Discontinuous Diffusion Synthetic Acceleration for S_n Transport on 2D Arbitrary Polygonal Meshes

Abstract

In this paper, a Diffusion Synthetic Acceleration (DSA) technique for S_n radiation transport is developed for on arbitrary polygonal grids using a Piece-Wise Linear Discontinuous (PWLD) finite element approximation. The discretization of the DSA equations uses an Interior Penalty technique, as in classically derived from a standard discontinuous finite element technique for the standard form of the diffusion equation. The MIP technique yields a system of linear equations that is Symmetric Positive Definite (SPD). Thus, solution techniques such as Preconditioned Conjugate Gradient (PCG) can be effectively employed. This results in an efficient DSA preconditioner for arbitrary polygonal meshes. Such grids (which include triangular and tetrahedral meshes as a subset) can be used to model complex objects; they can also be advantageously employed with locally refined spatial grids without the need to deal with "hanging nodes". Fourier analyses are performed for the MIP-DSA formulation using PWLD finite elements, and we show that this scheme is always stable and effective at reducing the spectral radius for iterative transport solves, even for grids with high-aspect ratio cells. Numerical results are presented for different grids (quadrilateral, hexagonal, polygonal, and rectangular with local mesh adaptation). Algebraic MultiGrid (AMG) and Symmetric Gauss-Seidel (SGS) are employed as conjugate gradient preconditioners for the MIP system. AMG is shown to be significantly more efficient than SGS.

1 Introduction

In this paper, we present a Diffusion Synthetic Acceleration (DSA) scheme that is fully compatible with the Piece-Wise Linear Discontinuous (PWLD) finite element discretization of the transport equation on arbitrary polygonal cells.

Arbitrary polygonal (polyhedral in 3D) cells can advantageously be employed, especially in the context of spatial discretizations based on discontinuous finite elements (DFE), for the following reasons: polygonal grids

1. may allow for a reduced numbers of unknowns and
2. can provide a natural transition for locally adapted meshes.

To illustrate these two points, first consider a hexagonal cell. Employing a PWLD discretization, such a cell possesses six unknowns. Using alternate DFE discretizations that perform well in the thick diffusive limit, such as linear discontinuous on triangles and bilinear discontinuous on quadrangles, the same hexagonal cell could be split into two quadrangles (for a total of 8 unknowns), two triangles and one quadrangle (10 unknowns), or four triangles (12 unknowns). By inserting an extra point inside the cell, the hexagon could also be divided into three quadrangles (12 unknowns), four quadrangles (16 unknowns) or six triangles (18 unknowns). A similar reasoning can be applied to any n-polygon. Arbitrary polygonal grids can also handle locally refined meshes in a natural manner. The example given in Figure 1 is typical of simulations performed with Adaptive Mesh Refinement (AMR). Solvers based on arbitrary polygonal cells can easily handle cells with various numbers of edges as follows: on Figure 1, the left cell is actually interpreted as a (degenerate) pentagon whereas the two cells on the right are quadrilaterals. PWLD spatial discretization can handle locally adapted meshes without any special treatment or further approximation of the coupling between cells. Moreover, the finite element representation of the refined side is piece-wise linear. Had the finite element representation been, for instance, bilinear discontinuous (BLD), the unrefined cell would have remained a rectangular with a linear (not piece-wise linear) representation of the solution along that side.

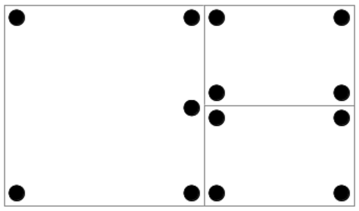


Figure 1: Adaptive mesh refinement grid with PWLD finite elements: left cell is a pentagon right cells are quadrangles.

Next, we recall the rationale for solving transport problems iteratively using diffusion acceleration schemes (preconditioners). Because analytical solutions are unavailable for most radiation transport problems of practical interest, one typically employs iterative techniques to solve the large system of equations that results from the spatial and angular discretizations of the transport equation. Standard iterative techniques for the first-order form of the discrete-ordinate (S_n) transport equation include Source Iteration (SI) and Krylov subspace techniques (usually GMRes [?]). For highly diffusive materials (i.e., with scattering ratios $c = \Sigma_s/\Sigma_t$ close to 1) and optically thick configurations (i.e., problems that are not leakage-dominated), these iterative techniques can become quite ineffective, requiring high iteration counts and possibly leading to false convergence. To mitigate these issues, SI and GMRes-based transport solves can be effectively accelerated (preconditioned) using Diffusion Synthetic Acceleration (DSA) [?, ?, ?, ?, ?, ?].

The spatial discretization of the DSA equations must be somewhat "consistent" with the one used for the S_n transport equations in order to yield unconditionally stable and efficient DSA schemes [?, ?, ?, ?, ?, ?]. However, the search for full consistency between the discretized transport equations and the discretized diffusion may not be computationally practical (especially for unstructured arbitrary meshes, [?]). For instance, Warsa, Wareing, and Morel [?] derived a fully consistent DSA scheme for linear discontinuous finite elements on unstructured tetrahedral meshes; their DSA scheme yields a P_1 system of equations that is computationally more expensive than partially consistent DSA schemes that are based upon discretizations of a standard diffusion equation. Several partially consistent schemes have been analyzed for discontinuous finite element (DFE) discretizations of the transport equation on unstructured meshes, for example, the modified-four-step (M4S) scheme [?], the Wareing-Larsen-Adams (WLA) scheme [?], and the Modified Interior Penalty (MIP) scheme [?].

To the authors' knowledge, no work is currently ongoing to adapt the M4S technique to polygonal meshes. This is very likely due to the fact that the M4S scheme does not yield a Symmetric Positive Definite (SPD) matrix and was found to be divergent for 3D tetrahedral meshes with linear discontinuous elements [?]. Recent work to develop a DSA scheme for polygonal cells has mainly focused on adapting the WLA scheme to polygonal meshes [?, ?]. The WLA scheme is a two-stage process, where first a diffusion solution is obtained using a *continuous* finite element discretization and then a *discontinuous* update is performed cell-by-cell in order to provide an appropriate discontinuous scalar flux correction to the DFE transport solver. In [?], the WLA scheme was found to be a stable and effective DSA technique, though its efficiency degraded as the problem became more optically thick and highly diffusive. In this paper, we extend the MIP technique to the PWLD discretization technique for arbitrary polygonal meshes. The MIP scheme is based on the standard Interior Penalty (IP) method for the discontinuous finite element discretization of diffusion equations. MIP was first derived in [?], where it was applied to triangular unstructured meshes (with locally adapted cells).

MIP does not suffer from the same problems as the WLA when the problem becomes optically thick and highly diffusive and, therefore, can be a viable alternative to the WLA scheme. Because MIP produces an SPD matrix, the authors of [?] have solved the resulting linear system using a Preconditioned Conjugate Gradient (PCG) technique (with Symmetric Gauss-Seidel, SGS, as preconditioner). In this paper, we also analyze the effectiveness of algebraic multigrid methods (AMG) [?, ?] to precondition the MIP-DSA diffusion solve and we compare AMG with PCG.

The remainder of this paper is organized as follows. In Section 2, we briefly review the PWLD discontinuous finite element discretization and the iterative solution techniques applied to the S_n transport equation. The MIP-DSA scheme is extended to the PWLD discretization for arbitrary polygons in Section 3. In Section 4, we introduce the Algebraic MultiGrid (AMG) approaches used here, which are based on the ML package of Trilinos [?] and the AGMG (AGgregation-based algebraic MultiGrid) technique of [?, ?, ?, ?]. In Section 5, we present a Fourier analysis for the MIP-DSA scheme discretized with PWLD, and we compare the different preconditioned CG approaches. Conclusions are given in Section 6.

2 Discretization and Solution Techniques for the S_n Transport Equation

In this Section, we review the S_n transport equation and the iterative solution techniques typically employed to solve it. We then describe the PWLD discontinuous spatial discretization for the transport equation with an emphasis on arbitrary polygonal/polyhedral grids.

2.1 The S_n Transport Equations

Given an angular quadrature set $\{\mathbf{\Omega}_m, w_m\}_{1\le m\le M}$, the one-group S_n transport equation with isotropic source and scattering is:

$$(\mathbf{\Omega}_m \cdot \mathbf{\nabla} + \Sigma_t(\mathbf{r}))\,\psi_m(\mathbf{r}) = \frac{1}{4\pi}\left(\Sigma_s(\mathbf{r})\phi(\mathbf{r}) + S(\mathbf{r})\right), \quad \text{for } \mathbf{r} \in \mathcal{D},\ 1 \le m \le M, \tag{1}$$

with $\psi_m(\mathbf{r}) = \psi(\mathbf{r}, \mathbf{\Omega}_m)$ the angular flux at position $\mathbf{r}$ in direction $\mathbf{\Omega}_m$, Σ_t and Σ_s the total and scattering cross sections, respectively, and $\mathcal{D}$ the spatial domain. The scalar flux is defined and numerically evaluated as follows:

$$\phi(\mathbf{r}) \equiv \int_{4\pi} \psi(\mathbf{r}, \mathbf{\Omega})d\mathbf{\Omega} \approx \sum_{m=1}^{M} w_m \psi_m(\mathbf{r}). \tag{2}$$

For brevity, we assume only incoming boundary conditions, that is, $\psi_m(\mathbf{r}_b) = \psi_m^{inc}(\mathbf{r}_b)$ for any point on the inflow boundary: $\mathbf{r}_b \in \partial\mathcal{D}_m^- = \{\partial\mathcal{D} \text{ such that } \mathbf{\Omega}_m \cdot \boldsymbol{n}_b < 0\}$, with $\boldsymbol{n}_b = \boldsymbol{n}(\mathbf{r}_b)$ the outward unit normal vector at $\mathbf{r}_b$. Equation (1) can be written in a compact form using operators:

$$\boldsymbol{L}\Psi = \boldsymbol{M}\boldsymbol{\Sigma}\Phi + S \equiv q, \tag{3}$$

$$\Phi = \boldsymbol{D}\Psi, \tag{4}$$

where Ψ is the vector of angular fluxes, Φ the scalar flux, q is the total (scattering+external) source, $\boldsymbol{L}$ is the streaming operator, $\boldsymbol{\Sigma}$ is the scattering matrix, $\boldsymbol{M}$ is the moment-to-direction operator, and $\boldsymbol{D}$ is the direction-to-moment operator. $\boldsymbol{L} = diag(\boldsymbol{L}_1, \ldots, \boldsymbol{L}_m, \ldots, \boldsymbol{L}_M)$ is a block-diagonal operator; given a total source q, one can solve independently for the resulting angular fluxes in all directions. The action of $\boldsymbol{L}^{-1}$ is often referred to as a transport sweep when discontinuous spatial approximations are employed because, for any direction $\mathbf{\Omega}_m$, the action of $\boldsymbol{L}_m^{-1}$ can be obtained by traversing the mesh (i.e., sweeping) in a specific ordering of the cells, thus requiring that a small linear system of equations be solved cell by cell. The order in which the elements are solved constitutes the graph of the sweep; for brevity and since this is not the focus of this article, we do not expand on situations where the graph presents dependencies (cycles). In such a case, these dependencies can either be lagged within the iterative procedure [?] or the solution vector consisting of the scalar flux is augmented by the angular flux unknowns that cause the cycle [?].

2.2 Solution Techniques

Equations (3) and (4) can be solved using the Source Iteration (SI) method (a stationary iterative technique also known as Richardson iteration). The ℓ^{th} iteration of the SI technique is given by :

$$\Phi^{(\ell+1)} = \boldsymbol{D}\boldsymbol{L}^{-1}\left(\boldsymbol{M}\boldsymbol{\Sigma}\Phi^{(\ell)} + S\right). \tag{5}$$

Alternatively, a subspace Krylov method (usually GMRes) can be employed to solve the linear system of equations, recast as follows:

$$\left(\boldsymbol{I} - \boldsymbol{D}\boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{\Sigma}\right)\Phi = \boldsymbol{D}\boldsymbol{L}^{-1}S \tag{6}$$

SI and GMRes-based transport solves require transport sweeps (the action of $\boldsymbol{L}^{-1}$). When the scattering ratio $c = \frac{\Sigma_s}{\Sigma_t}$ tends to one in optically thick domains, the number of SI and GMRes iterations can become large. To speed up convergence, a DSA preconditioner is needed; this is further discussed in Section 3.

2.3 Discontinuous Finite Element Discretization on Arbitrary Grids

To seek a DFE solution for the angular flux, the domain $\mathcal{D}$ is partitioned into arbitrary polygonal cells. For a given streaming direction $\boldsymbol{\Omega}_m$, the discontinuous finite element formulation, on a given spatial cell K, is given by:

$$-\int_K \left(\psi_m \boldsymbol{\Omega}_m \cdot \boldsymbol{\nabla} b_i + \Sigma_t \Psi_m b_i\right)\, d\mathbf{r} + \int_{\partial K^+} \boldsymbol{\Omega}_m \cdot \boldsymbol{n} \Psi_m b_i \, d\mathbf{r} = \int_K q\, b_i \, d\mathbf{r} + \int_{\partial K^-} |\boldsymbol{\Omega}_m \cdot \boldsymbol{n}| \psi_m^{\uparrow}\, b_i \, d\mathbf{r}, \tag{7}$$

where b_i represents a generic discontinuous finite element basis function, ∂K^- is the inflow face of element K, and ∂K^+ is the outflow face of element K. The angular flux values on an inflow face, denoted by $\psi_m^{\uparrow}$ in eq. (7), are taken from the upwind neighbor element of that face.

Next, we define the b_i basis functions for the PWLD discretization. First, we introduce the within-cell point c for any 2D polygonal cell. The coordinates of c are the weighted averages of the polygon's vertices:

$$u_c = \sum_{i=1}^{N_V} \alpha_i u_i, \tag{8}$$

where $u = x$ or y, N_V is the number of vertices for the cell under consideration, and the (positive) weights α_i are such that $\sum_{i=1}^{N_V} \alpha_i = 1$. The basis function at vertex i is defined by (see also [?]):

$$b_i(x,y) = t_i(x,y) + \alpha_i t_c(x,y), \tag{9}$$

where $t_i(x,y)$ is a linear function such that $t_i(x,y)$ is unity at vertex i and zero at vertices $i-1$, $i+1$, and c. The $t_c(x,y)$ function is a "tent" function in the interior of the cell; it is unity at the within-cell point c and zero at all of the vertices of the cell. The number of PWLD basis functions is equal to the number of vertices in the polygon (i.e., $1 \leq i \leq N_V$). In this paper, the arbitrary positive weights α_i are chosen to be equal to $\frac{1}{N_V}$. For example, on a quadrilateral cell, we employ $\alpha_i = \frac{1}{4}$. Finally, note that on triangular cells, the PWLD basis functions reduce to the standard Linear Discontinuous (LD) basis functions if ones chooses $\alpha_i = \frac{1}{3}$. Given the definition of the PWLD finite elements, it may seem complicated to build the elementary transport matrices on an arbitrary polygonal cell, but the construction of such matrices can be greatly simplified using the notion of "sub-cells", where a "sub-cell", in 2D, is defined as the triangular cell linking an edge of the polygonal cell to its within-cell point. Note that these "sub-cells" are never created as part of the polygonal grid. However, from a simple code implementation perspective, the computation of the elementary matrices of a polygonal cell is greatly simplified by looping over its sub-cells.

3 Diffusion Synthetic Acceleration (DSA) on Arbitrary Polygonal Cells

3.1 DSA Solution Principle

As noted earlier, standard iterative techniques applied to transport solves can be slowly converging in thick diffusive configurations. A DSA scheme must be employed to accelerate their convergence. The idea behind synthetic acceleration is that the error between the (yet unknown) transport solution and the current iterate can be estimated from a computationally less expensive process, yielding a corrective term to be added to the current iterate in order to improve the next iterate. In DSA, a corrective scalar flux contribution, $\delta\Phi$, is sought through the following diffusion equation,

$$\boldsymbol{A}\,\delta\Phi = \boldsymbol{\Sigma}\left(\Phi^{(\ell+1/2)} - \Phi^{(\ell)}\right). \tag{10}$$

where the source term is a scattering term due to the difference between the previous iterate scalar flux $\Phi^{(\ell)}$ and the newest scalar flux, $\Phi^{(\ell+1/2)}$, obtained after a single transport sweep. The next scalar flux iterate is obtained by adding the scalar flux correction to the latest scalar flux, yielding:

$$\Phi^{(\ell+1)} = \Phi^{(\ell+1/2)} + \delta\Phi. \tag{11}$$

$\boldsymbol{A}$ is the diffusion matrix of the DSA scheme. Ideally, $\boldsymbol{A}$ should be SPD (such that efficient iterative techniques can be employed in its linear solve) and easy to form (even on arbitrary grids).

3.2 Modified Interior Penalty DSA Scheme for Polygons

Here, we discuss the Modified Interior Penalty method for the diffusion equation for arbitrary polygons as a DSA scheme. In its discretization, this DSA scheme employs the same discontinuous finite elements used in the discretization of the transport operator. MIP as a DSA scheme has been shown to be always stable for isotropic scattering on triangular cells [?]. The MIP diffusion solve is based on the standard Interior Penalty (IP) method [?]. Consider the following diffusion update equation:

$$(-\boldsymbol{\nabla}\cdot \mathrm{D}\boldsymbol{\nabla} + \Sigma_a)\,\delta\phi = Q_0 \qquad \text{for } \mathbf{r}\in\mathcal{D} \tag{12}$$

$$\frac{1}{4}\delta\phi - \frac{1}{2}\mathrm{D}\partial_n\delta\phi = 0 \qquad \text{for } \mathbf{r}\in\partial\mathcal{D} \tag{13}$$

where D is the diffusion coefficient and Σ_a is the absorption cross section. For DSA, the volumetric source term is given by the scattering due to the difference in scalar flux between two iterations, $Q_0 = \Sigma_s\left(\Phi^{(\ell+1/2)} - \Phi^{(\ell)}\right)$, and the known incoming angular flux from the transport problem translates into the diffusion vacuum boundary condition given in Equation (13), where $\partial_n = \boldsymbol{n}\cdot\boldsymbol{\nabla}$ and $\boldsymbol{n}$ is the outer normal unit vector on the domain's boundary, $\partial\mathcal{D}$. The weak form for the MIP-DSA equation is given by:

$$a(\delta\phi,\phi^*) = l(\phi^*) \tag{14}$$

with the bilinear (matrix) form:

$$\begin{aligned} a(\delta\phi,\phi^*) = {} & (\Sigma_a\delta\phi,\phi^*)_{\mathcal{D}} + (\mathrm{D}\boldsymbol{\nabla}\delta\phi,\boldsymbol{\nabla}\phi^*)_{\mathcal{D}} + \left(\kappa_e^{MIP}[\![\delta\phi]\!],[\![\phi^*]\!]\right)_{E_h^i} + \left([\![\delta\phi]\!],\{\!\{\mathrm{D}\partial_n\phi^*\}\!\}\right)_{E_h^i} + \\ & \left(\{\!\{\mathrm{D}\partial_n\delta\phi\}\!\},[\![\phi^*]\!]\right)_{E_h^i} + \left(\kappa_e^{MIP}\delta\phi,\phi^*\right)_{\partial\mathcal{D}^d} - \frac{1}{2}\left(\delta\phi,\mathrm{D}\partial_n\phi^*\right)_{\partial\mathcal{D}^d} - \frac{1}{2}\left(\mathrm{D}\partial_n\delta\phi,\phi^*\right)_{\partial\mathcal{D}^d} \end{aligned} \tag{15}$$

and the linear (right-hand-side) form

$$l(\phi^*) = (Q_0,\phi^*)_{\mathcal{D}}\ . \tag{16}$$

The notations used are as follows: the domain integral is split onto the element integrals, $(f,g)_{\mathcal{D}} = \sum_{K\in\mathbb{T}_h}(f,g)_K$, with the element integrals defined as $(f,g)_K = \int_K fg\ d\mathbf{r}$; and the integral over all interior edges is $(f,g)_{E_h^i} = \sum_{e\in E_h^i}(f,g)_e$, with the edge integrals defined as $(f,g)_e = \int_e fg\ ds$. In the above, $\mathbb{T}_h$ denotes the partition of domain $\mathcal{D}$ into non-overlapping elements K, E_h^i is the set of interior edges, $\delta\phi$ is the numerical solution for the diffusion correction, and ϕ^* is any test function. Any entry A_{ij} of matrix $\boldsymbol{A}$ is simply given by $A_{ij} = a(b_j, b_i)$. The resulting matrix $\boldsymbol{A}$ is SPD and thus can be solved using a preconditioned conjugate gradient technique. The jump and mean value of a variable u at the interface between two elements is defined as follows:

$$[\![u]\!] = u^+ - u^- \text{ and } \{\!\{u\}\!\} = \frac{u^+ + u^-}{2}, \tag{17}$$

respectively. The definition of the left/right values along a given edge e is

$$u^\pm = \lim_{s\to 0^\pm} u(\boldsymbol{r} + s\boldsymbol{n}_e), \tag{18}$$

where $\boldsymbol{n}_e$ is the normal unit vector associated with the given edge e (on the boundary $\boldsymbol{n}_e$ is the external normal). The MIP penalty coefficient is given by

$$\kappa_e^{MIP} = \max\left(\kappa_e^{IP}, \frac{1}{4}\right) \tag{19}$$

with:

$$\kappa_e^{IP} = \begin{cases} \dfrac{c}{2}\left(\dfrac{\mathrm{D}^+}{h_\perp^+} + \dfrac{\mathrm{D}^-}{h_\perp^-}\right) & \text{on interior edges, i.e., for } e \in E_h^i \\ c\dfrac{\mathrm{D}}{h_\perp} & \text{on boundary edges, i.e., for } e \in \partial\mathcal{D} \end{cases} \tag{20}$$

where c is a constant (chosen equal to 4 here) and $h_\perp$ is the length of the cell in the direction orthogonal to the edge e. When polygonal cells are employed, there is no simple way to compute $h_\perp$. To estimate $h_\perp$, we assume that the polygonal cells do not deviate significantly from regular polygons. In such cases, if the cell has an even number of edges, the orthogonal length equals two times the apothem, i.e., two times the segment between the midpoint of a side of the polygon and the center of this polygon $\left(\text{apothem} = 2 \times \frac{\text{area}}{\text{perimeter}}\right)$. If a cell has an odd number of edges, its orthogonal length is given by the apothem plus the circumradius, i.e., the radius of the circle circumscribed to the polygon $\left(\text{circumradius} = \sqrt{\frac{2\times\text{area}}{N_V \sin\left(\frac{2\pi}{N_V}\right)}}\right)$. A summary of the definitions used for $h_\perp$ for any polygon as a function of the number of vertices is given in Table 1.

Table 1: Orthogonal length $h_\perp$ for different polygonal types.

Number of vertices	3	4	> 4 and even	> 4 and odd
$h_\perp$	$2 \times \frac{\text{area}}{L_e}$	$\frac{\text{area}}{L_e}$	$4 \times \frac{\text{area}}{\text{perimeter}}$	$2 \times \frac{\text{area}}{\text{perimeter}} + \sqrt{\frac{2\times\text{area}}{N_V \sin\left(\frac{2\pi}{N_V}\right)}}$

4 MIP-DSA Solves Based on Algebraic Multigrid Techniques

4.1 AMG Principles

As mentioned above, a common way to solve a SPD system of equations is to use a conjugate gradient technique preconditioned with Symmetric Gauss-Seidel (PCG-SGS) or SSOR (PCG-SSOR); SGS is simply

SSOR with a damping factor of one. In this research, we will compare the calculation time and the number of iterations using conjugate gradient preconditioned with Symmetric Gauss-Seidel (PCG-SGS) to those of CG preconditioned with an algebraic multigrid method. PCG-SGS was chosen because little difference was noted when employing other damping factors for MIP-DSA on triangular grids [?]. An algebraic multigrid method was employed to precondition the Krylov solver for the even-parity finite element-spherical harmonics (FE-P_N) method; see, for instance, [?]. In [?], the use of AMG as a preconditioner resulted in a 60% reduction in the solution time compared to ILU(0) preconditioning and even more reduction compared to SSOR preconditioning. In our implementation, we have linked our code with the ML package [?] of the Trilinos library and with the AGMG library [?]. ML is a multigrid preconditioning package that uses a smoothed aggregation algebraic multigrid to build a preconditioner for a Krylov method. AGMG is an aggregation-based algebraic multigrid library written in Fortran 90. The first multigrid methods developed were geometric multigrid techniques, used as stand-alone solvers. In many applications, they achieve the so-called "textbook multigrid efficiency", i.e., "the solution to the governing system of equations [is attained] in a computational work that is a small multiple of the operation counts associated with discretizing the system" [?]. However, in many other applications, multigrid methods, and particularly algebraic multigrid methods, cannot achieve such efficiency, [?] and, in such cases, they are often used as preconditioner for Krylov subspace methods.

Before describing a general multigrid method, we recall the process using a two-grid method. Consider the following system

$$\boldsymbol{A}_f u_f = b_f, \tag{21}$$

defined on a fine grid Γ_f. The two-grid algorithm is as follows:

1. Perform ν_1 pre-smoothing iterations using a smoother (Jacobi, Gauss-Seidel, or ILU) and an initial guess u_0: $u = S^{\nu_1}(u_0, b_f)$
2. Compute the residual on the fine grid Γ_f and restrict it to the coarse grid Γ_c: $r_c = \boldsymbol{R}(b_f - \boldsymbol{A}_f u)$;
3. Solve the system on the coarse grid: $v = \boldsymbol{A}_c^{-1} r_c$;
4. Interpolate the coarse grid correction to the fine grid and add the correction to u: $u = u + \boldsymbol{P}v$;
5. Perform ν_2 post-smoothing iterations: $u = S^{\nu_2}(u, b_f)$.

When using AMG, the matrix $\boldsymbol{A}_c$ on the coarse grid is given by the Galerkin approximation:

$$\boldsymbol{A}_c = \boldsymbol{R}\boldsymbol{A}_f\boldsymbol{P}, \tag{22}$$

where $\boldsymbol{P}$ is a prolongation matrix and $\boldsymbol{R}$ is a restriction matrix. Generally, solving the system $\boldsymbol{A}_c v = r_c$ on the coarse grid is still quite expensive, therefore this step is recursively replaced by n_γ sequences of the two-grid method until the system can be efficiently inverted with a direct solver. When $\gamma = 1$, respectively $\gamma = 2$, the multigrid method is said to use a $V-$cycle, respectively a $W-$cycle. Figure 2 shows typical $V-$ and $W-$ cycles. In Figure 2, a dot represents a smoothing operation and a square a direct inversion; the grid transfer operators are symbolized by the line segments.

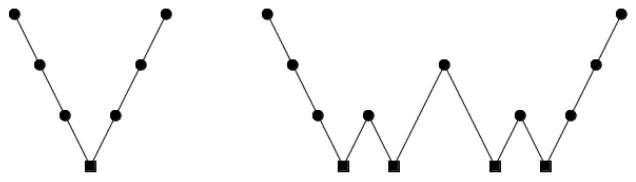


Figure 2: $V-$ and $W-$cycles.

The main difference between geometric and algebraic multigrid techniques lies in the method used to coarsen the grid. Algebraic multigrid methods only use the properties of the matrix. Among the algebraic multigrid methods, there are three main categories: the classical Ruge-Stueben AMG, the plain aggregation AMG, and the smoothed aggregation AMG. ML uses smoothed aggregation AMG and AGMG uses plain aggregation AMG. Next, we briefly explain the coarsening step in the ML and AGMG implementations. The coarsening step is the most important step because if the coarsening occurs too fast, convergence rates will decrease. However, if coarsening is too slow, more memory may be required to solve the problem.

4.2 The ML Package of Trilinos

When using a smoothed aggregation scheme, the smoothed interpolation operators $\boldsymbol{P}_k$ are the transpose of the coarsening operators $\boldsymbol{R}_k = \boldsymbol{P}_k^T$. Therefore, when the $\boldsymbol{P}_k$ matrices are built, the coarsening operator is also known. First, the graph of the matrix is constructed: if element (i,j) or (j,i) of the matrix is non-zero, an edge is built between the vertex i and the vertex j [?]. Second, the vertices are aggregated. When using ML on a single processor, two aggregation schemes can be used: the uncoupled scheme or the maximally independent sets (MIS) scheme. The uncoupled scheme tries to build aggregates of size 3^d where d is the dimension of the problem; its algorithm proceeds as follows [?]:

Step 1: As long as there are points not adjacent to an aggregate:

1. Choose a point which is not an adjacent to an aggregate. This point is a new root point.
2. Define a new aggregate as the root point and its neighbors

Step 2: Add all the points left to the existing aggregates or form new aggregates with them.

The MIS scheme used in ML applies the MIS algorithm [?] to the graph of the matrix $\boldsymbol{A}^2$. These two coarsening schemes use a fixed ratio of coarsening between levels. Once aggregation is done, a tentative prolongation matrix, $\tilde{\boldsymbol{P}}_k$ is constructed [?]. A example of $\tilde{\boldsymbol{P}}_k$ is given by:

$$\tilde{\boldsymbol{P}}_k(i,j) = \begin{cases} 1 & \text{if the } i^{th} \text{ point is contained in the } j^{th} \text{ aggregate} \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

This tentative prolongation operator could be used as is but smoothing it allows to have a more robust scheme. Let $\boldsymbol{S}_k$ be a smoother, for example damped Jacobi. Then, the prolongation matrix is given by:

$$\boldsymbol{P}_k = \boldsymbol{S}_k \tilde{\boldsymbol{P}}_k. \quad (24)$$

4.3 The AGMG Package

Unlike in ML, the prolongation operator in AGMG is not smoothed; this results in a cheaper setup and a decrease in memory requirements [?]. However, such a scheme could be less robust. To counteract this weakness, the aggregation scheme is more involved. Coarsening algorithms that control the size of the aggregates tend to produce a few badly shaped aggregates. Since the convergence of AMG is bounded by the worst aggregate, even a small number of badly shaped aggregates can have a huge impact on the convergence. In AGMG, the aggregation algorithm has as input the upper bound of the two-grid condition number. When the aggregates are constructed, their quality is checked. Obviously, this increases the cost of the coarsening and it is thus important that the coarsening is fast enough. Such an algorithm does not control the size of the aggregates, and, therefore, it is difficult to control the speed of coarsening. However, controlling the condition number rather than the coarsening speed can be an effective alternative approach. By monitoring the condition number, bad aggregates will not be created and, instead, a few aggregates below the target size may be generated. This does not affect the efficiency of the method in a noticeable way [?]. A simple way to create the aggregates would be to try to exhaust all of the possible combinations, compute their quality, and then choose the optimal coarsening. In practice, this would be too costly, and, in AGMG, the aggregation step is done by a few passes of a pairwise aggregation algorithm. Each pass aggregates the variables two by two to allow a simple computation of the aggregate quality and to keep the cost per iteration low. The advantage of controlling the condition number becomes even more important when a $K-$cycle, or Krylov-cycle, is used instead of the more common $V-$ or $W-$cycles. The difference between the $K-$cycle and the $V-$ or $W-$cycles is that $K-$cycles use a few iterations of a Krylov solver preconditioned by a coarser grid to solve the coarse grid problem in the two-grid algorithm [?]. The advantage of the $K-$cycle is an increased robustness compared to $V-$ and $W-$cycle. This scheme is nonlinear and requires, when the system is SPD, the use of flexible CG [?, ?, ?, ?] as the Krylov solver. Even when the condition number of the two-grid method is large, the convergence properties of the $K-$cycle can be independent of the number of levels [?]. The computational cost of the $K-$cycle is about the same than that of a $W-$cycle. If the number of unknowns does not decrease sufficiently from one level to the next, the $K-$cycle at one level is replaced by a $V-$cycle at this level. At that level, no Krylov solver is used in order to decrease the computational cost of the method.

5 Results

In this section, we present two series of results. First, Fourier analyses are carried out to analyze the performance of the MIP-DSA acceleration scheme for an homogeneous infinite medium meshed with rectangular cells and discretized with PWLD. The effects of the S_n order and the cell aspect ratio on the spectral radius of the iterative scheme are also studied. Second, the MIP-DSA technique is implemented in a 2D S_n code that uses arbitrary polygonal grids with a PWLD spatial discretization. Numerical examples employ several mesh types: arbitrary quadrilaterals; arbitrary polygons; a regular layout of hexagons/triangles/rectangles; and a grid that mimics adaptive mesh refinement performed on rectangular cells. The MIP-DSA diffusion solves are performed using various linear solvers: Conjugate Gradient (CG); Conjugate Gradient Preconditioned with Symmetric Gauss-Seidel (PCG-SGS); Conjugate Gradient Preconditioned with ML using Uncoupled aggregation (PCG-ML/U); Conjugate Gradient Preconditioned with ML using MIS aggregation (PCG-ML/MIS); and Conjugate Gradient Preconditioned with AGMG (AGMG).

5.1 Fourier Analyses

Fourier analysis is often performed to assess some of the properties of DSA-accelerated transport solves [?, ?, ?]. In a Fourier analysis, the eigenvalues of the iteration matrix are analyzed, assuming a Fourier ansatz for the error modes. Specifically, the iteration matrices for the SI and SI+DSA schemes are given by

$$\boldsymbol{DL^{-1}M\Sigma} \quad \text{and} \quad \boldsymbol{I} - (\boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{\Sigma})(\boldsymbol{I} - \boldsymbol{DL^{-1}M\Sigma}), \tag{25}$$

respectively. The error modes are of the form $\exp(i\boldsymbol{\Lambda}\cdot\boldsymbol{r})$ with the wave number $\boldsymbol{\Lambda} = [\lambda_x,\, \lambda_y]^T$. This expression for the error modes is inserted into the discretized equations and the spectral radius (largest eigenvalue in magnitude) of the iteration matrices are sought for $0 \leq \lambda_x \leq 2\pi/X$ and $0 \leq \lambda_y \leq 2\pi/Y$, where X and Y are the dimensions of the rectangular domain.

5.1.1 Spectral Radius as a Function of the S_n Order

This Fourier analysis is carried on a square cell ($X = Y$) using a Gauss-Legendre-Chebyshev (GLC) angular quadrature. The medium is homogeneous with a scattering ratio $c = 0.9999$; periodic boundary conditions are used. The results are plotted on Figure 3, where the $x-$axis is the mesh size in mean free paths and the $y-$axis is the spectral radius. There are four curves corresponding to different S_n orders: S_2, S_4, S_8, and S_{16}. From Figure 3, we conclude that MIP is stable for every cell size. The spectral radius is always less than 0.5, except for S_2 where it is about 0.7. In the fine mesh limit, the continuum results are recovered [?]. Notably, the spectral radius of SI+DSA using an S_2 quadrature in 2D undiscretized space is $0.5c$. Also, as the angular quadrature is refined, the standard limit value of $0.2247c$ for the spectral result is reached.

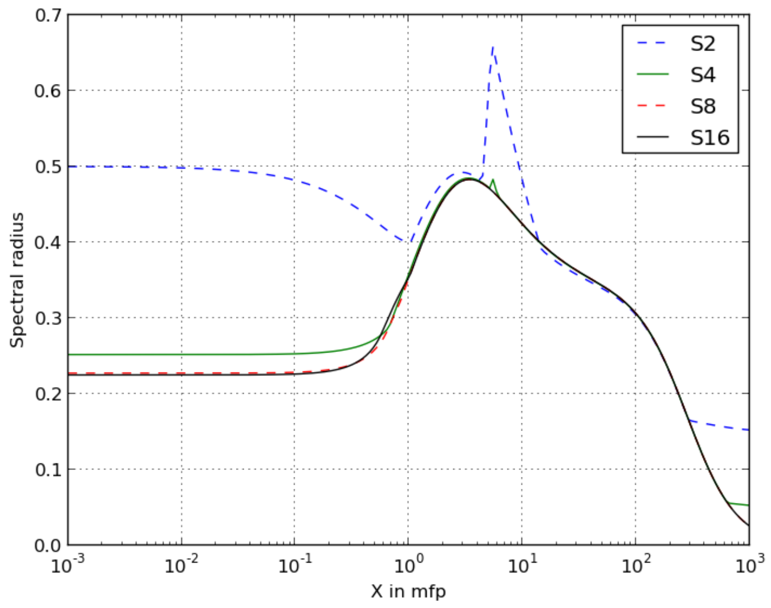


Figure 3: Spectral radius as a function of the mesh optical thickness for various S_n orders (Fourier analysis on a square cell).

5.1.2 Spectral Radius as a Function of the Cell Aspect Ratio

For this Fourier analysis, we use a S_{16} GLC quadrature. The medium is again homogeneous with $c = 0.9999$, and periodic boundary conditions apply. On Figure 4, the five curves correspond to the following cell aspect ratios: $\frac{Y}{X} = \frac{1}{16}$; $\frac{1}{4}$; 1; 4; 16; and 100. We note that the MIP-DSA scheme is stable for every aspect ratio tested, including an aspect ratio value of 100, and that the maximum spectral radius shows little sensitivity to the aspect ratio.

5.2 Performance of MIP-DSA Implemented in a PWLD S_n Transport Code

The MIP-DSA scheme has been implemented in a 2D S_n code that employs a PWLD discretization for arbitrary polygonal grids. Several test cases are presented.

5.2.1 Homogeneous Test Problem

We compare the different linear solvers employed for MIP-DSA using a homogeneous medium, $100cm\times100cm$, $\Sigma_t = 1cm^{-1}$ and $\Sigma_s = 0.999cm^{-1}$, with vacuum boundary conditions and a unit source of intensity $1cm^{-3}s^{-1}$. We use an S_8 angular quadrature. A relative tolerance of 10^{-8} is used for the Source Iteration solver while the MIP-DSA solvers employs a relative tolerance of 10^{-10}. The domain is discretized using two different meshes:

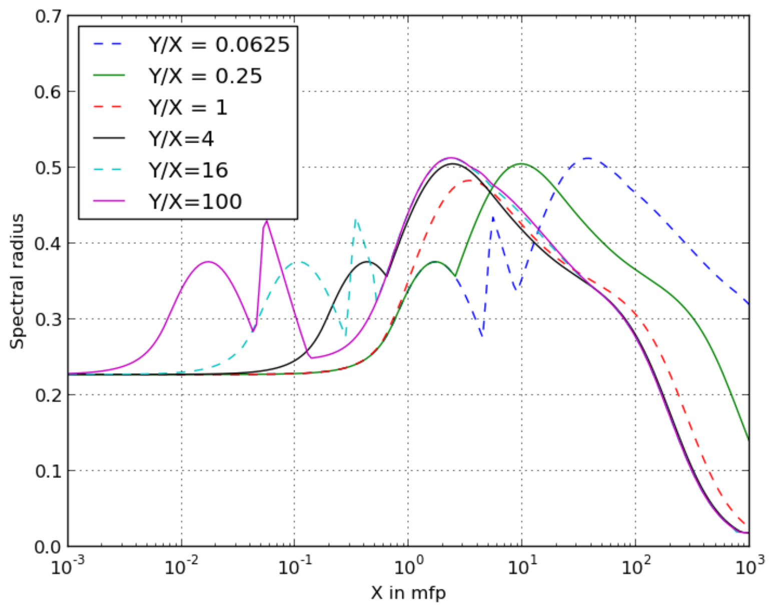
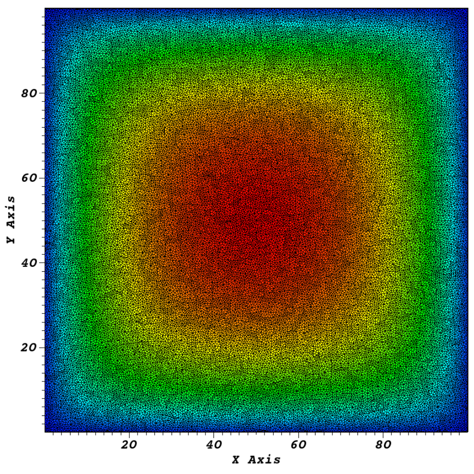


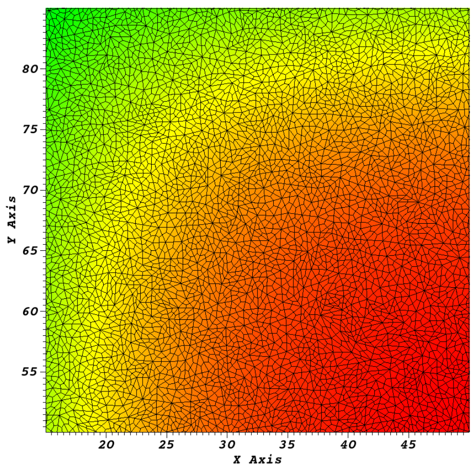
Figure 4: Spectral radius as a function of the mesh optical thickness for various cell aspect ratios (Fourier analysis carried out using an S_{16} angular quadrature).

1. a quadrilateral grid composed of 49263 quadrilateral cells (197,052 degrees of freedom);
2. a polygonal grid composed of 45,204 triangles, 823 quadrilaterals, 4,978 pentagons, 4,155 hexagons, 725 heptagons, and 24 octagons, for a total of 55,909 cells and 193,991 degrees of freedom. This example will allow us to test MIP and the different preconditioners on a mesh composed of different cell types.

The meshes and the numerical solutions are given in Figures 5 and 6. In Table 2, results obtained with the different linear solvers for MIP-DSA are compared for the quadrilateral grid. In Table 2, *SI iterations* is the number iterations of needed to solve the problem, *Precond init* is the time, in seconds, needed to initialize the preconditioner used by CG, *MIP calculation* is the total time, in seconds, spent solving DSA during the calculation, *CG iterations* is the total number of CG iterations used to solve MIP, and *Total calculation* is the time, in seconds, needed to solve the problem. We note that the accelerated transport solves only require 24 SI iterations, regardless of the linear solver employed in MIP-DSA (as expected). Preconditioning CG significantly reduces the number of CG iterations. We observe that algebraic multigrid processes, PGC-ML and AGMG, require about the same number of iterations (two orders of magnitude less than unpreconditioned CG). However, AMG is significantly faster than PCG-ML. We also note that PCG-SGS iteration is slower than one unpreconditioned CG iteration. Profiling of the code reveals that the bottleneck is the function *Ifpack_PointRelaxation::ApplyInverseSGS_FastCrsMatrix* of Trilinos. This function applies the forward and the backward substitutions required by SGS. It is unclear why these substitutions are so costly. Also note that SGS is employed as a pre- and post-smoother in the ML package of Trilinos and that the same function is once again the bottleneck of the method. The different linear solvers are compared for the polygonal grid in Table 3. We note that using different spatial cell types in the same grid does not affect the performance of MIP-DSA or that of its preconditioners.

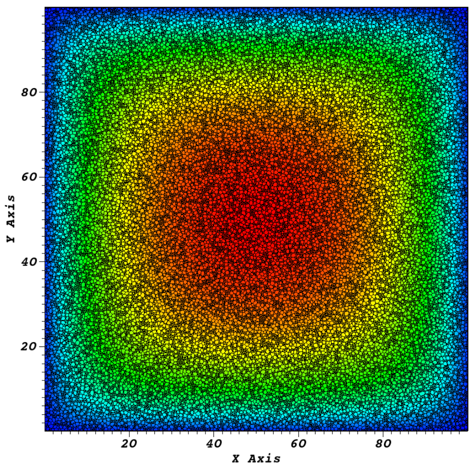


(a) Whole domain mesh

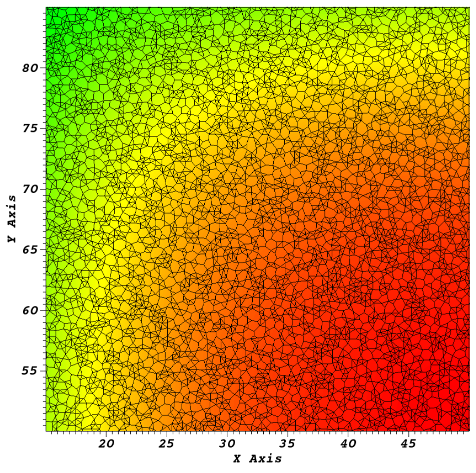


(b) Zoom

Figure 5: Grids and scalar flux solutions for the homogenous test problem using quadrilateral cells



(a) Whole domain mesh



(b) Zoom

Figure 6: Grids and scalar flux solutions for the homogenous test problem using arbitrary polygonal cells

Table 2: Comparison of different preconditioners for the homogeneous test problem (quadrilateral cells)

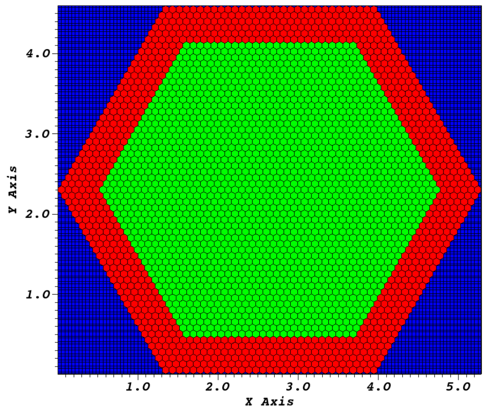
	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	7311	24	24	24	24	24
Precond init (s)	NA	NA	0.171358	1.8255	9.56078	0.332
MIP calculation (s)	NA	1095.7	1311.76	192.622	197.632	29.9727
CG iterations	NA	56649	17332	630	604	578
Total calculation (s)	39176.7	1264.98	1477.95	363.202	367.841	194.568

Table 3: Comparison of different preconditioners for the homogeneous test problem (polygonal cells)

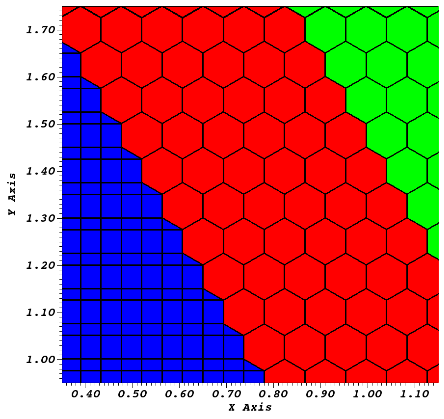
	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	7311	23	23	23	23	23
Precond init (s)	NA	NA	0.06388	1.73379	8.0426	0.388
MIP calculation (s)	NA	877.861	1263.31	198.63	191.989	31.242
CG iterations	NA	46262	16712	652	603	555
Total calculation (s)	42666.7	1060.53	1447.53	382.275	384.422	216.946

5.2.2 Test with Heterogeneous Media

In this example, a heterogeneous geometry with three materials is used. It is composed of 184 triangles, 3,720 quadrilaterals, and 2,791 regular hexagons of side $0.05cm$ for a total of 6,695 cells and 32,178 degrees of freedom (spatial unknowns per angular direction). The domain is $5.28275cm$ by $4.6cm$. Reflective boundary conditions are used. A S_{16} angular quadrature is used. The SI solver has a relative tolerance of 10^{-8}, and the relative tolerance for MIP-DSA is 10^{-10}. Figure 7 shows the problem geometry and the material properties are in given Table 4. The different linear solvers for MIP-DSA are compared in Table 5. The



(a) Whole domain mesh



(b) Zoom emphasizing triangular, rectangular, and hexagonal cells

Figure 7: Material zones for the heterogeneous test problem

Table 4: Material Properties For the Different Regions

	Inner region	Intermediate region	Outer region
Σ_t (cm^{-1})	1.5	1.0	1.0
Σ_s (cm^{-1})	1.4999	0.999	0.3
source $(cm^{-3}s^{-1}$	1.0	0.0	0.0

Table 5: Comparison of preconditioners (heterogeneous problem)

	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	278	17	17	17	17	17
Precond init (s)	NA	NA	0.0160661	0.368768	1.41632	0.07
MIP calculation (s)	NA	58.422	126.93	33.2225	31.3045	2.924
CG iterations	NA	12214	6679	415	386	248
Total calculation (s)	910.566	120.889	190.413	99.7524	97.4666	70.6424

remarks made in Section 5.2.1 for the homogeneous test problem remain mostly unchanged. MIP-DSA is effective for this heterogeneous test case, and AGMG is still the fastest solver. It is interesting to note that, contrary to the homogeneous tests where the number of CG iterations was about identical for all algebraic multigrid preconditioners, in this heterogeneous test case, AGMG requires significantly fewer iterations than the Trilinos implementations, PCG-ML/U and PCG-ML/MIS.

5.2.3 Locally Refined Grids

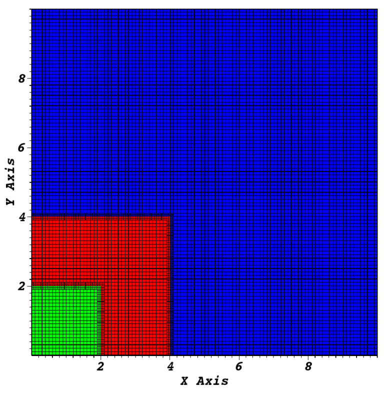
In this example, a 3-material domain of size $10cm \times 10cm$ is used. Figure 8a shows the material zoning and the mesh used. Material properties are given in Table 6 and their spatial distribution is given in Figure 8a. The bottom and left sides of the domain have reflective boundary conditions, while the other two sides have vacuum boundary conditions.

The grid used mimics meshes obtained via adaptive mesh refinement: the rectangular cells at the interfaces between two materials are refined once more, leading to a grid composed of 10,482 quadrilaterals, 236 pentagons, and 2 hexagons for a total of 10,720 cells and 43120 degrees of freedom. The distribution of cells is given in Figure 8b.

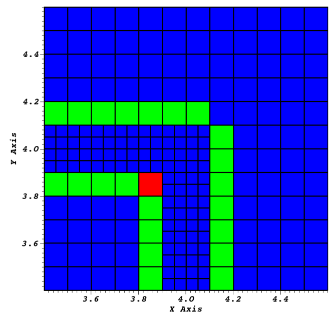
A S_{16} angular quadrature is employed. The tolerance on SI is 10^{-8}, and the tolerance on the CG solvers is 10^{-10}. The different linear solvers for MIP-DSA are compared in Table 7. The conclusions drawn from this test case are similar to the ones made for our previous tests. This test case demonstrates that degenerate polygons (here, pentagons and hexagons) do not seem to affect the MIP-DSA acceleration.

Table 6: Material properties, AMR-like test problem

	Inner region	Intermediate region	Outer region
Σ_t (cm^{-1})	1.5	1.0	1.0
Σ_s (cm^{-1})	1.44	0.9	0.3
Source $(cm^{-3}s^{-1})$	1.0	0.0	0.0



(a) Material regions



(b) Polygons distribution (zoom): quadrilaterals (blue cells), pentagons (green cells), hexagons (red cells)

Figure 8: AMR-like test domain

Table 7: Comparison of preconditioners for the AMR mesh

	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	184	19	19	19	19	19
Precond init (s)	NA	NA	0.043463	0.358002	1.19301	0.0111
MIP calculation (s)	NA	48.1908	81.0992	25.2699	25.0699	2.56198
CG iterations	NA	11300	4734	361	361	264
Total calculation (s)	802.985	138.825	172.423	116.018	116.517	94.1963

5.2.4 High Aspect Ratio Grids

In these last two examples, a square domain of $100cm \times 100cm$ with vacuum boundaries is employed. There are 10,000 cells (thus, 40,000 degrees of freedom). Again, the relative tolerance on SI is 10^{-8} and the relative tolerance for CG is 10^{-10}. An S_8 GLC angular quadrature is used. $\Sigma_t = 1cm^{-1}$, $\Sigma_s = 0.999cm^{-1}$ and the source is $1n/(cm^3s)$. In the first run, the domain is discretized using 100 subdivisions of the x and y axes, i.e., 10,000 square cells with an aspect ratio of one. In the second run, the domain is discretized using 1,000 subdivision along x and 10 along y (the aspect ratio is then 100). As expected, solving the MIP-DSA equations requires more CG iterations when the aspect ratio increases. PCG-ML/U and PCG-ML/MIS are significantly more affected by the increase in the aspect ratio than the other methods. AGMG is the least affected by the change of aspect ratio and is again the best performing preconditioner.

Table 8: Preconditioners comparison for a rectangular grid with an aspect ratio of 1

	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	7311	21	21	21	21	21
Precond init (s)	NA	NA	0.01422	0.051373	1.13144	0.044
MIP calculation (s)	NA	32.3825	73.8422	24.0707	25.0065	1.7114
CG iterations	NA	8363	4853	376	375	221
Total calculation (s)	7356.96	56.8993	98.2609	50.1247	51.5396	25.9306

Table 9: Preconditioners comparison for a rectangular grid with an aspect ratio of 100

	No-DSA	CG	PCG-SGS	PCG-ML/U	PCG-ML/MIS	AGMG
SI iterations	7304	24	24	24	24	24
Precond init (s)	NA	NA	0.0164239	0.362463	1.03128	0.052
MIP calculation (s)	NA	372.227	742.902	941.06	922.258	6.93176
CG iterations	NA	84802	43466	14180	13896	821
Total calculation (s)	9035.6	414.301	784.77	985.796	966.77	44.7032

6 Conclusions

We have extended the Modified Interior Penalty (MIP) form of Diffusion Synthetic Acceleration (DSA) to arbitrary polygonal grids discretized with Piece-Wise Linear Discontinuous finite elements. The MIP-DSA equation employs the same discontinuous finite element trial spaces as the spatial discretization of the S_n transport equation. As such, only a few additional elementary matrices need to be implemented to an existing S_n code.

Fourier analyses show that the PWLD discretization of the MIP diffusion synthetic accelerator is always stable and effective, including for cells with high-aspect ratios. Numerical experiments have been performed with grids containing various types of polygonal cells (random quadrilaterals, random polygons, hexagons), including grids with different polygon types for a given mesh and tests with degenerate polygons that mimic grids obtained in adaptive mesh refinement strategies. In these tests, MIP-DSA always performed effectively, reducing significantly the number of Source Iterations needed for convergence. We noted that the effectiveness of MIP does not seem to be affected by the meshes employed.

The MIP-DSA matrix is Symmetric Positive Definite (SPD) and we solved the resulting linear system of equations using a standard Conjugate Gradient method with different preconditioners. Algebraic multigrid techniques (AGMG and ML) were found to be the most effective preconditioner, with AGMG being more than 20 times faster than unpreconditioned CG.

We conclude that MIP-DSA is an effective alternative to carry out diffusion synthetic accelerations for S_n radiation transport problems on arbitrary polygonal grids and grids obtained through mesh adaptivity. Future extensions of this work will include the implementation of the MIP-DSA equations in a 3D parallel S_n code.