.0000011</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0134820</td>
<td>0.5880000</td>
</tr>
<tr>
<td>2</td>
<td>0.9536659</td>
<td>0.0419533</td>
<td>0.0000000</td>
<td>0.3075751</td>
<td>0.0001802</td>
<td>0.0000003</td>
<td>0.0000003</td>
<td>0.0000003</td>
<td>0.0013607</td>
<td>0.0016030</td>
<td>0.4166000</td>
</tr>
<tr>
<td>3</td>
<td>0.6713391</td>
<td>0.0905403</td>
<td>0.0000000</td>
<td>0.0409798</td>
<td>0.0670154</td>
<td>0.0049246</td>
<td>0.0007649</td>
<td>0.0001458</td>
<td>0.0015424</td>
<td>0.0003400</td>
<td>0.0000000</td>
</tr>
<tr>
<td>4</td>
<td>0.6215841</td>
<td>0.2108010</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.337204</td>
<td>0.1242644</td>
<td>0.0184459</td>
<td>0.0304014</td>
<td>0.0090000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>5</td>
<td>0.7927936</td>
<td>0.1951705</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.225836</td>
<td>0.1551156</td>
<td>0.0176976</td>
<td>0.0290938</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>6</td>
<td>0.5278111</td>
<td>0.2476514</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0015182</td>
<td>0.3836842</td>
<td>0.1681179</td>
<td>0.1256622</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>7</td>
<td>0.2891938</td>
<td>0.2450183</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0437284</td>
<td>0.9076112</td>
<td>0.3530487</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>

Material 2

| $g$ | $D^0$ | $\Sigma_t^0$ | $\Sigma_t^{-1}$ | $\Sigma_t^{-2}$ | $\Sigma_t^{-3}$ | $\Sigma_t^{-4}$ | $\Sigma_t^{-5}$ | $\Sigma_t^{-6}$ | $\Sigma_t^{-7}$ | $\Sigma_t^0$ | $\chi^0$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9414476</td>
<td>0.0683040</td>
<td>0.1033918</td>
<td>0.0616757</td>
<td>0.000144</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0159816</td>
<td>0.5880000</td>
</tr>
<tr>
<td>2</td>
<td>0.9463810</td>
<td>0.0419533</td>
<td>0.0000000</td>
<td>0.3102656</td>
<td>0.0387188</td>
<td>0.0001399</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0015424</td>
<td>0.0003400</td>
<td>0.4166000</td>
</tr>
<tr>
<td>3</td>
<td>0.6588437</td>
<td>0.2108010</td>
<td>0.0000000</td>
<td>0.4160882</td>
<td>0.0669145</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0015424</td>
<td>0.0003400</td>
<td>0.0000000</td>
</tr>
<tr>
<td>4</td>
<td>0.5854027</td>
<td>0.1951705</td>
<td>0.0000000</td>
<td>0.3411193</td>
<td>0.1242644</td>
<td>0.0184459</td>
<td>0.0304014</td>
<td>0.0090000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>5</td>
<td>0.6296201</td>
<td>0.2476514</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0437284</td>
<td>0.9076112</td>
<td>0.3530487</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>
Material 3

<table>
<thead>
<tr>
<th>1</th>
<th>0.8996137</th>
<th>0.0667524</th>
<th>0.1087219</th>
<th>0.0611430</th>
<th>0.0002033</th>
<th>0.0000010</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9222381</td>
<td>0.0391507</td>
<td>0.0000000</td>
<td>0.3222889</td>
<td>0.0354180</td>
<td>0.0001643</td>
</tr>
<tr>
<td>3</td>
<td>0.6232103</td>
<td>0.0894575</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.4454075</td>
<td>0.0611758</td>
</tr>
<tr>
<td>4</td>
<td>0.5665006</td>
<td>0.2233427</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.3650651</td>
</tr>
<tr>
<td>5</td>
<td>0.7291666</td>
<td>0.2249611</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.3650651</td>
</tr>
<tr>
<td>6</td>
<td>0.4990789</td>
<td>0.2915378</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.3650651</td>
</tr>
<tr>
<td>7</td>
<td>0.2788687</td>
<td>0.3157843</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.3650651</td>
</tr>
</tbody>
</table>

Material 4

<table>
<thead>
<tr>
<th>1</th>
<th>2.6455026</th>
<th>0.0598000</th>
<th>0.0662000</th>
<th>0.0591000</th>
<th>0.0002830</th>
<th>0.0000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.1376564</td>
<td>0.0530000</td>
<td>0.0000000</td>
<td>0.2400000</td>
<td>0.0524000</td>
<td>0.0002500</td>
</tr>
<tr>
<td>3</td>
<td>1.1737089</td>
<td>0.1010000</td>
<td>0.0000000</td>
<td>0.1830000</td>
<td>0.0924000</td>
<td>0.0004900</td>
</tr>
<tr>
<td>4</td>
<td>1.1862396</td>
<td>0.2021000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0780000</td>
<td>0.0108000</td>
</tr>
<tr>
<td>5</td>
<td>0.9980040</td>
<td>0.2343000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0030373</td>
<td>0.0097000</td>
</tr>
<tr>
<td>6</td>
<td>0.5889282</td>
<td>0.2490000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0030373</td>
<td>0.0097000</td>
</tr>
<tr>
<td>7</td>
<td>0.2849003</td>
<td>0.0700000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0030373</td>
<td>0.0097000</td>
</tr>
</tbody>
</table>

References


