Notes on Numerical Methods for Two-Dimensional Neutron Transport Equation

Mohammed Seaïd*

Fachbereich Mathematik
TU Darmstadt, 64289 Darmstadt, Germany

Abstract

Detailed numerical methods for two-dimensional neutron transport equation are presented. Using the discrete ordinates for angle collocation and the Diamond differencing for space discretization, the neutron transport equation is transformed to a system of sparse matrices. To solve the final system we formulate the source iteration, a full BICGSTAB and GMRES algorithms. Additionally, the diffusion limit and the diffusion synthetic acceleration are included in these notes. The robustness, efficiency and convergence rates of these methods are illustrated by two numerical examples.

Keywords. Neutron transport equation, Discrete ordinates method, Diamond differencing, Diffusion limit, Diffusion synthetic acceleration, Linear algebra

AMS subject classifications. 65R20; 65N06; 65F10; 82D75

Contents

*E-mail:seaid@mathematik.tu-darmstadt.de

1 Introduction

The problem of non-energetic transport of neutrons in a substance surrounded by vacuum can be formulation by the following integro-differential equation

$$
\begin{aligned}
\frac{1}{v}\frac{\partial \psi}{\partial t} + \Omega \cdot \nabla \psi + (\sigma + \kappa)\psi &= \frac{\sigma}{4\pi}\int_{S^2} \psi(t, \mathbf{x}, \Omega')d\Omega' + q(t, \mathbf{x}, \Omega), \quad \text{in} \quad [0, T) \times \mathcal{D} \times S^2, \\
\psi(t, \mathbf{x}, \Omega) &= g(t, \mathbf{x}, \Omega), \qquad \text{on} \quad [0, T) \times \partial\mathcal{D}^- \times S^2, \\
\psi(0, \mathbf{x}, \Omega) &= \psi^0(\mathbf{x}, \Omega), \qquad \text{in} \quad \mathcal{D} \times S^2,
\end{aligned}
\tag{1}
$$

where $\mathcal{D}$ is a space domain with smooth boundary $\partial\mathcal{D}$, $[0, T)$ is a time interval, and S^2 is the unit sphere. Here $\psi(t, \mathbf{x}, \Omega)$ is the angular flux at time t and position point $\mathbf{x} := (x, y, z)^T$ in the direction $\Omega := (\mu, \xi, \eta)^T$ with fixed speed v, $\sigma := \sigma(t, \mathbf{x})$ is the scattering cross section, $\kappa := \kappa(t, \mathbf{x})$ is the absorption cross section, and $q(t, \mathbf{x}, \Omega)$ is an external source. $g(t, \mathbf{x}, \Omega)$ and $\psi^0(\mathbf{x}, \Omega)$ are known boundary and initial functions, respectively. We assume that σ and κ are non-negative functions. The boundary region $\partial\mathcal{D}^-$ is defined as

$$
\partial\mathcal{D}^- := \Big\{ \mathbf{x} \in \partial\mathcal{D} | \quad \mathbf{n}(\mathbf{x}) \cdot \Omega < 0 \Big\}, \tag{2}
$$

with $\mathbf{n}(\mathbf{x})$ is the outward normal at the point $\mathbf{x}$ on $\partial\mathcal{D}$. Despite the equation (1) is linear, computing its numerical solution is not trivial due to:

1. The large number of dependent unknowns. In general, the solution ψ in (1) is a function of eight independent variables, three space variables (x, y, z), three angle variables (μ, ξ, η), one energy variable E, and one time variable t. After discretizing these variables the computer memory requirements and the computational cost become drastically immense. This imposes severe restrictions on computational methods for (1).

2. In many neutron transport equation (1), the solution is not a smooth function of the dependent variables $(t, \mathbf{x}, E, \Omega)$. Steep fronts and even shock discontinuities can arise, which need to be resolved accurately in applications and often cause severe numerical difficulties.

3. It is well known that the equation (1) change the behaviour from a physical situation to another. For example, the equation (1) behaves like hyperbolic in void-like regions; in optically dense region, it behaves like elliptic for steady-state case and parabolic for time-dependent case. To construct an unified computational algorithm that resolve accurately all the behaviour cases is extremely difficult.

For physical interest, we define a scattering ratio γ and an optical coefficient ϑ associated to the equation (1) as

$$
\gamma := \max_{\mathbf{x} \in \mathcal{D}} \left(\frac{\sigma(\mathbf{x})}{\sigma(\mathbf{x}) + \kappa(\mathbf{x})} \right), \qquad \text{and} \qquad \vartheta := \min_{\mathbf{x} \in \mathcal{D}} \Big(\sigma(\mathbf{x}) + \kappa(\mathbf{x}) \Big) \operatorname{diam}(\mathcal{D}), \tag{3}
$$

respectively. $\operatorname{diam}(\mathcal{D})$ is the diameter of the space domain $\mathcal{D}$. Two extreme situations in computational neutron transport remain active field of research

1. $\gamma = 1$, pure scattering, no absorption ($\kappa = 0$)

2. $\vartheta \gg 1$, optically thick, dense absorption ($\kappa \gg 1$)

and the conventional methods suffer some difficulties to solve accurately these two cases as mentioned out in [2, 1, 18]. For simplicity in presentation, we consider in these notes only the two-dimensional version of (1). Thus, $\mathcal{D} \subset I\!R^2$, $\mathbf{x} := (x,y)^T$, and $\Omega := (\mu,\eta)^T$. By introducing the scalar flux ϕ

$$\phi(\mathbf{x}) := \frac{1}{4\pi}\int_{S^2} \psi(\mathbf{x},\Omega')d\Omega', \tag{4}$$

the two-dimensional time-independent neutron transport equation reads

$$\begin{aligned} \mu\frac{\partial\psi}{\partial x} + \eta\frac{\partial\psi}{\partial y} + (\sigma+\kappa)\psi &= \sigma\phi(x,y) + q(x,y,\mu,\eta), \quad \text{in} \quad \mathcal{D}\times S^2, \\ \psi(x,y,\mu,\eta) &= g(x,y,\mu,\eta), \qquad\qquad \text{on} \quad \partial\mathcal{D}^-\times S^2. \end{aligned} \tag{5}$$

The aim of these notes is to give a detailed overview on the quality and efficiency of classical and modern algorithms in computational neutron transport. At first we will show how to discretize the equation (5) in angle and space. By the example of discrete ordinates and Diamond difference methods, we will demonstrate how to develop iterative solvers for the fully discrete system. Many studies of these solvers have been done in a number of books and papers by, among others, [18, 1, 2, 22, 21, 5, 3] and the herein cited references.

Our notes contain several approaches to the construction of a suitable algorithm which can serve as black-box solver for the general neutron transport equation (1). The easiest possibility is the use of Richardson iteration known by astrophysicists as $\wedge$-iteration. This strategy has been theoretically analysed in details in [1, 18]. Another basic solver is the P_1/Diffusion approach, which means the equation (1) is replaced by a scaled diffusion problem such that at the limit tend to approximate the solution of (1). Generalized P_1/Diffusion and other simplified P_N approximations have been introduced in [17, 15, 24]. Iterative solvers for linear systems based on Krylov subspace method like the BICGSTAB [25] or GMRES [23] are also implemented in these notes. Generalzing the idea of constructing preconditioner for an accelerated $\wedge$-iteration we show how to use the P_1/Diffusion approach as an optimal preconditioner for a given $\wedge$-iteration. The resulting solver known as diffusion synthetic acceleration was introduced in [2] and studied from a linear algebra point of view in [5, 3]. Finally we apply these solvers to various test cases from neutron transport problem. Dependent on the scattering ratio γ and the optical coefficient ϑ of the problem under consideration, we can identify the most suitable solver in terms of accuracy and computing cost.

These notes are organized as follows: In section 2 the discrete ordinates method is presented. The space discretization is formulated in section 3. Iterative schemes for the full discrete problem are discussed in section 4. The section 5 is devoted to the diffusion limit and the diffusion synthetic acceleration method. Section 6 contains numerical results and comparison between the different algorithms in terms of accuracy and efficiency. Finally some conclusions are listed in section 7. For sake of completeness, the ray effect of the discrete ordinates is shown by an example in appendix A and a Fortran code for the source iteration is set as appendix B.

2 Discrete Ordinates Method

The method of discrete ordinates was introduced and used in [6] to solve well many basic problems in the area of radiative transfer. The method consists of replacing the integral terms in some form of the Boltzmann equation by numerical quadrature approximations of those terms, and then a resulting set of ordinary differential equations is solved. In much of the literature [6, 22, 21, 18] the discrete ordinate method is detailed only for the one-dimensional slab geometry case and the theoretical results remain valid for the multi-dimensional cases. In this section we formulate and overview this method for the two-dimensional model (5).

A standard approach for the integral expression over the unit sphere S^2 in (12) is the quadrature rules of the form

$$\int_{S^2} \psi(\mathbf{x}, \Omega') d\Omega' \simeq \sum_{l=1}^{\bar{N}} \omega_l \psi(\mathbf{x}, \Omega'_l), \tag{6}$$

where $\Omega'_l := (\mu_l, \xi_l, \eta_l)^T$, for $l = 1, 2, \dots, \bar{N}$, with $\bar{N} = n(n+2)$, and n is the number of directions cosines. Since $\Omega'_l \in S^2$, we have

$$\mu_l^2 + \xi_l^2 + \eta_l^2 = 1, \qquad \text{for all} \quad l = 1, 2, \dots, \bar{N}.$$

We assume n an even number of quadrature points so that the points (μ_l, ξ_l, η_l) are nonzero, symmetric about the origin, and

$$\mu_l^2 = \mu_1^2 + 2\frac{l-1}{n-2}(2 - 3\mu_1^2). \tag{7}$$

For the weights ω_l we assume all are positive and satisfy

$$\sum_{l=1}^{\bar{N}} \omega_l = 4\pi, \quad \sum_{l=1}^{\bar{N}} \omega_l \mu_l = 0, \quad \sum_{l=1}^{\bar{N}} \omega_l \xi_l = 0, \quad \text{and} \quad \sum_{l=1}^{\bar{N}} \omega_l \eta_l = 0. \tag{8}$$

A simple way to guaranty the conditions (8) is to set all weights positive and equal to $\frac{4\pi}{\bar{N}}$.

Note that the approximation (6) can be derived using the spherical coordinates. If the direction vector $\Omega := (\sin\varphi\cos\theta, \sin\varphi\sin\theta, \cos\varphi)^T$, then

$$\int_{S^2} \psi(\mathbf{x}, \Omega') d\Omega' = \int_0^\pi \int_0^{2\pi} \psi(\mathbf{x}, \theta', \varphi') d\theta' \sin\varphi' d\varphi'.$$

The trapezoidal rule for each one-dimensional integral separately yields

$$\int_0^\pi \int_0^{2\pi} \psi(\mathbf{x}, \theta', \varphi') d\theta' \sin\varphi' d\varphi' \simeq \sum_{l=1}^{L} \sum_{k=1}^{K} w_{lk} \psi(\mathbf{x}, \theta_l, \varphi_k), \tag{9}$$

where the quadrature weights

$$w_{lk} = \frac{\pi}{2LK} \sin\varphi_l.$$

By taking the sum over k in w_{lk}, the expression (9) is equivalent to the quadrature rule (6) with

$$\omega_l = \sum_{k=1}^{K} w_{lk}, \quad \text{and} \quad \bar{N} = LK.$$

Other methods to discretize the unit sphere S^2 are the so-called S_n-direction sets. A review dealing with the S_n sets can be found in [8], and a comparison between different S_n sets for radiative transfer have been done in [10]. These S_n sets satisfy the conditions (7) and (8). Furthermore, they are arranged on $n/2$ levels, invariant under 90° rotations, and they have equal positive weights, see figure 1 for an illustration of S_{12} set in two-dimensional case. Here the direction ξ is omitted.

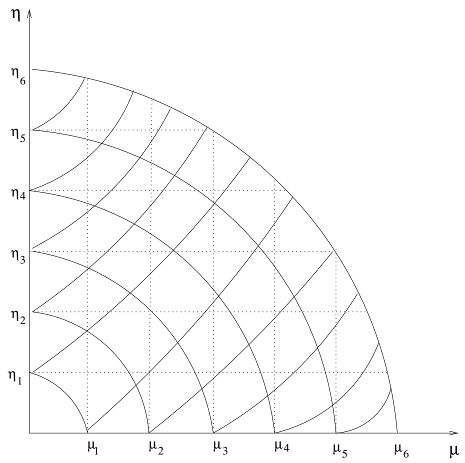

Figure 1: The S_{12}-direction set for the two-dimensional problems.

Let $S_{\bar{N}}$ be a chosen set of discrete directions in the unit sphere S^2, then the two-dimensional direction set is just the simplification of one direction in $S_{\bar{N}}$ such that the simplified set is symmetric, has nonzero direction, and with positive weights. Hence a semi-discrete formulation of the neutron transport equation (5) is given by

$$\begin{aligned} \mu_l \frac{\partial \psi_l}{\partial x} + \eta_l \frac{\partial \psi_l}{\partial y} + (\sigma + \kappa)\psi_l &= \sigma\phi(x,y) + q_l(x,y), \quad \text{in} \quad \mathcal{D} \times S_{\bar{N}}, \\ \psi_l(x,y) &= g_l(x,y), \qquad\qquad \text{on} \quad \partial\mathcal{D}^- \times S_{\bar{N}}. \end{aligned} \tag{10}$$

$\psi_l(x,y)$, $q_l(x,y)$ and $g_l(x,y)$ are approximations to $\psi(x,y,\mu_l,\eta_l)$, $q(x,y,\mu_l,\eta_l)$ and $g(x,y,\mu_l,\eta_l)$, respectively. Note that the angular discretization (10) transforms the original integro-differential equation (5) into a system of $\bar{N}$ coupled differential equations.

Remark 1 *One of our favourite S_n-direction set is the C-60 known as buckyball in [7]. the set contains 60 equal weighted directions with high symmetry configuration. The C-60 set is reproduced in table 1. In our numerical examples we used others S_n sets and C-60 yields the best results. However, the main disadvantage using these sets is we can not refine the ordinates within the same set as we can do using the usual trapezoidal or Gauss quadrature rules.*

3 Space Discretization

The discrete ordinates method can be applied in combination with finite elements, finite differences or spectral methods. In [5] the author combines the Petrov-Galerkin method with the discrete ordinates collocation for the neutron transport equation (1). Since it is easier to combine

Table 1: The C-60 directions set used in our numerical test problems.

l	μ_l	η_l	ω_l	l	μ_l	η_l	ω_l
1	-0.9642754578	-0.1716393065	0.2094395102	31	0.0655603813	-0.5149179195	0.2094395102
2	-0.9642754578	-0.1716393065	0.2094395102	32	0.0655603813	-0.5149179195	0.2094395102
3	-0.8987150765	0.1716393065	0.2094395102	33	0.1060789252	0.1716393065	0.2094395102
4	-0.8987150765	0.1716393065	0.2094395102	34	0.1060789252	0.9392336205	0.2094395102
5	-0.8331546952	0.5149179195	0.2094395102	35	0.1060789252	0.9392336205	0.2094395102
6	-0.8331546952	0.5149179195	0.2094395102	36	0.1060789252	0.1716393065	0.2094395102
7	-0.7926361513	-0.5149179195	0.2094395102	37	0.2121578505	0.7270757700	0.2094395102
8	-0.7926361513	-0.5149179195	0.2094395102	38	0.2121578505	0.7270757700	0.2094395102
9	-0.6865572261	-0.7270757700	0.2094395102	39	0.2777182317	-0.9392336205	0.2094395102
10	-0.6615153887	0.1716393065	0.2094395102	40	0.2777182317	-0.9392336205	0.2094395102
11	-0.6615153887	0.1716393065	0.2094395102	41	0.3432786130	0.9392336205	0.2094395102
12	-0.5554364635	-0.5149179195	0.2094395102	42	0.4493575383	-0.5149179195	0.2094395102
13	-0.5554364635	0.7270757700	0.2094395102	43	0.4493575383	-0.5149179195	0.2094395102
14	-0.5554364635	0.7270757700	0.2094395102	44	0.4898760822	0.1716393065	0.2094395102
15	-0.5554364635	-0.5149179195	0.2094395102	45	0.4898760822	0.1716393065	0.2094395102
16	-0.4898760822	-0.1716393065	0.2094395102	46	0.5554364635	0.5149179195	0.2094395102
17	-0.4898760822	-0.1716393065	0.2094395102	47	0.5554364635	-0.7270757700	0.2094395102
18	-0.4493575383	0.5149179195	0.2094395102	48	0.5554364635	-0.7270757700	0.2094395102
19	-0.4493575383	0.5149179195	0.2094395102	49	0.5554364635	0.5149179195	0.2094395102
20	-0.3432786130	-0.9392336205	0.2094395102	50	0.6615153887	-0.1716393065	0.2094395102
21	-0.2777182317	0.9392336205	0.2094395102	51	0.6615153887	-0.1716393065	0.2094395102
22	-0.2777182317	0.9392336205	0.2094395102	52	0.6865572261	0.7270757700	0.2094395102
23	-0.2121578505	-0.7270757700	0.2094395102	53	0.7926361513	0.5149179195	0.2094395102
24	-0.2121578505	-0.7270757700	0.2094395102	54	0.7926361513	0.5149179195	0.2094395102
25	-0.1060789252	-0.1716393065	0.2094395102	55	0.8331546952	-0.5149179195	0.2094395102
26	-0.1060789252	-0.9392336205	0.2094395102	56	0.8331546952	-0.5149179195	0.2094395102
27	-0.1060789252	-0.9392336205	0.2094395102	57	0.8987150765	-0.1716393065	0.2094395102
28	-0.1060789252	-0.1716393065	0.2094395102	58	0.8987150765	-0.1716393065	0.2094395102
29	-0.0655603813	0.5149179195	0.2094395102	59	0.9642754578	0.1716393065	0.2094395102
30	-0.0655603813	0.5149179195	0.2094395102	60	0.9642754578	0.1716393065	0.2094395102

the upwinding with finite volume discretization than other methods, we consider in these notes a space discretization based on volume control and cell averaging. For simplicity, we assume that the space domain is a rectangle, $\mathcal{D} := [a,b] \times [c,d]$. Thus the numerical grid is defined by

$$\mathcal{D}_h := \Big\{ \mathbf{x}_{ij} = (x_i, y_j)^T, x_i = i(\Delta x)_i, y_j = j(\Delta y)_j, i = 1,2\dots,N, j = 1,2\dots,M \Big\},$$

$x_0 = a, x_N = b$, $y_0 = c, y_M = d$, and h denotes the maximum cell size $h := \max_{ij}\big((\Delta x)_i, (\Delta y)_j\big)$. We define the averaged grid points as

$$(\Delta x)_{i+\frac{1}{2}} := x_{i+1} - x_i, \quad (\Delta y)_{j+\frac{1}{2}} := y_{j+1} - y_j, \quad x_{i+\frac{1}{2}} := \frac{x_{i+1} + x_i}{2}, \quad y_{j+\frac{1}{2}} := \frac{y_{j+1} + y_j}{2}.$$

We use the notation f_{ij} to denote the approximation value of the function f at the grid point (x_i, y_j). Using the semi-discrete formulation (10), a fully discrete approximation for the equation (5) can be directly written as

$$\begin{aligned} \mu_l \frac{\psi_{l,i+1j} - \psi_{l,ij}}{(\Delta x)_{i+\frac{1}{2}}} + \eta_l \frac{\psi_{l,ij+1} - \psi_{l,ij}}{(\Delta y)_{j+\frac{1}{2}}} \quad &+ \quad \big(\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}\big)\psi_{l,i+\frac{1}{2}j+\frac{1}{2}} = \\ &\sigma_{i+\frac{1}{2}j+\frac{1}{2}}\phi_{i+\frac{1}{2}j+\frac{1}{2}} + q_{l,i+\frac{1}{2}j+\frac{1}{2}}, \end{aligned} \tag{11}$$

where the cell averages of ψ are given by

$$
\begin{aligned}
\psi_{l,i+1j} &= \frac{1}{(\Delta x)_{i+\frac{1}{2}}} \int_{y_j}^{y_{j+1}} \psi_l(x_i, y)dy, \\
\psi_{l,ij+1} &= \frac{1}{(\Delta y)_{j+\frac{1}{2}}} \int_{x_i}^{x_{i+1}} \psi_l(x, y_j)dx, \\
\psi_{l,ij} &= \frac{1}{(\Delta x)_{i+\frac{1}{2}}(\Delta y)_{j+\frac{1}{2}}} \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \psi_l(x, y)dxdy,
\end{aligned} \tag{12}
$$

To approximate the fluxes (12), we use the well known Diamond difference method which consist on centred differences and approximating the function values at the cell centres by the average of their values at the neighbouring nodes. See the figure 2 for an illustration of the grids used in these notes. The function value of $f_{i+\frac{1}{2}j+\frac{1}{2}}$ at the cell centre is simply approximated by bilinear interpolation as

$$
f_{i+\frac{1}{2}j+\frac{1}{2}} = \frac{f_{ij} + f_{i+ij} + f_{ij+1} + f_{i+1j+1}}{4}. \tag{13}
$$

Hence the scalar flux $\phi_{i+\frac{1}{2}j+\frac{1}{2}}$ in (11) is given by

$$
\phi_{i+\frac{1}{2}j+\frac{1}{2}} = \sum_{l=1}^{\bar{N}} \omega_l \frac{\psi_{ij} + \psi_{i+ij} + \psi_{ij+1} + \psi_{i+1j+1}}{4}.
$$

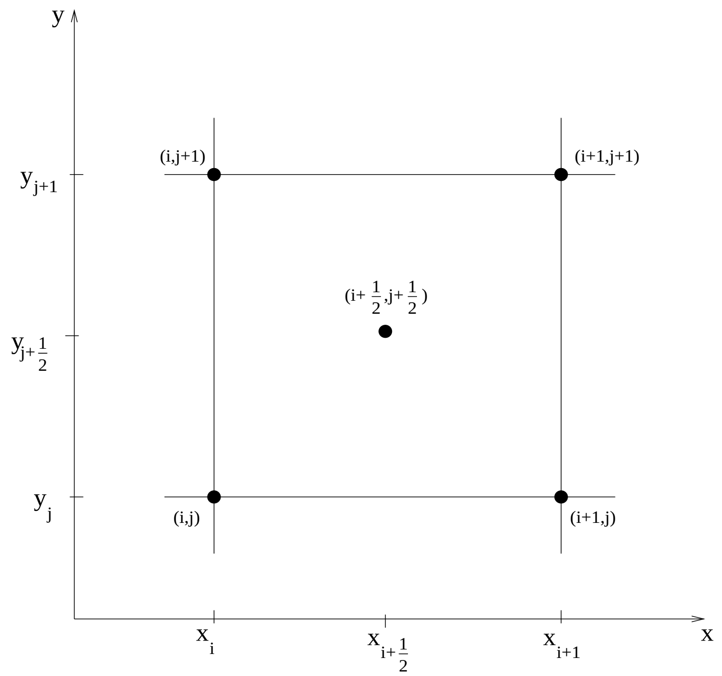

Figure 2: The staggered grid used for the space discretization.

For the boundary conditions in (10) we can proceed as follows:

when $x = a$, the normal $\mathbf{n} = (-1,0)^T$, then $\mathbf{n}\cdot\Omega_l = -\mu_l$, and for $\mu_l > 0$ we have $\psi_{l,0j} = g_{l,0j}$

when $x = b$, the normal $\mathbf{n} = (1,0)^T$, then $\mathbf{n}\cdot\Omega_l = \mu_l$, and for $\mu_l < 0$ we have $\psi_{l,Nj} = g_{l,Nj}$

when $y = c$, the normal $\mathbf{n} = (0,-1)^T$, then $\mathbf{n}\cdot\Omega_l = -\eta_l$, and for $\eta_l > 0$ we have $\psi_{l,i0} = g_{l,i0}$

when $y = d$, the normal $\mathbf{n} = (0,1)^T$, then $\mathbf{n}\cdot\Omega_l = \eta_l$, and for $\eta_l < 0$ we have $\psi_{l,iM} = g_{l,iM}$

Remark 2 *If the space domain $\mathcal{D}$ present some points on the boundary $\partial\mathcal{D}^-$ where the normal is not unique, corners for example in the case of a rectangular domain, then, is possible to define a new normal on those points with multiple normal. For instance, at the left lower corner point $\mathbf{x} = (a,c)^T$ in the rectangle a new normal can be define as $\tilde{\mathbf{n}} = (-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})^T$, and for $\tilde{\mathbf{n}}\cdot\Omega_l = -\frac{\sqrt{2}}{2}\mu_l - \frac{\sqrt{2}}{2}\eta_l < 0$, we have $\psi_{l,00} = g_{l,00}$. Similar work can be done for other three remaining corners.*

In order to simplify the notations and to get closer to a compact linear algebra formulation of (11), we first define the matrix entries

$$
\begin{aligned}
d_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}, \\
\bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}, \\
\underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}, \\
e_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}.
\end{aligned}
$$

Define the vectors

$$
\Psi_l \equiv \begin{pmatrix} \Psi_{l,0} \\ \vdots \\ \Psi_{l,M} \end{pmatrix} \in I\!R^{(N+1)(M+1)}, \quad \text{with} \quad \Psi_{l,j} \equiv \begin{pmatrix} \psi_{l,0j} \\ \vdots \\ \psi_{l,Nj} \end{pmatrix} \in I\!R^{N+1},
$$

$$
\Phi \equiv \begin{pmatrix} \Phi_{\frac{1}{2}} \\ \vdots \\ \Phi_{M-\frac{1}{2}} \end{pmatrix} \in I\!R^{NM}, \quad \text{with} \quad \Phi_{j-\frac{1}{2}} \equiv \begin{pmatrix} \phi_{\frac{1}{2}j-\frac{1}{2}} \\ \vdots \\ \phi_{N-\frac{1}{2}j-\frac{1}{2}} \end{pmatrix} \in I\!R^{N};
$$

$$
\text{and} \qquad \mathbf{Q}_l \equiv \begin{pmatrix} Q_{l,\frac{1}{2}} \\ \vdots \\ Q_{l,M-\frac{1}{2}} \end{pmatrix} \in I\!R^{NM}, \quad \text{with} \quad Q_{l,j-\frac{1}{2}} \equiv \begin{pmatrix} q_{l,\frac{1}{2}j-\frac{1}{2}} \\ \vdots \\ q_{l,N-\frac{1}{2}j-\frac{1}{2}} \end{pmatrix} \in I\!R^{N}.
$$

Recall that the $S_{\bar{N}}$-direction set used for the discrete ordinates formulation (10) avoid the zero component in a given direction $\Omega_l = (\mu_l, \eta_l)$. So, only one of the four cases; $\mu_l < 0$ and $\eta < 0$, $\mu_l < 0$ and $\eta > 0$, $\mu_l > 0$ and $\eta < 0$, or $\mu_l > 0$ and $\eta > 0$ can hold. Here we define the matrices

$\mathbf{H}_l$ and Σ_l for the case $\mu_l < 0$ and $\eta < 0$, and the other three cases can be derived similarly.

$$\mathbf{H}_l \equiv \begin{pmatrix} D_l & \underline{E}_l & & & \\ & \ddots & \ddots & & \\ & & D_l & \underline{E}_l & \\ & & & D_l & S \\ & & & & S \end{pmatrix} \in I\!R^{(N+1)(M+1)\times(N+1)(M+1)}, \qquad \text{with}$$

$$D_l \equiv \begin{pmatrix} d & \bar{e} & & \\ & \ddots & \ddots & \\ & & d & \bar{e} \\ & & & 1 \end{pmatrix} \in I\!R^{(N+1)\times(M+1)}, \quad \underline{E}_l \equiv \begin{pmatrix} \underline{e} & e & & \\ & \ddots & \ddots & \\ & & \underline{e} & e \\ & & & 1 \end{pmatrix} \in I\!R^{(N+1)\times(M+1)},$$

$$\text{and} \qquad S \equiv \begin{pmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \\ & & & 1 \end{pmatrix} \in I\!R^{(N+1)\times(M+1)}.$$

$$\Sigma_l \equiv \begin{pmatrix} \Sigma_{l,\frac{1}{2}} & & \\ & \ddots & \\ & & \Sigma_{l,M-\frac{1}{2}} \\ & & \mathbf{0} \end{pmatrix} \in I\!R^{(N+1)(M+1)\times NM}, \qquad \text{with}$$

$$\Sigma_{l,j} \equiv \begin{pmatrix} \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}}+\kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4} & & \\ & \ddots & \\ & & \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}}+\kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4} \\ & & 0 \end{pmatrix} \in I\!R^{(N+1)\times M}.$$

With these definitions, the equation (11) can be written in the unknowns Ψ and Φ as

$$\left(\begin{array}{ccc|c} \mathbf{H}_1 & & & -\Sigma_1 \\ & \ddots & & \vdots \\ & & \mathbf{H}_{\bar{N}} & -\Sigma_{\bar{N}} \\ \hline -\omega_1\mathbf{S} & \dots & -\omega_{\bar{N}}\mathbf{S} & \mathbf{I} \end{array}\right) \left(\begin{array}{c} \Psi_1 \\ \vdots \\ \Psi_{\bar{N}} \\ \hline \Phi \end{array}\right) = \left(\begin{array}{c} \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_{\bar{N}} \\ \hline \mathbf{0} \end{array}\right), \tag{14}$$

where $\mathbf{I}$ is the $N \times M$ identity matrix and $\mathbf{0}$ is the N null vector. The usual technique to solve the equation (14), is to eliminate the angular flux $\Psi_1, \dots, \Psi_{\bar{N}}$ using the Gaussian elimination. Therefore the storage requirements is reduced and the resulting equation

$$\left(\mathbf{I} - \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\Sigma_l\right)\Phi = \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\mathbf{Q}_l, \tag{15}$$

is solved for the scalar flux Φ, which does not depend on direction variables. Furthermore, solving (15) does not need to store the dense $NM \times NM$ schur matrix,

$$\mathcal{A} \equiv \mathbf{I} - \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\Sigma_l. \tag{16}$$

For instance, to apply this matrix to a given NM vector $\mathbf{U}$, only three NM vectors are needed. The first is used to store the product $\mathbf{U}$ by Σ_l, in the second we store the solution of the linear system with the matrix $\mathbf{H}_l$. Multiplying by $\mathbf{S}$ and subtracting the weighted resulting vector from $\mathbf{U}$ is stored in the third vector.

Since the key idea in all the incoming numerical methods dealing with the equation (15) is inverting the matrix $\mathbf{H}_l$ for $l = 1, \dots, \bar{N}$, we set up the following algorithm performing this step

`Algorithm 1:` **sweeping**$(N,M,\bar{N},\Delta x,\Delta y,\sigma,\kappa,\mu,\eta,\mathbf{Q},\Psi,\mathbf{U})$

do $l = 1, \dots, \bar{N}$

do $i = 1, \dots, N$

do $j = 1, \dots, M$

$$d_{l,i+\frac{1}{2}j+\frac{1}{2}} = \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}$$

$$\bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} = \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}$$

$$\underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} = \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}$$

$$e_{l,i+\frac{1}{2}j+\frac{1}{2}} = \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}{4}$$

end do

end do

if ($\mu_l < 0$ and $\eta < 0$) then

do $i = 1, \dots, N+1$

$\psi_{l,iM+1} = q_{l,iM+1}$

end do

do $j = 1, \dots, M+1$

$\psi_{l,N+1j} = q_{l,N+1j}$

end do

do $i = N, \dots, 1$

do $j = M, \dots, 1$

$$\psi_{l,ij} = \frac{u_{i+\frac{1}{2}j+\frac{1}{2}} - \underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j} - e_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j+1} - \bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij+1}}{d_{l,i+\frac{1}{2}j+\frac{1}{2}}}$$

end do

end do

end if

if ($\mu_l < 0$ and $\eta > 0$) then

do $i = 1, \dots, N+1$

$\psi_{l,i1} = q_{l,i1}$

end do

do $j = 1, \dots, M+1$

$\psi_{l,N+1j} = q_{l,N+1j}$

end do

do $i = N, \dots, 1$

do $j = 1, \dots, M$

$$\psi_{l,ij+1} = \frac{u_{i+\frac{1}{2}j+\frac{1}{2}} - e_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j} - \bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij} - \underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j+1}}{d_{l,i+\frac{1}{2}j+\frac{1}{2}}}$$

end do
end do
end if
if ($\mu_l > 0$ and $\eta < 0$) then
do $i = 1, \ldots, N+1$
$\psi_{l,iM+1} = q_{l,iM+1}$
end do
do $j = 1, \ldots, M+1$
$\psi_{l,1j} = q_{l,1j}$
end do
do $i = 1, \ldots, N$
do $j = M, \ldots, 1$

$$\psi_{l,i+1j} = \frac{u_{i+\frac{1}{2}j+\frac{1}{2}} - \underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij} - \bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j+1} - e_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij+1}}{d_{l,i+\frac{1}{2}j+\frac{1}{2}}}$$

end do
end do
end if
if ($\mu_l > 0$ and $\eta > 0$) then
do $i = 1, \ldots, N+1$
$\psi_{l,i1} = q_{l,i1}$
end do
do $j = 1, \ldots, M+1$
$\psi_{l,1j} = q_{l,1j}$
end do
do $i = 1, \ldots, N$
do $j = 1, \ldots, M$

$$\psi_{l,i+1j+1} = \frac{u_{i+\frac{1}{2}j+\frac{1}{2}} - \bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,i+1j} - e_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij} - \underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}}\psi_{l,ij+1}}{d_{l,i+\frac{1}{2}j+\frac{1}{2}}}$$

end do
end do
end if
end do

Note that the `Algorithm 1` is based on the Gaussian elimination known in computational neutron transport as sweeping procedures. Additionally, for each direction in $S_{\bar{N}}$ only one sweep is needed. See figure 3 for a sweep illustration.

Remark 3 *When reflective boundaries arise on no more than one vertical and one horizontal boundary, the* `Algorithm 1` *start first sweeping at the boundaries with known incoming flux then, reflected flux from the boundary is used for backsweeping. If both horizontal and/or vertical boundaries are reflective, an iterative process must be done on the boundaries. Suppose for example, both vertical boundaries are reflective i.e.,*

$$\psi_{0j}(\mu_l, \eta_l) = \psi_{0j}(-\mu_l, \eta_l), \quad \text{for} \quad \mu_l > 0 \quad \text{and} \quad \psi_{N+1j}(\mu_l, \eta_l) = \psi_{N+1j}(-\mu_l, \eta_l) \quad \text{for} \quad \mu_l < 0.$$

Then, the angular fluxes at the vertical boundaries which were calculated in one step are used as inflow boundary for the next step of iteration. The iterations are stopped as soon as, the inequality

$$\|\psi^{old} - \psi^{new}\|_{L^\infty} \leq \delta_r \|\psi^{old}\|_{L^\infty} + \delta_a$$

is satisfy. Here δ_a, δ_r are given tolerances and $\|.\|_{L^\infty}$ is the L^∞-norm.

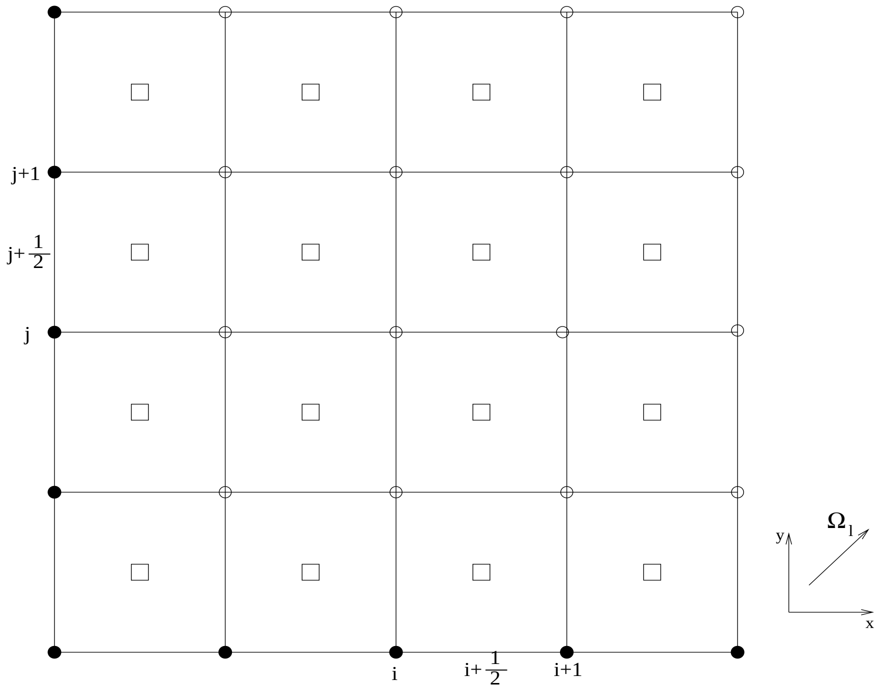

Figure 3: Sweep illustration for $\mu_l > 0$ and $\eta > 0$. $\bullet$ known boundary flux ψ, $\circ$ computed flux ψ at cell interfaces, and $\square$ computed flux ϕ at cell centre.

4 Iterative Methods

In this section we introduce some numerical methods used in the literature to solve the linear system (14), which can be rewritten in common linear algebra notation as

$$\mathbf{A}\mathbf{X} = \mathbf{b}, \tag{17}$$

with

$$\mathbf{A} \equiv \left(\begin{array}{ccc|c} \mathbf{H}_1 & & & -\Sigma_1 \\ & \ddots & & \vdots \\ & & \mathbf{H}_{\bar{N}} & -\Sigma_{\bar{N}} \\ \hline -\omega_1 \mathbf{S} & \dots & -\omega_{\bar{N}} \mathbf{S} & \mathbf{I} \end{array}\right), \quad \mathbf{X} \equiv \left(\begin{array}{c} \Psi_1 \\ \vdots \\ \Psi_{\bar{N}} \\ \hline \Phi \end{array}\right), \quad \text{and} \quad \mathbf{b} \equiv \left(\begin{array}{c} \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_{\bar{N}} \\ \hline \mathbf{0} \end{array}\right).$$

In the same spirit we can rewrite the system (15) as

$$\mathcal{A}\Phi = \mathcal{B}, \tag{18}$$

where $\mathcal{A}$ is the Schur matrix given in (16) and the right hand side $\mathcal{B} = \dfrac{1}{4\pi}\displaystyle\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\mathbf{Q}_l$.

Recall that the matrices $\mathbf{A}$ and $\mathcal{A}$ are sparse and nonsymmetric. In large scale problems iterative methods are computationally more efficient than direct methods; however, most iterative methods for nonsysmmetric systems, with the possible exception of multigrid methods, are less efficient than their symmetric counterparts.

The most popular and easiest iterative method to solve (18) is the Richardson iteration known in the computational neutron transport as Source Iteration (SI) method. Given an initial guess $\Phi^{(0)}$, the $(k+1)$-iterate solution is obtained by

$$\Phi^{(k+1)} = \Phi^{(k)} + \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\mathbf{Q}_l - \mathcal{A}\Phi^{(k)},$$

or simply

$$\Phi^{(k+1)} = \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\Big(\mathbf{Q}_l + \Sigma_l\Phi^{(k)}\Big). \tag{19}$$

In the following the SI algorithm is presented and a Fortran code with this algorithm is given in the Appendix B.

`Algorithm 2:` The SI algorithm

given the initial guess $\Psi^{(0)}$ compute $\Phi^{(0)} = \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\Psi^{(0)}$

do $k = 0, \ldots, Kmax$
  do $l = 1, \ldots, \bar{N}$
   compute $\mathbf{W} = \mathbf{Q}_l + \Sigma_l\Phi^{(k)}$
  end do
  call **sweeping**$(N, M, \bar{N}, \Delta x, \Delta y, \sigma, \kappa, \mu, \eta, \mathbf{Q}, \Psi^{(k+1)}, \mathbf{W})$
  compute $\Phi^{(k+1)} = \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\Psi_l^{(k+1)}$
  compute $\mathbf{Res}^{(k+1)} = \|\Phi^{(k+1)} - \Phi^{(k)}\|_{L^2}$
  if $\Big(\frac{\mathbf{Res}^{(k+1)}}{\mathbf{Res}^{(0)}} \le tol\Big)$ stop
end do

Here $Kmax$ is the maximum number of the iterations, tol is a given tolerance, $\|.\|_{L^2}$ is the discrete L^2-norm, and $\mathbf{Res}^{(k)}$ denotes the residual vector at iteration k.

Note that iteration (19) is equivalent to a preconditioned block Gauss-Seidel method applied to (17), where the preconditioner is the block lower triangle of the matrix $\mathbf{A}$. Thus, if $\mathbf{M}$ is the block lower triangle of $\mathbf{A}$, then

$$\mathbf{M}\mathbf{X}^{(k+1)} = (\mathbf{M} - \mathbf{A})\mathbf{X}^{(k)} + \mathbf{b},$$

and

$$\mathbf{X}^{(k+1)} = \Big(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}\Big)\mathbf{X}^{(k)} + \mathbf{M}^{-1}\mathbf{b}. \tag{20}$$

Therefore the $(k+1)$-iterate scalar flux satisfy

$$\begin{aligned}\Phi^{(k+1)} &= \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\Psi_l^{(k+1)} \\ &= \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\mathbf{H}_l^{-1}\Big(\mathbf{Q}_l + \Sigma_l\Phi^{(k)}\Big),\end{aligned}$$

which is identical to (19). Regarding to the matrix formulations (17) and (18), we have the following properties:

1. The matrices $\mathbf{A}$ and $\mathcal{A}$ are nonsymmetric. In general they are not diagonally dominant.
2. When $\bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} \leq 0$ and $\underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} \leq 0$, for all l, i, j, the matrix $\mathbf{A}$ is weakly diagonally dominant.
3. Since σ and κ are nonnegative functions, and $S_{\bar{N}}$ has nonzero directions, the matrix $\mathbf{A}$ has positive diagonal elements and nonpositive off-diagonal elements.

The fact that $\bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} \leq 0$ and $\underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} \leq 0$ is equivalent to

$$h := \max_{ij}\big((\Delta x)_i, (\Delta y)_j\big) \leq \max_{ij}\left(\frac{2|\mu_l|}{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}, \frac{2|\eta_l|}{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}}\right), \quad \forall\, l, \tag{21}$$

which means physically that the cell size is no more than two mean free paths of the particles being simulated. Needless to say that the condition (21) gives the bound of the coarser mesh should be used in the computations.

Upon the properties listed above and Fourier analysis we have the following lemma whose proof can be found in [3, 11] for the one-dimensional problem. With the same arguments the result can be extended to the two dimensional case.

Lemma 1 *Assume $\sigma(\mathbf{x}) \geq 0$, $\kappa(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathcal{D}$, and assume that $\kappa(\mathbf{x}) \geq c > 0$ on $\mathcal{D}$. Then for each direction $\Omega_l \in S_{\bar{N}}$,*

$$\|\Theta^{1/2}\mathbf{S}\mathbf{H}_l^{-1}\Sigma_l\Theta^{-1/2}\| < \gamma \leq 1,$$

where $\Theta := \mathrm{diag}\Big\{(\sigma_{\frac{1}{2}\frac{1}{2}} + \kappa_{\frac{1}{2}\frac{1}{2}})h, \ldots, (\sigma_{N-\frac{1}{2}M-\frac{1}{2}} + \kappa_{N-\frac{1}{2}M-\frac{1}{2}})h\Big\}$ and γ is defined in (3). ■

Consequently, the lemma 1 leads to the following convergence result for the SI algorithm.

Theorem 1 *Under the assumption of lemma 1, the iterations (19) converge to the solution Φ of (15), and if $\mathbf{e}^{(k)} := \Phi - \Phi^{(k)}$ denotes the error at iteration k, then*

$$\|\Theta^{1/2}\mathbf{e}^{(k+1)}\| < \|\Theta^{1/2}\mathbf{e}^{(k)}\|, \qquad k = 0, 1, \ldots,$$

where Θ is defined in the lemma 1 and γ is given in (3). □

Proof. From (19) we have

$$\Theta^{1/2}\mathbf{e}^{(k+1)} = \frac{1}{4\pi}\sum_{l=1}^{\bar{N}} \omega_l\big(\Theta^{1/2}\mathbf{S}\mathbf{H}_l^{-1}\Sigma_l\Theta^{-1/2}\big)\Theta^{1/2}\mathbf{e}^{(k)},$$

by applying norms on both sides and use the fact that the weights ω_l satisfy

$$\omega_l \geq 0, \qquad \text{and} \quad \sum_{l=1}^{\bar{N}} \omega_l = 4\pi,$$

we end up with

$$\|\Theta^{1/2}\mathbf{e}^{(k+1)}\| < \frac{1}{4\pi}\sum_{l=1}^{\bar{N}}\omega_l\gamma\|\Theta^{1/2}\mathbf{e}^{(k)}\| = \gamma\|\Theta^{1/2}\mathbf{e}^{(k)}\|. \tag{22}$$

Since the inequality (22) is strict and γ independent of k with $\gamma \leq 1$, the the iterations (19) converge to the solution of (15). Moreover, the convergence rate is bounded by γ. ■

It is well known in iterative methods for linear algebra [13, 9, 11] that the preconditioned Richardson iteration (21) converges rapidly as long as the norm of the matrix $(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})$ is small. This condition is ensured by taking γ small. As theorem 1 indicates, the convergence rate of the SI method is restricted by the scattering ratio γ. Hence, for $\gamma \ll 1$ theorem 1 shows that the SI method converges rapidly, but for $\gamma \approx 1$ (large optical opacity) convergence becomes slow and may restrict the efficiency of the SI algorithm.

In order to overcome the disadvantage of SI method to efficiently solve the problem (15) when $\gamma \approx 1$, we propose two Krylov subspace based methods, especially the BI-Conjugate Gradient Stabilized (BICGSTAB) [25] and the Generalized Minimal Residual (GMRES) [23], which work much better in this case. The main idea behind these approaches is that the Krylov subspace methods can be interpreted as the weighted Richardson iteration

$$\mathbf{X}^{(k+1)} = \alpha\Big(\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\Big)\mathbf{X}^{(k)} + \mathbf{P}^{-1}\mathbf{b}, \qquad 0 < \alpha < 2, \tag{23}$$

where the relaxation parameters α and the preconditioner $\mathbf{P}$ are variables within each iteration step. Note that when $\alpha = 1$ and $\mathbf{P} = \mathbf{M}$ the iteration (23) is reduced to the SI method.

The BICSTAB and GMRES algorithms to solve the linear system (18) can be implemented in the conventional way as in [25, 23, 13, 9, 11], with the only difference that the sparse matrix $\mathcal{A}$ can not be explicitly stored. All what is needed, however, is a subroutine that performs a matrix-vector multiplication as shown in the following algorithm

`Algorithm 3:` The matrix-vector multiplication

given a vector $\mathbf{U}$, to apply the matrix $\mathcal{A}$ to $\mathbf{U}$ we proceed as:
do $l = 1, \ldots, \bar{N}$
 compute $\mathbf{V} = \Sigma_l \mathbf{U}$
end do
call **sweeping**$(N, M, \bar{N}, \Delta x, \Delta y, \sigma, \kappa, \mu, \eta, \mathbf{Q}, \mathbf{V}, \mathbf{W})$
do $l = 1, \ldots, \bar{N}$
 compute $\mathbf{V} = \mathbf{S}_l \mathbf{W}$
end do
set $\mathbf{U} = \mathbf{U} - \dfrac{1}{4\pi}\sum_{l=1}^{\bar{N}}\omega_l \mathbf{V}$

Note that only three vectors ($\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$) are needed to perform the multiplication of the matrix $\mathcal{A}$ to the vector $\mathbf{U}$. Moreover, only three calls for the algorithm 3 are required from the BICGSTAB or GMRES subroutines.

Remark 4 *Preconditioned BICGSTAB or GMRES methods can be also used. For instance, in the case when the matrix $\mathcal{A}$ in (16) is diagonally dominant, the BICGSTAB or GMRES* *methods can be accelerated by using the diagonal as a preconditioner. This approach which requires additional computational work can be easily implemented. It is worth to say that since, the matrix $\mathcal{A}$ does not have an explicit representation, ILU type preconditioners can not be used to solve (15).*

5 Diffusion Synthetic Acceleration Method

It has been shown in [17, 16, 15], under the physical assumptions that the medium is optically thick and the scattering is dominate, the neutron transport equation (5) can be approximated by the diffusion problem

$$\begin{aligned} -\nabla \cdot \Big(\frac{1}{3(\sigma+\kappa)} \nabla \varphi \Big) + \kappa \varphi &= q \qquad \text{in} \quad \mathcal{D}, \\ \varphi + \frac{2}{3(\sigma+\kappa)} \mathbf{n} \cdot \nabla \varphi &= 4\pi g, \quad \text{on} \quad \partial \mathcal{D}. \end{aligned} \tag{24}$$

The authors in [17, 16, 15] used asymptotic analysis to prove that, in diffusive limit, the solution to the equation (24) approaches asymptotically solution of the full neutron transport equation (5). Further analysis and other asymptotic approximations to the transport problem in radiative heat transfer context can be found in [24]. The main advantages to consider the diffusion approach lie on the fact that equation (24) does not depend on the angle variable Ω, is linear elliptic equation, simple to solve numerically with less computational cost and memory requirement, and when κ is positive (24) has a unique solution.

In order to build a discretization for the diffusion problem (24) which is consistent to the one used for the neutron transport equation (5) and converges asymptotically to the same solution as the mesh size h tends to zero, we consider in this section the same grid structure as figure 2 and the same notations as those used in section 3.

Hence a space discretization for the equation (24) reads as

$$-\mathcal{D}_h^2 \Big(\frac{1}{3(\sigma+\kappa)} \varphi \Big)_{ij} + \kappa \varphi_{i+\frac{1}{2}j+\frac{1}{2}} = q_{i+\frac{1}{2}j+\frac{1}{2}}, \tag{25}$$

where the difference operator $\mathcal{D}_h^2$ is given by $\mathcal{D}_h^2 := \mathcal{D}_x^2 + \mathcal{D}_y^2$, with

$$\begin{aligned} \mathcal{D}_x^2(\xi\omega)_{ij} &:= \frac{\xi_{ij}+\xi_{i+1j}}{2} \frac{\omega_{i+1j}-\omega_{ij}}{(\Delta x)^2_{i+\frac{1}{2}}} - \frac{\xi_{i-1j}+\xi_{ij}}{2} \frac{\omega_{ij}-\omega_{i-1j}}{(\Delta x)^2_{i+\frac{1}{2}}}, \\ \mathcal{D}_y^2(\xi\omega)_{ij} &:= \frac{\xi_{ij}+\xi_{ij+1}}{2} \frac{\omega_{ij+1}-\omega_{ij}}{(\Delta y)^2_{j+\frac{1}{2}}} - \frac{\xi_{ij-1}+\xi_{ij}}{2} \frac{\omega_{ij}-\omega_{ij-1}}{(\Delta y)^2_{j+\frac{1}{2}}}, \end{aligned}$$

and the functions $\varphi_{i+\frac{1}{2}j+\frac{1}{2}}$ and $q_{i+\frac{1}{2}j+\frac{1}{2}}$ appeared in (25) are given by the formula (13). The gradient in the boundary conditions is approximated by upwinding without using ghost points. For example, on the left boundary of the domain ($x = x_0$) the boundary discretization is

$$\varphi_{\frac{1}{2}j+\frac{1}{2}} - \frac{2}{3(\sigma_{\frac{1}{2}j+\frac{1}{2}} + \kappa_{\frac{1}{2}j+\frac{1}{2}})} \frac{\varphi_{\frac{3}{2}j+\frac{1}{2}} - \varphi_{\frac{1}{2}j+\frac{1}{2}}}{(\Delta x)_{\frac{1}{2}}} = 4\pi g_{\frac{1}{2}j+\frac{1}{2}},$$

and similar work has to be done for the other boundaries. All together, the above discretization leads to a linear system of form

$$\mathcal{T}\varphi = \mathcal{R}, \tag{26}$$

where $\mathcal{T}$ is $N \times M$ nonsymmetric positive definite matrix obtained from the difference diffusion operator (25) with boundary conditions included, and $\mathcal{R}$ is NM vector containing the right hand q and boundary function g. The system (26) can be solved using one of the iterative methods BICGSTAB or GMRES already discussed in section 4. In our numerical examples presented in these notes we used the preconditioned BICGSTAB with the diagonal as preconditioner.

As mentioned early the diffusion approach (24) is a good approximation to the full neutron transport equation (5) only when the transport field is optically thick ($\vartheta \gg 1$) or with dense absorption ($\kappa \gg 1$). In medium with small absorption or pure scattering ($\kappa = 0$) the diffusion approach (24) becomes unable to approximate accurately the correct solution of the full transport problem. Nevertheless, this approach can be used to accelerate the source iteration algorithm in all the regimes. The resulting accelerated algorithm, widely known in computational neutron transport as Diffusion Synthetic Acceleration (DSA) method, was first introduced in [2] and studied in a number of papers, for instance see [5, 3].

The implementation of DSA method to approximate the solution of the neutron transport equation (5) is carried out in the following algorithm

`Algorithm 4:` The DSA algorithm

given the initial guess $\Psi^{(0)}$ compute $\Phi^{(0)} = \dfrac{1}{4\pi}\displaystyle\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\Psi^{(0)}$

do $k = 0, \ldots, Kmax$

  do $l = 1, \ldots, \bar{N}$

   compute $\quad \mathbf{W} = \mathbf{Q}_l + \Sigma_l \Phi^{(k)}$

  end do

  call **sweeping**$(N, M, \bar{N}, \Delta x, \Delta y, \sigma, \kappa, \mu, \eta, \mathbf{Q}, \Psi^{(k+1)}, \mathbf{W})$

  compute $\Phi^{(k+\frac{1}{2})} = \dfrac{1}{4\pi}\displaystyle\sum_{l=1}^{\bar{N}} \omega_l \mathbf{S}\Psi_l^{(k+1)}$

  compute φ by solving the diffusion problem

$$\begin{aligned} -\nabla \cdot \left(\frac{1}{3(\sigma+\kappa)}\nabla\varphi\right) + \kappa\varphi &= \sigma\left(\Phi^{(k+\frac{1}{2})} - \Phi^{(k)}\right), \\ \varphi + \frac{2}{3(\sigma+\kappa)}\mathbf{n}\cdot\nabla\varphi &= 0. \end{aligned}$$

  set $\Phi^{(k+1)} = \Phi^{(k+\frac{1}{2})} + \varphi$

  compute $\mathbf{Res}^{(k+1)} = \|\Phi^{(k+1)} - \Phi^{(k)}\|_{L^2}$

  if $\left(\dfrac{\mathbf{Res}^{(k+1)}}{\mathbf{Res}^{(0)}} \le tol\right)$ stop

end do

Recall that in the matrix notation of section 4 the SI iteration is given by the Richardson iteration applied to the system (17) as

$$\mathbf{X}^{(k+1)} = \left(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}\right)\mathbf{X}^{(k)} + \mathbf{M}^{-1}\mathbf{b},$$

where $\mathbf{M}$ is the block lower triangle of $\mathbf{A}$. Roughly speaking, the DSA method can be viewed as preconditioned Richardson iteration with the diffusion matrix $\mathcal{T}$ like preconditioner,

$$\mathbf{X}^{(k+1)} = \left(\mathbf{I} - \mathcal{T}^{-1}\mathbf{A}\right)\mathbf{X}^{(k)} + \mathcal{T}^{-1}\mathbf{b},$$

and $\mathcal{T}^{-1}$ is obtained by solving the diffusion linear system (26).

Note that the first lines in `Algorithm 4` are similar to the `Algorithm 2`. However, the source iteration algorithm gives only the intermediate solution $\Phi^{(k+\frac{1}{2})}$ which has to be corrected by adding the solution φ obtained by the diffusion approach. Furthermore, if BICGSTAB is used for the diffusion approach, then an inner iterations have to be added to the iteration used by the SI algorithm and an outer SI iterations may require less accuracy from the inner BICGSTAB iterations (since the main issue to consider the DSA method is to obtain an accelerated algorithm).

6 Numerical Examples

Table 2: The values of σ, κ and boundary function g for different test problems in example 1.

	$\sigma(x,y)$	$\kappa(x,y)$	$g_{\Gamma_l}(y)$	$g_{\Gamma_r}(y)$	$g_{\Gamma_b}(x)$	$g_{\Gamma_t}(x)$
Test 1	0.99	0.01	0	1	x	x
Test 2	99	1	0	1	x	x
Test 3	1	10	y	$1-y$	x	$1-x$
Test 4	10	0	y	$1-y$	x	$1-x$

To asses the performance of the methods introduced in the previous sections we have run some numerical experiments of two examples for the neutron transport equation (5). In all our tests, the iterations in the numerical methods are terminated as soon as the inequality

$$\text{Relative Residual} := \frac{Res^{(k)}}{Res^{(0)}} \leq 10^{-6}, \tag{27}$$

is satisfied. Here $Res^{(0)}$ and $Res^{(k)}$ denote the initial residual and the residual at the iteration k in the iterative algorithm, respectively. We used the discrete L^2-norm for the computation of these residuals.

The convergence rates along with cross section plots of the results give a good ideas of the accuracy of the algorithms. The efficiency of the solvers is compared in the CPU time context. All the calculations reported in this section have been carried out in Fortran implementation with double precision on a PC with AMD-K6 200 processors.

The first example is the equations (5) in the unit square $\mathcal{D} = [0,1]\times[0,1]$ covered by 100×100 grid points and augmented with the following boundary function g

$$\begin{aligned} g(0,y,\Omega) &= g_{\Gamma_l}(y), \qquad g(1,y,\Omega) = g_{\Gamma_r}(y), \qquad \text{for} \quad 0 \leq y \leq 1; \\ g(x,0,\Omega) &= g_{\Gamma_b}(x), \qquad g(x,1,\Omega) = g_{\Gamma_t}(x), \qquad \text{for} \quad 0 \leq x \leq 1. \end{aligned}$$

We set $q(x,y,\Omega) = 0$. The coefficients σ, κ; the functions g_{Γ_l}, g_{Γ_r}, g_{Γ_b} and g_{Γ_t}, are chosen for four different test problems according to the table 2.

The main issues we wish to address in these test problems are concerned with the comparison on convergence and efficiency of all the methods presented in these notes using different values

Table 3: The number of iterations (# Iter) and the CPU time (in seconds) for SI, GMRES, BICGSTAB, DSA and Diffusion methods for the four test problems in example 1.

	Test 1		**Test 2**		**Test 3**		**Test 4**	
	# Iter	CPU	# Iter	CPU	# Iter	CPU	# Iter	CPU
SI	17	7.65	866	317.29	6	3.10	262	107.98
GMRES	4	5.51	50	51.14	2	3.68	16	17.43
BICGSTAB	4	3.61	52	28.12	3	3.12	14	8.68
DSA	7	19.97	21	14.84	4	4.01	8	14.27
Diffusion	224	3.48	58	0.94	60	0.9	154	2.33

of σ, κ and g to show the advantages of a method over the others. To this end we first plot in figure 4 the convergence rates for the four test problems. A log-scale on the y-axis is used. A first remark concerning these plots is that when $\gamma \approx 1$ the SI method converges slowly, for instance, in **Test 2** ($\gamma = 0.99$) SI needs 866 iterations to converge and in **Test 4** ($\gamma = 1$) needs 262 iterations. This fact was early ensured by theorem 1. However, in both tests, DSA method shows fast convergence over all the others methods. On the other hand, when $\gamma \ll 1$ the BICGSTAB method can compete with DSA. In **Test 1** and **Test 3**, a few iterations are enough for the convergence of all methods, but still SI method is the slowest.

In table 3 we display the number of iterations needed by each method for the four tests together with the consumed CPU time. It is clear that the BICGSTAB method uses less CPU time in all tests except in **Test 2** ($\vartheta = 100$). The diffusion results are also included in table 3, They are less CPU time consuming specially when $\vartheta \gg$ (**Test 2** and **Test 3**). However, the diffusion results should not be compared to other methods since the problem they solve has different structure than those solved by SI, BICGSTAB, GMRES, or DSA methods.

In figure 5 we plot the scalar flux ϕ obtained by DSA method for the four test problems. Similar results are plotted in figure 6 but using the diffusion procedure. The SI, BISCATAB and GMRES results are not presented here, because they overlap those obtained by DSA method. In order to compare these results, we show in figure 7 a cross section at the main diagonal ($y = x$) of the scalar flux obtained by all methods. As can be slightly seen the diffusion failed to approach accurately the DSA results when $\gamma = 0.99$; $\vartheta = 1$ (**Test 1**), and $\gamma = 0.09$; $\vartheta = 11$ (**Test 3**). In other two tests (**Test 2** and **Test4**), diffusion approach resolves the neutron transport equation correctly as the DSA method does, but with less computational effort referring to the CPU time in table 3.

Our second example consists of tests arising in radiative transfer problems. Usually the transport equation (5) is coupled to the heat equation to model radiative heat transfer phenomena, compare [19, 6, 10, 24] for detailed studies on radiative transfer. Since our goal in these notes is concerned with numerical tools for simulating the transport equation, we fix the temperature profile in the radiative transfer equation and we try to solve the transport equation coupled to this temperature profile. Thus, the problem statements we consider here are

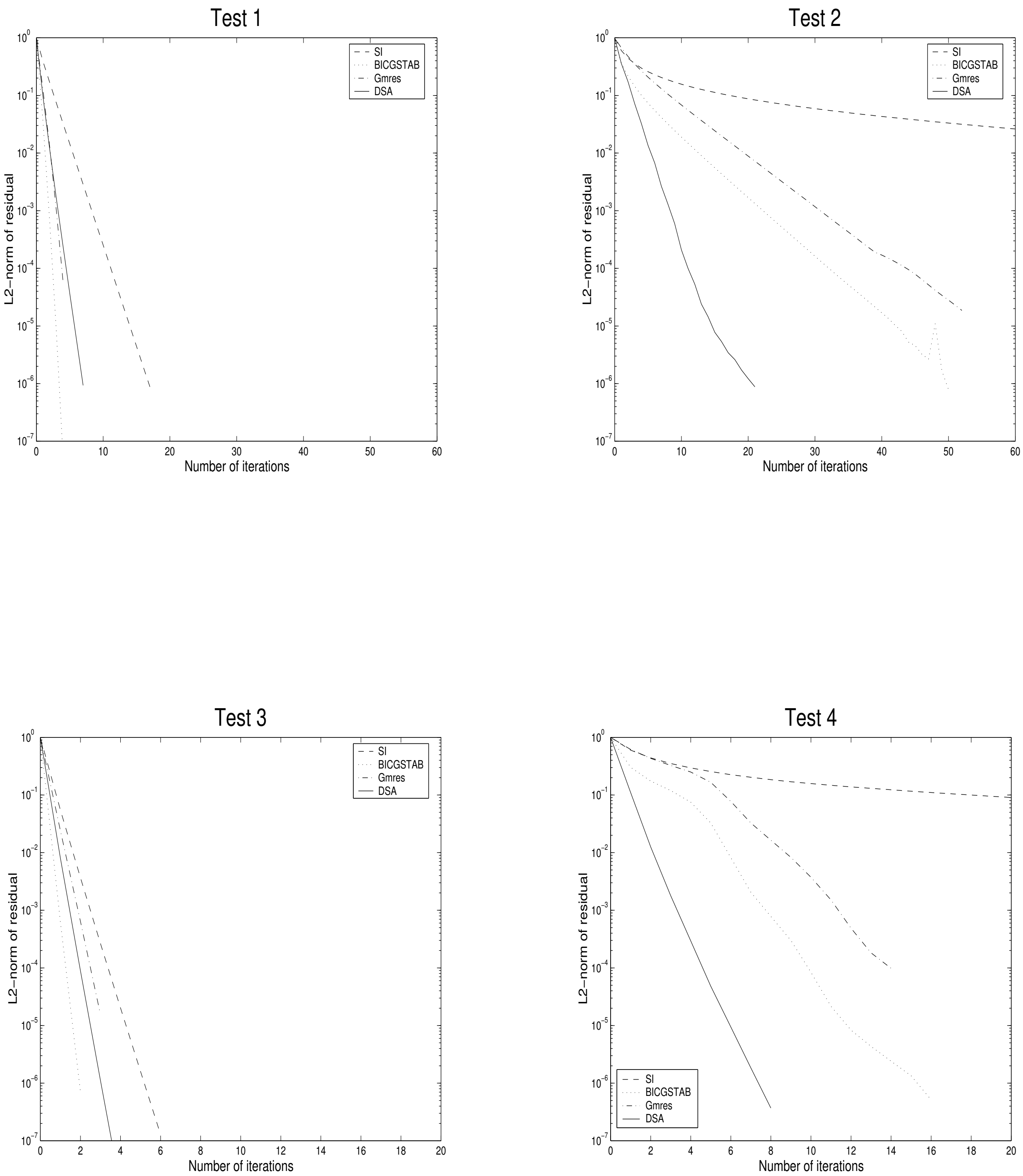

Figure 4: The convergence plots for the four test problems from table 2.

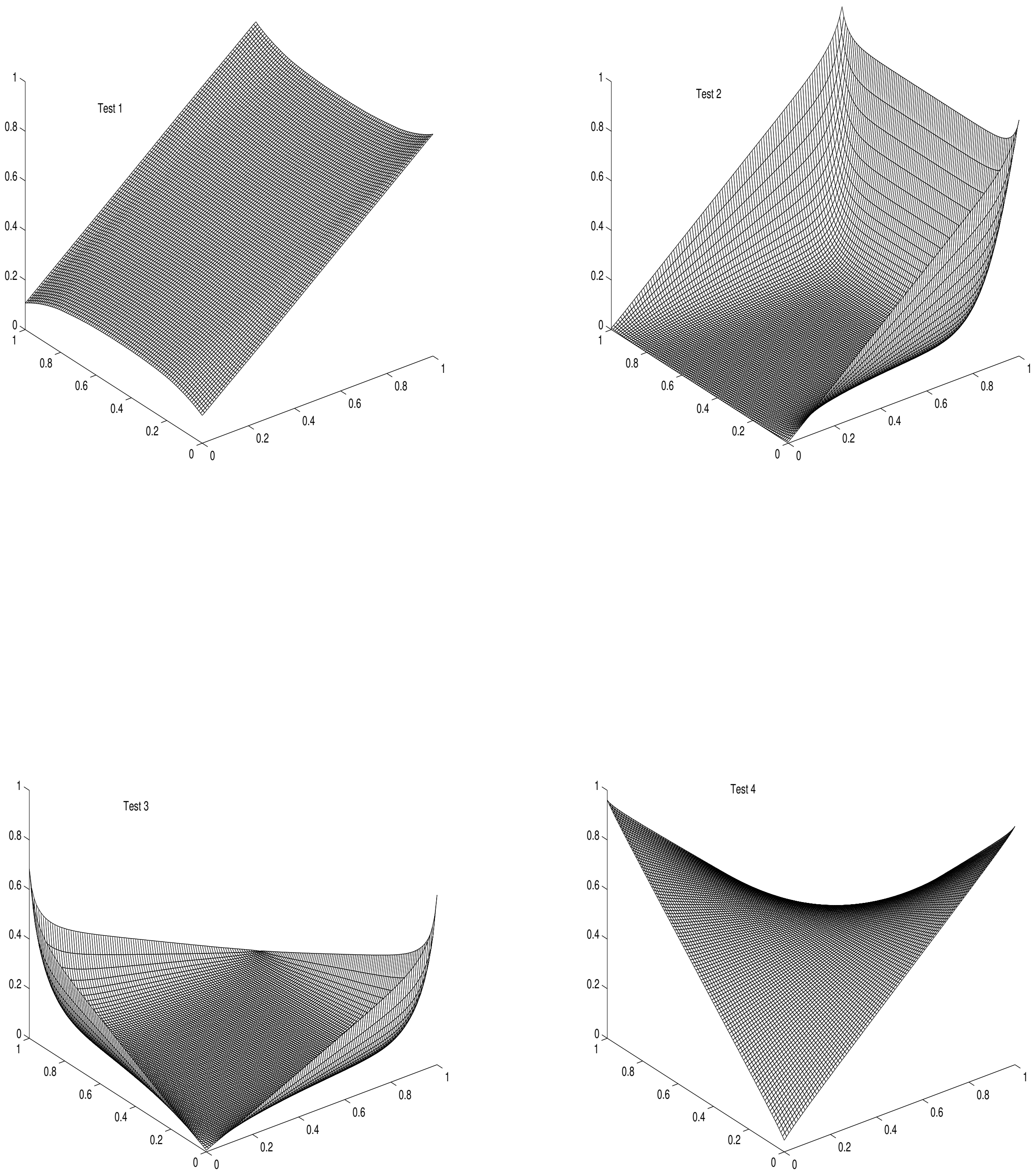

Figure 5: The scalar flux ϕ obtained by DSA method for the four test problems from table 2.

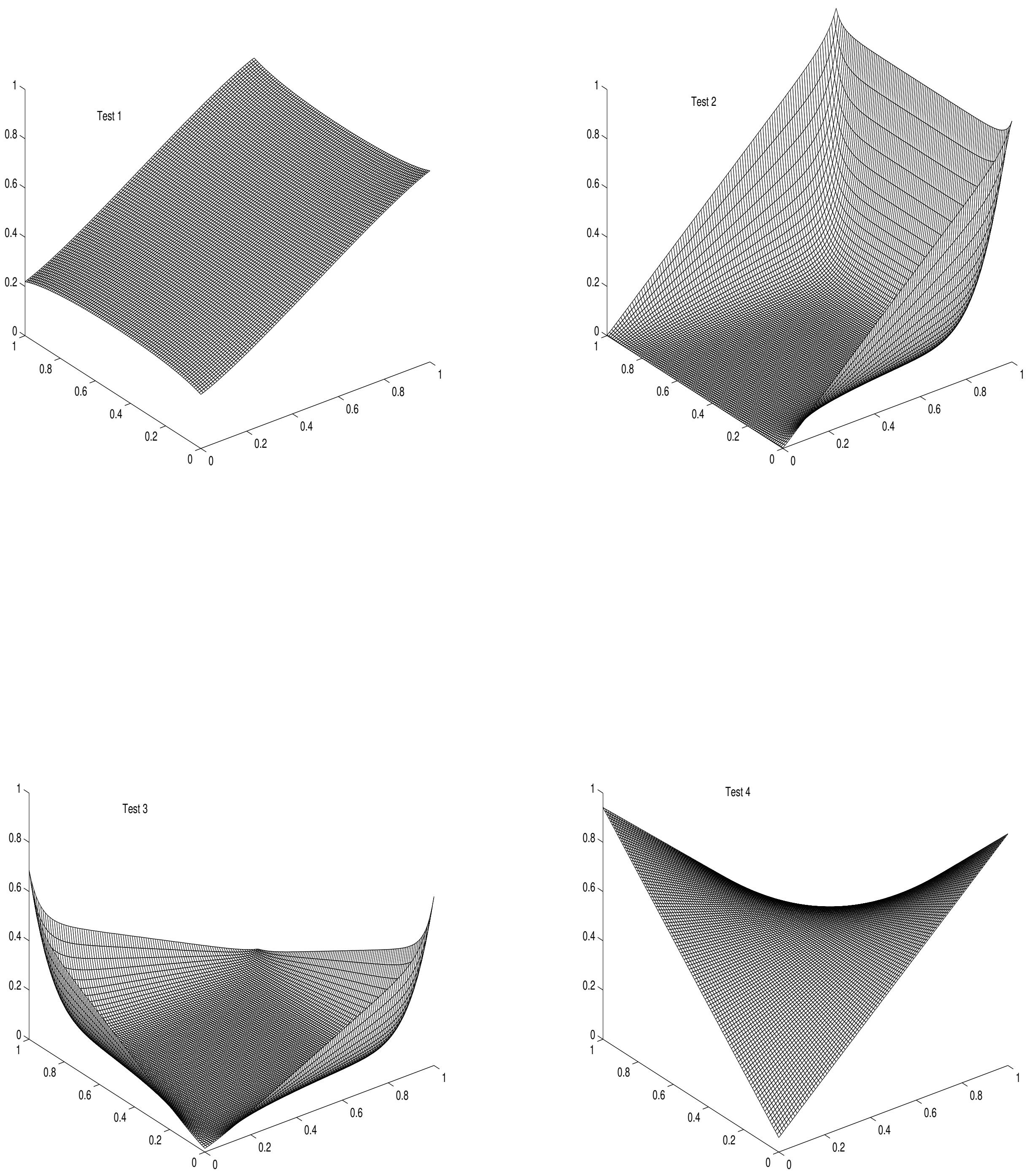

Figure 6: The scalar flux ϕ in the Diffusion approach for the four test problems from table 2.

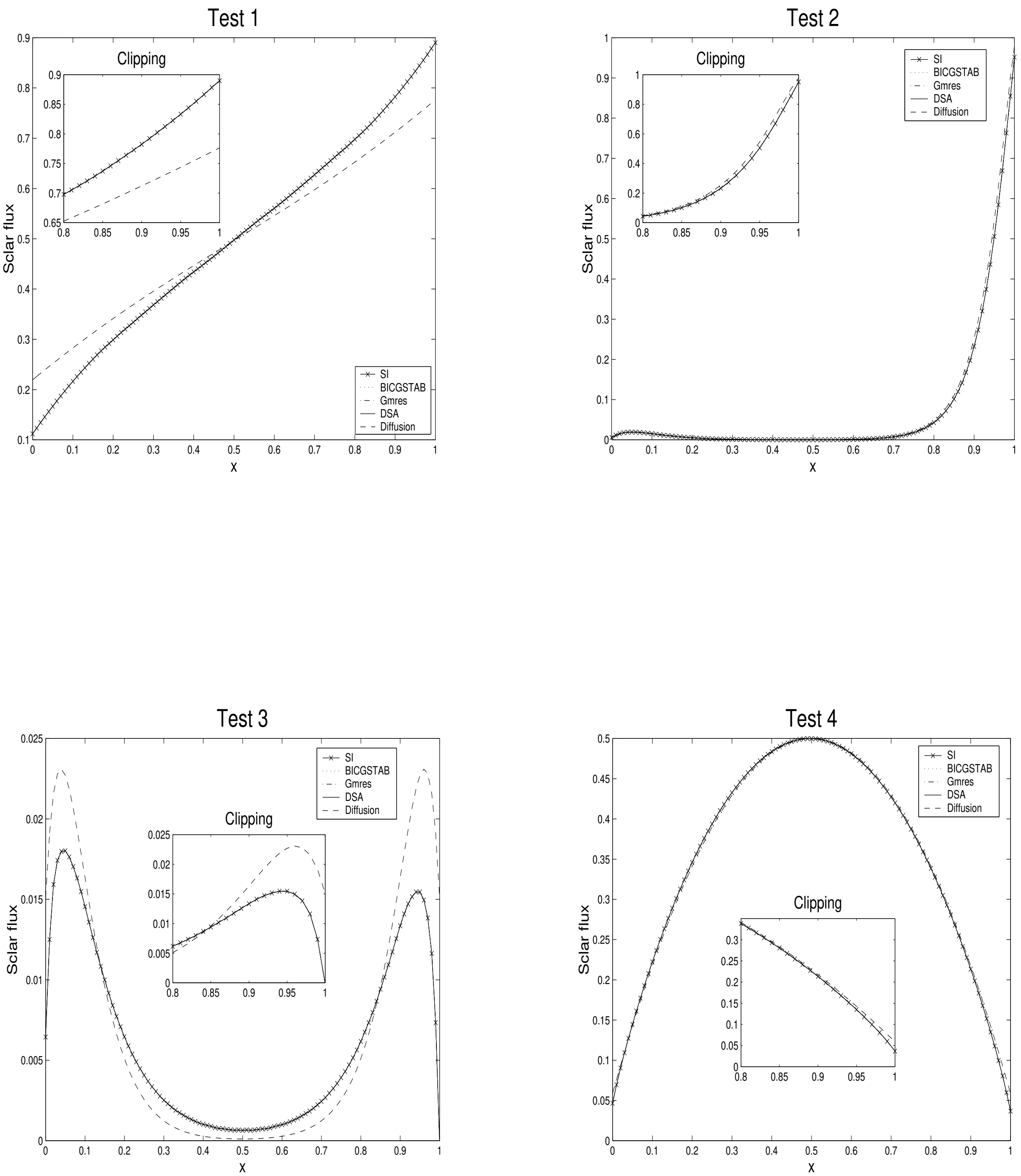

Figure 7: The cross section at $y = x$ of the scalar flux ϕ for the four test problems from table 2.

The frequency-independent problem

$$\Omega \cdot \nabla I + (\sigma + \kappa) I = \frac{\sigma}{4\pi} \int_{S^2} I(\mathbf{x}, \Omega') d\Omega' + \kappa B(T). \tag{28}$$

The frequency-dependent problem

$$\Omega \cdot \nabla I_\nu + \big(\sigma_\nu + \kappa_\nu\big) I_\nu = \frac{\sigma_\nu}{4\pi} \int_{S^2} I_\nu(\mathbf{x}, \Omega', \nu) d\Omega' + \kappa_\nu B(T, \nu). \tag{29}$$

Here $I_\nu = I(\mathbf{x}, \Omega, \nu)$, $T = T(\mathbf{x})$, $\sigma_\nu = \sigma(\mathbf{x}, \nu)$ and $\kappa_\nu = \kappa(\mathbf{x}, \nu)$ denote respectively, the radiation intensity, the temperature, the scattering and the opacity within the frequency $\nu > 0$. B is the Planck function given by

$$B(T, \nu) = \frac{2\hbar\nu^3}{c^2} \big(e^{\hbar\nu/k_B T} - 1\big)^{-1}, \tag{30}$$

where $\hbar$, k_B and c are Planck constant, Boltzmann constant and the speed of light, respectively.

Notice that, in the frequency-independent problem (28) the function $B = B(T) = a_R T^4$, with a_R is a radiation constant ($a_R = 1.8067.10^{-8}\ J/K$). The computational domain is a square of 1 cm side discretized into 100×100 grid cells. The temperature we used in our computations is a linear profile between 800 K and 1800 K in the unit square *i.e.*,

$$T(x, y) = 800x + 1000, \qquad (x, y) \in [0, 1] \times [0, 1].$$

Using this temperature profile we set the boundary conditions for the intensity according to the radiative equilibrium

$$I(\hat{\mathbf{x}}) = B(T(\hat{\mathbf{x}})), \qquad \hat{\mathbf{x}} \in \partial\mathcal{D}^-, \tag{31}$$

for the frequency-independent problem (28), and

$$I_\nu(\hat{\mathbf{x}}) = B(T(\hat{\mathbf{x}}), \nu), \qquad \hat{\mathbf{x}} \in \partial\mathcal{D}^-, \tag{32}$$

for the frequency-dependent problem (29).

First, we solve the grey problem (28)-(31) using the methods studied in the previous sections. In figure 8 we report the convergence plots for two different values of the absorption κ while the scattering is fixed to $\sigma = 1\ cm^{-1}$ in both tests. It is apparent that the convergence of SI method become slow when the scattering ratio γ change from 0.09 ($\kappa = 10\ cm^{-1}$) to 1 ($\kappa = 0\ cm^{-1}$). These results are in good agreement with theorem 1 in section 4. The accuracy plots given in figure 9 represent a cross section at $y = 0.5\ cm$ on the scalar flux obtained by all the methods with the diffusion approach included. As the opacity κ decreases, the diffusion results become slightly far from the results obtained for the full transport problem.

We now turn our attention to the frequency-dependent problem (29)-(32). In order to discretize the equations (29)-(32) respect to the frequency variable ν, we assume $\tilde{N}$ frequency bands $[\nu_\iota, \nu_{\iota+1}]$, $\iota = 1, \ldots, \tilde{N}$ with piecewise constant absorption

$$\kappa_\nu = \kappa_\iota, \qquad \forall\ \nu \in [\nu_\iota, \nu_{\iota+1}] \quad \iota = 1, \ldots, \tilde{N}.$$

We define the frequency-averaged intensity in the band $[\nu_\iota, \nu_{\iota+1}]$ by

$$I_\iota = \int_{\nu_\iota}^{\nu_{\iota+1}} I_{\nu'}(\mathbf{x}, \Omega, \nu') d\nu'. \tag{33}$$

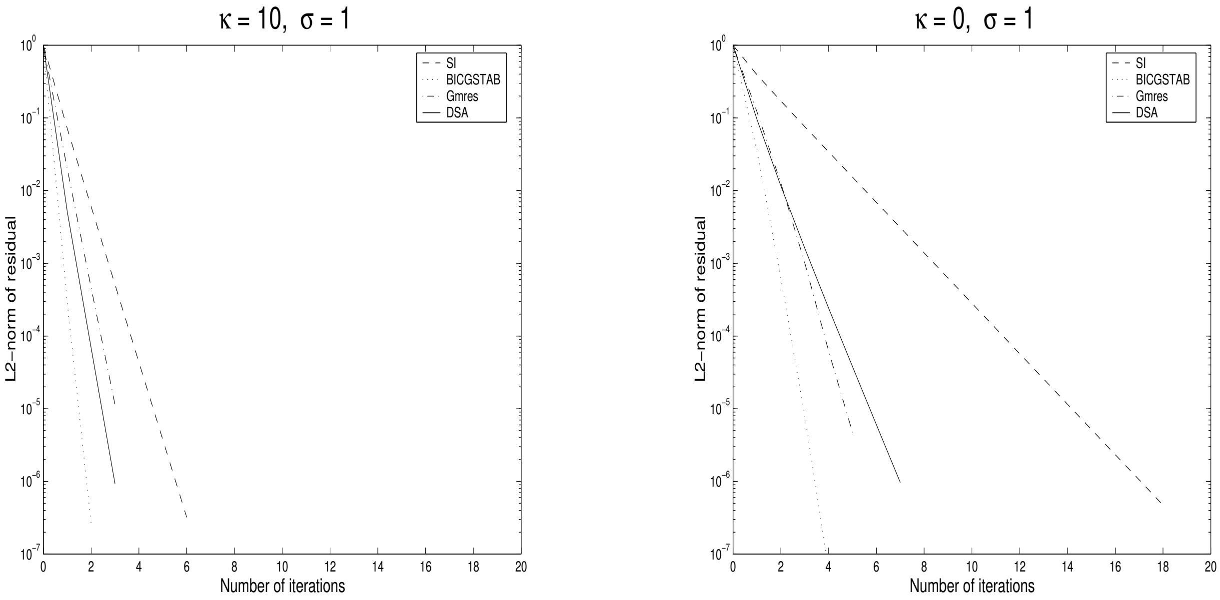

Figure 8: The convergence plots for the grey problem (28)-(31) with $\sigma = 1$ and two different values of opacity.

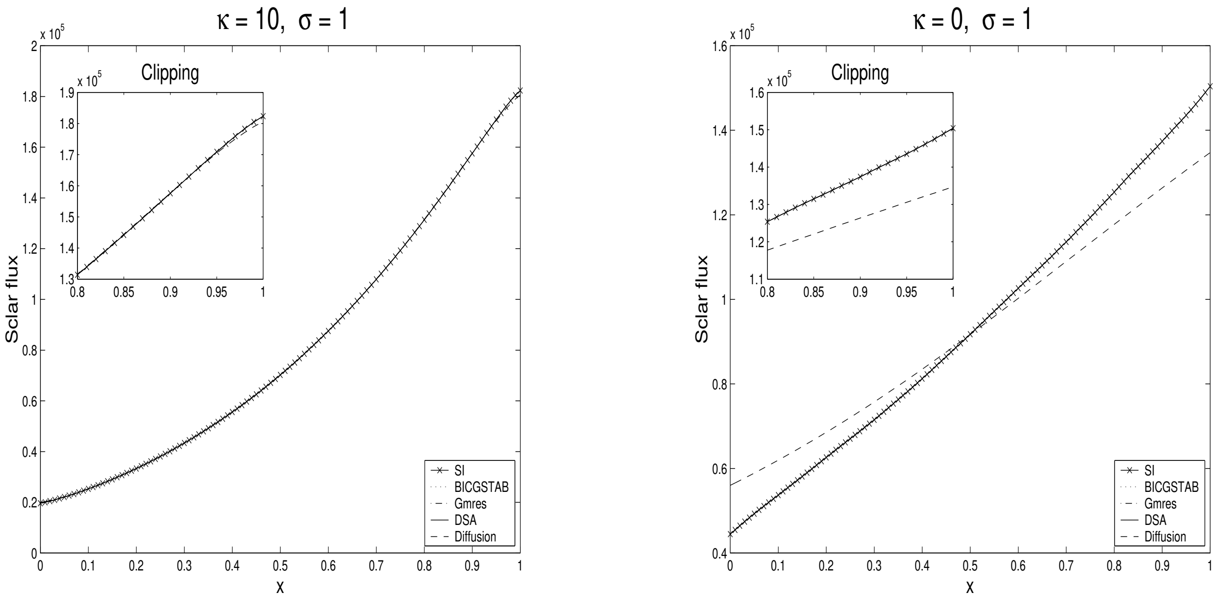

Figure 9: The cross section at $y = 0.5$ of the scalar flux ϕ for the grey problem (28)-(31) with $\sigma = 1$ and two different values of opacity.

Table 4: The bands used in the numerical simulation of the frequency-dependent problem.

Band ι	ν_ι (μm)	$\nu_{\iota+1}$ (μm)	κ_ι (m^{-1})
1	∞	5	0.40
2	5	0.3333	0.50
3	0.3333	0.2857	7.70
4	0.2857	0.2500	15.45
5	0.2500	0.2222	27.98
6	0.2222	0.1818	267.98
7	0.1818	0.1666	567.32
8	0.1666	0.1428	7136.06
	0.1428	0	opaque

Then, the equations (29)-(32) are transformed to a system of $\tilde{N}$ transport equations of the form

$$\begin{aligned} \Omega\cdot\nabla I_\iota + \left(\sigma_\iota + \kappa_\iota\right) I_\iota &= \frac{\sigma_\iota}{4\pi}\int_{S^2} I_\iota(\mathbf{x},\Omega',\nu_\iota)d\Omega' + \kappa_\iota \int_{\nu_\iota}^{\nu_{\iota+1}} B(T,\nu')d\nu', \\ I_\iota(\hat{\mathbf{x}}) &= \int_{\nu_\iota}^{\nu_{\iota+1}} B(T,\nu')d\nu', \qquad \hat{\mathbf{x}}\in\partial\mathcal{D}^-. \end{aligned} \tag{34}$$

Note that after the discretization of ordinates in $\bar{N}$ directions and the space in $N\times M$ gridpoints, one has to deal with systems with $\tilde{N}\times\bar{N}\times N\times M$ unknowns and, finding solutions to such systems requires much memory storage and much computational cost. In our numerical simulations we use eight frequency bands $[\nu_\iota,\nu_{\iota+1}]$, $\iota = 1,\dots,8$ given in table 4. These values are frequently used in the glass manufacturing, we refer to [24] for more physical details.

Using two different values for the scattering ($\sigma = 1\ cm^{-1}$ and $\sigma = 100\ cm^{-1}$), we summarize in table 5 the CPU time and the number of iterations used by all methods except the BICSTAB method, because their results are identical to the GMRES ones. It is important to mention two points with respect to the results in table 5. First, we observe that by decreasing the scattering ratio γ and keeping σ fixed to 100 cm^{-1} or 1 cm^{-1} the number of iterations reduce asymptoticly in all the methods with the advantage of the GMRES method over the others. Second, when $\sigma = 100\ cm^{-1}$ the SI method required unreasonable number of iterations for the first frequency bands, consequently the CPU time used is very large. In contrast, the Diffusion approach uses only 0.012% of the CPU time used by SI method for this case, and the results obtained by both approaches are similar, see figure 10.

In order to quantify the solution of (34) we define the frequency-mean scalar flux φ as

$$\begin{aligned} \varphi(\mathbf{x}) &= \frac{1}{4\pi}\int_{S^2}\int_0^\infty I(x,\Omega',\nu')d\Omega' d\nu' \\ &= \frac{1}{4\pi}\sum_{l=1}^{\bar{M}}\sum_{\iota=1}^{\tilde{N}} I_{\iota,l}(\mathbf{x}). \end{aligned}$$

Table 5: The number of iterations and the CPU time (in minutes) for SI, GMRES, DSA and Diffusion methods for the eight frequency-bands problem with two different values of σ.

	Band ι	Scattering ratio γ	SI	GMRES	DSA	Diffusion
$\sigma = 1\ cm^{-1}$	1	0.71428	16	6	7	217
	2	0.66666	15	6	7	212
	3	0.11494	8	4	4	91
	4	0.06077	6	4	4	46
	5	0.03450	6	4	3	25
	6	0.00371	4	2	3	4
	7	0.00175	4	2	3	3
	8	0.00014	3	1	2	2
	CPU	——	25.63	3.94	14.69	0.21
$\sigma = 100\ cm^{-1}$	1	0.99601	1700	92	32	87
	2	0.99502	1321	89	30	82
	3	0.92850	178	26	29	25
	4	0.86617	95	18	29	17
	5	0.78137	57	14	27	11
	6	0.27175	12	5	9	3
	7	0.14985	9	4	6	3
	8	0.01381	5	2	3	2
	CPU	——	981.16	6.43	69.72	0.12

The figure 10 shows a cross section of φ at $y = 0.5\ cm$ for the two values of σ. The main message taken from this figure is that, the diffusion results coincides with the transport results only when the scattering is large ($\sigma = 100\ cm^{-1}$) and for this case the SI scheme is unreasonably slow (compare the CPU time in table 5). Therefore, it is worth to use the diffusion approach because, at least for this test problem, it gives results that are as accurate as those obtained for the full transport equation, but with less computational cost.

7 Conclusions

We have combined the discrete ordinates collocation and the Diamond differencing to reconstruct numerical methods for the two-dimensional neutron transport equation. These methods include the source iteration scheme, full BICGSTAB and GMRES algorithms, and the diffusion synthetic acceleration method. We have compared the results obtained by these methods on several test problems. The principal conclusions achieved through this comparison are the following:

1. For neutron transport equation with small scattering ratio ($\gamma \ll 1$) and moderate optical coefficient ϑ, the SI method can be a reasonable solver, but still not efficient enough as BICGSTAB, GMRES or DSA methods.

2. For neutron transport equation with large or pure scattering ($\gamma \approx 1$), the SI method become very slow and loses efficiency. In parallel, the DSA method is the best and presents

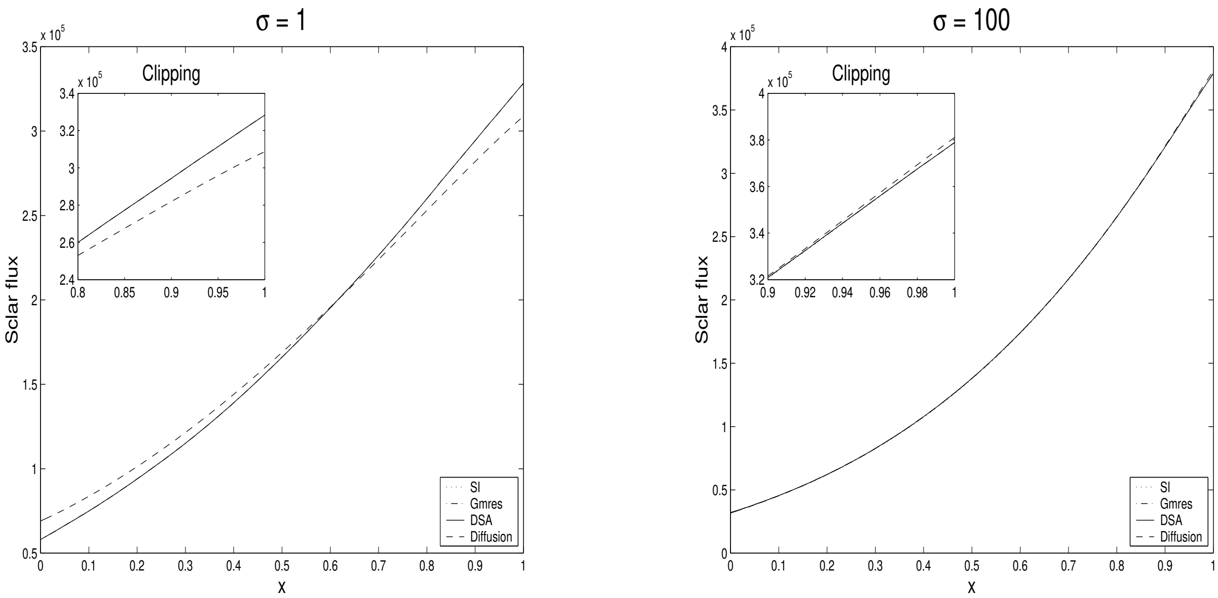

Figure 10: The cross section at $y = 0.5$ of the mean scalar flux φ for the problem (29)-(32) with the eight frequency-bands given in table 5 and two different values of σ.

fast convergence rate over all other methods.

3. For neutron transport equation in optically thick regime ($\vartheta \gg 1$), the diffusion approach may be a valid alternative for the iterative methods since it gives results that are as accurate as those obtained by DSA method, but with less computational cost, and diffusion approach does not need extra discretization for the angular directions.

Nevertheless, borrowing the idea of simplified P_N approximations to the transport equation and following the argument of section 5, it is feasible to devise generalized preconditioners to the SI method with high accelerated convergence. Such methods can be used in radiative heat transfer and radiation hydrodynamic couplings rather than the transport equation (1). Results on these methods will be reported in the near future.

We want to point out that general time-dependent neutron transport problem (1) can also numerically solved in a similar manner. By using the discrete ordinates and the Diamond differencing methods, and by using the same notations as in section 3, the equations (1) are transformed to the following system of ODE's

$$\begin{aligned} \frac{1}{v}\frac{d}{dt}\psi_{l,i+\frac{1}{2}j+\frac{1}{2}} + \mu_l \frac{\psi_{l,i+1j} - \psi_{l,ij}}{(\Delta x)_{i+\frac{1}{2}}} + \eta_l \frac{\psi_{l,ij+1} - \psi_{l,ij}}{(\Delta y)_{j+\frac{1}{2}}} \quad + & \quad \left(\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}}\right)\psi_{l,i+\frac{1}{2}j+\frac{1}{2}} = \\ & \quad \sigma_{l,i+\frac{1}{2}j+\frac{1}{2}}\phi_{i+\frac{1}{2}j+\frac{1}{2}} + q_{l,i+\frac{1}{2}j+\frac{1}{2}}, \\ \psi_{l,ij}(t) \quad = & \quad g_{l,ij}(t), \\ \psi_{l,ij}(0) \quad = & \quad \psi^0_{l,ij}, \end{aligned} \tag{35}$$

where each centred valued function $f_{l,i+\frac{1}{2}j+\frac{1}{2}}$ appeared in (35) is given by

$$f_{l,i+\frac{1}{2}j+\frac{1}{2}} = \frac{f_{l,ij} + f_{l,i+ij} + f_{ij+1} + f_{l,i+1j+1}}{4}.$$

For the time integration of (35), one can use any ODE solver, however the presence of the term $1/v$ in the front of the time-derivative operator, makes the use of explicit schemes inefficient, because these explicit schemes are subject to a CFL condition of the form

$$\lambda := v\frac{\Delta t}{h} \leq 1, \tag{36}$$

where $h := \max\limits_{ij}\big((\Delta x)_i, (\Delta y)_j\big)$ is the mesh size, Δt is the time stepsize and v is the neutron speed (extremely large, order of speed of light). Therefore, implicit schemes which alleviate the stability restriction (36), should be used. For simplicity, we consider here the implicit Euler method to integrate the equations (35).

Let the time interval $[0, T]$ be divided into NT subintervals $[t_n, t_{n+1}]$ of length Δt such that $t_n = n\Delta t$ and $T = NT\Delta t$. We use the notation $W^n_{l,ij}$ to denote the value of the function W at $(t_n, \mu_l, \eta_l, x_i, y_j)$. Then, the fully discrete formulation of the equation (1) can be written as

$$\begin{aligned}\mu_l\frac{\psi^{n+1}_{l,i+1j} - \psi^{n+1}_{l,ij}}{(\Delta x)_{i+\frac{1}{2}}} + \eta_l\frac{\psi^{n+1}_{l,ij+1} - \psi^{n+1}_{l,ij}}{(\Delta y)_{j+\frac{1}{2}}} \quad &+ \quad \left(\sigma^{n+1}_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa^{n+1}_{i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}\right)\psi^{n+1}_{l,i+\frac{1}{2}j+\frac{1}{2}} = \\ &\sigma^{n+1}_{l,i+\frac{1}{2}j+\frac{1}{2}}\phi^{n+1}_{i+\frac{1}{2}j+\frac{1}{2}} + q^{n+1}_{l,i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}\psi^{n}_{l,i+\frac{1}{2}j+\frac{1}{2}}.\end{aligned} \tag{37}$$

Once again, the discrete equation (37) can be reformulated in matrices as in (14) by using the following new matrix entries

$$\begin{aligned}
d_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}}{4},\\
\bar{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}}{4},\\
\underline{e}_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}}{4},\\
e_{l,i+\frac{1}{2}j+\frac{1}{2}} &:= \frac{-|\mu_l|}{2(\Delta x)_{i+\frac{1}{2}}} + \frac{-|\eta_l|}{2(\Delta y)_{j+\frac{1}{2}}} + \frac{\sigma_{i+\frac{1}{2}j+\frac{1}{2}} + \kappa_{i+\frac{1}{2}j+\frac{1}{2}} + \frac{1}{v\Delta t}}{4}.
\end{aligned}$$

By doing so and changing the right hand side as in (37), the SI, BICGSTAB, GMRES, DSA methods and the Diffusion approach studied in the previous sections remain valid to solve the problem (37) in the same way as have been done for the time-independent problem (11) with the only difference that another loop must be added for the time integration. Furthermore, the convergence rate of the source iteration is governed, at each time step, by the new scattering ratio

$$\gamma(t_n) := \max_{\mathbf{x}\in\mathcal{D}}\left(\frac{\sigma(t_n, \mathbf{x})}{\sigma(t_n, \mathbf{x}) + \kappa(t_n, \mathbf{x}) + \frac{1}{v\Delta t}}\right), \qquad n = 1, \ldots, \tilde{N}.$$

We would like to mention that, the space discretization used in these notes is second order. Therefore, to be consistent that the fully dicrestized scheme maintain the same order of accuracy, a second order time integration scheme should be used. For example, Crank-Nicolson method can be a good candidate, since it can be formulated easily as (37) and the resulting linear systems have the same structures as those obtained by Euler method.

Acknowledgements. I would like to thank Professor Axel Klar for his guidelines during the preparation of these notes. I am also grateful to Guido Thömmes for helpful discussions. I want to acknowledge the faithful conversations with Thomas Götz during my visits to Department of Mathematics, university of Kaiserslautern. This work was supported by Sonderforschungabereich 568 and the GraduiertenKolleg at Darmstadt university.

References

[1] M. L. Adams, E. W. Larsen, "Fast Iterative Methods for Deterministic Particle Transport Computations," Preprint.

[2] R. E. Alcouffe, "Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete-Ordinates Equations," Nucl, Sci, Eng. **64**, 344–355 (1977).

[3] S. F. Ashby, P. N. Brown, M. R. Dorr, A. C. Hindmarsh, "A Linear Algebraic Analysis of Diffusion Synthetic Acceleration for the Boltzmann Transport Equations," SIAM. J. Numer. Anal. **32**, 128–178 (1995).

[4] L. L. Briggs, W. F. Miller Jr, E. E. Lewis, "Ray effect Mitigation in Discrete Ordinate-Like Angular Finite Element Approximations in Neutron Transport," Nucl, Sci, Eng. **57**, 205–215 (1975).

[5] P. N. Brown, "A Linear Algebraic Development of Diffusion Synthetic Acceleration for Three-Dimensional Transport Equations," SIAM. J. Numer. Anal. **32**, 179–214 (1995).

[6] S. Chandrasekhar, "Radiative Transfer," Oxford Univ. Press, London, 1950.

[7] J. Cui "Finite Pointset Methods on the Sphere and Their Application in Physical Geodesty," Ph.D. Thesis, Dept. of Math, University Kaiserslautern (1995).

[8] W. A. Fiveland "The Selection of Discrete Ordinate Quadrature Sets for Anisotropic Scattering," ASME HTD. Fundam. Radiat. Heat Transfer **160**, 89–96 (1991).

[9] G. H. Golub, C. F. Van Loan, "Matrix Computations," The Johns Hopkins University Press, Baltimore and London, Third edition, 1996.

[10] Th. Götz "Coupling Heat Conduction and radiative Transfer," J. Quantitative Spectroscopy & radiative Transfer **72**, 57–73 (2002).

[11] A. Greenbaum, "Iterative Methods for Solving Linear Systems," SIAM. Philadelphia, 1997.

[12] C. Johnson, J. Pitkäranta, "Convergence of a Fully Discrete Scheme for Two-Dimensional Neutron Transport," SIAM. J. Numer. Anal. **20**, 951–966 (1983).

[13] C. T. Kelly, "Iterative Methods for Linear and Nonlinear Equations," SIAM. Philadelphia, 1995.

[14] K. D. Lathrop, "Remedies for Ray Effects," Nucl, Sci, Eng. **45**, 255–265 (1968).

[15] E. W. Larsen, G. Thömmes, A. Klar, " Simplified P_N Approximations to the Equations of Radiative Heat Transfer in Glass I: Modelling", Preprint.

[16] E. W. Larsen, J. E. Morel, "Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes II," J. Comput. Physics **83**, 212–236 (1989).

[17] E. W. Larsen, J. E. Morel, W. F. Miller Jr, "Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes," J. Comput. Physics **69**, 283–324 (1987).

[18] E. E. Lewis, W. F. Miller Jr. "Computational Methods of Neutron Transport," John Wiley & Sons, New York, 1984.

[19] D. Mihalas, B. W. Mihalas, "Foundations of Radiation Hydrodynamics," Oxford Univ. Press, New York (1984).

[20] W. F. Miller Jr, W. H. Reed, "Ray Effect Mitigation Methods for Two-dimensional Neutron Transport Theory," Nucl, Sci, Eng. **62**, 391–403 (1977).

[21] J. Pitkäranta, "On the Spatial Differencing of the Discrete Ordinate Neutron Transport Equation," SIAM. J. Numer. Anal. **15**, 859–869 (1980).

[22] J. Pitkäranta, L. R. Scott, "Error Estimates for the Combined Spatial and Angular Approximations of the Transport Equation for Slab Geometry," SIAM. J. Numer. Anal. **20**, 922–950 (1983).

[23] Y. Saad, M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," SIAM. J. Sci. Statist. Comput. **7**, 856–869 (1986).

[24] G. Thömmes, "Radiative Heat Transfer Equations for Glass Cooling Problems: Analysis and Numerics," Ph.D. Thesis, TU Darmstadt, (2002).

[25] H. A. van der Vorst, "BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems," SIAM. J. Sci. Statist. Comput. **13**, 631–644 (1992).

Appendix A: The Ray Effect

The discrete ordinates method described in section 2 solves the transport equation (5) for a close set of angular directions $S_{\bar{N}}$, as long as it approximates the angular integral by a weighted sum of flux values. Of course, in two dimensional problems not all the possible directions in the unit sphere S^2 are calculated, but the set is reduced by means of symmetry relations. Moreover, in a three dimensional transport field projected in x-y plane, some directions appear to be between others when they really are not.

One of the main drawbacks of the discrete ordinates method for solving the neutron transport equation is that due to the existence of privileged directions in $S_{\bar{N}}$ set, the solution has some degree of numerically induced anisotropy (by anisotropy we mean that the probability of scattering for the particles is not the same for all directions). In cases with very little if any scattering, and localized external sources, this effect may strongly disturb the correct solution and become worse with sets of few angular directions. This undesirable effect is called the ray effect. As mentioned in [18], the ray effects occur normally in two-dimensional problems where the external sources are localized and the effects of scattering are relatively small.

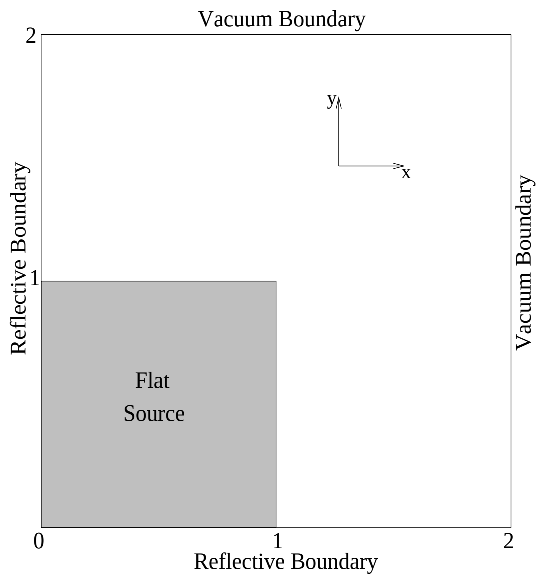

Figure 11: The model problem used to exhibit ray effects.

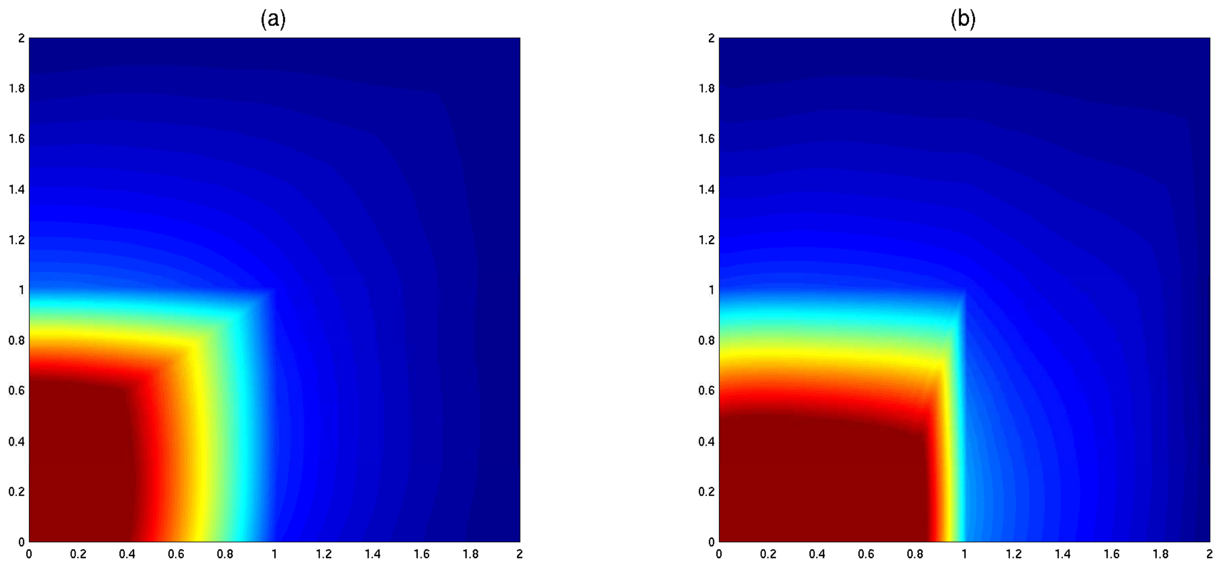

Figure 12: The angular flux ψ along the direction **(a)** $\Omega_7 = (-0.79263, -0.51491)$ and **(b)** $\Omega_{23} = (-0.21215, -0.72707)$.

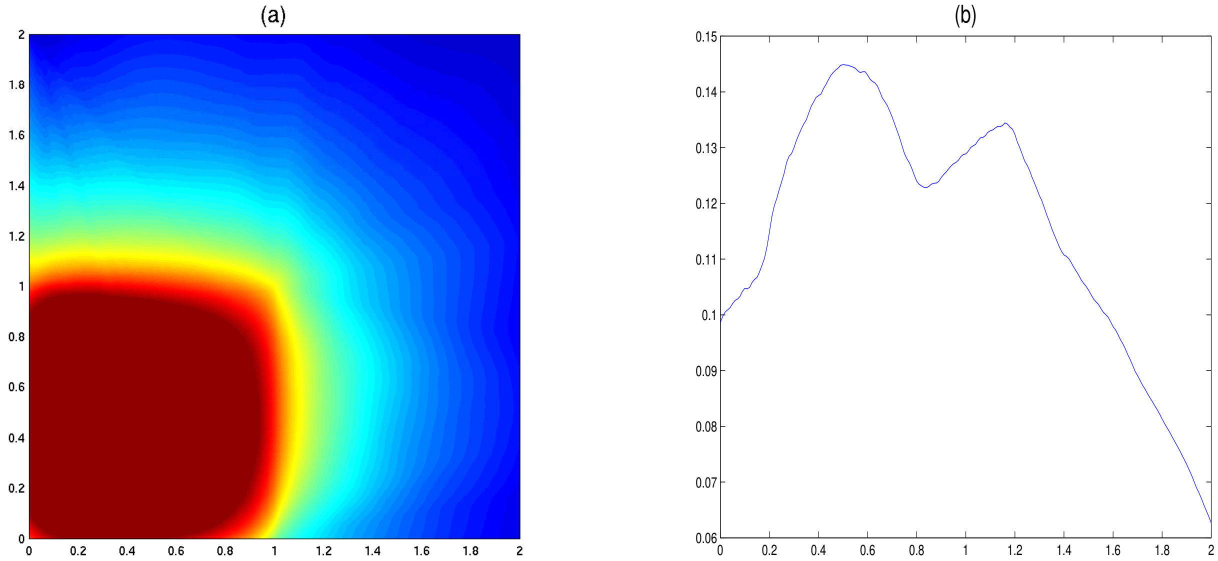

Figure 13: The scalar flux ϕ **(a)** and its corresponding section at $x = 1.967$ **(b)**.

In order to make the ray effect visible we use in this appendix, the same model problem from [18] chosen by the authors to exhibit the ray effect. The model consists of the equations (5) with a flat source q given by

$$q(x, y) = \begin{cases} 1, & \text{if} \quad 0 \le x \le 1 \quad \text{and} \quad 0 \le y \le 1, \\ 0, & \text{else.} \end{cases}$$

The domain geometry and the boundary conditions used for this test are shown in figure 11. Both the square source and the surrounding medium have the same optical properties. The scattering and the absorption parameters are those used in [18]. Thus, $\sigma = 0.5 \text{ cm}^{-1}$ and $\kappa = 0.25 \text{ cm}^{-1}$. The angular and the scalar fluxes are computed using the DSA method on 200×200 gridpoints and C-60 set directions. First we plot in figure 12 the angular flux corresponding to the two directions $\Omega_7 = (-0.79263, -0.51491)^T$ and $\Omega_{23} = (-0.21215, -0.72707)^T$. The colormap shows the flux field ψ in the domain, the ray effect is clearly visible as irregularities in what should be squared isocontours. Another exhibition of the ray effect is shown in figure 13 where the scalar flux ϕ and a section at $x = 1.967$ are plotted as suggested in [18]. The ray effects may be seen from the figure 13 as nonphysical oscillations.

There are many ways to overcome the disadvantage of the ray effects from a given computational neutron transport code. The simplest way is to refine the discrete ordinates set by increasing the number of angular directions. Then, the frequency of the oscillations becomes higher and the magnitude becomes lower. The ray effects can be completely eliminated by using the so-called P_N approximations. In contrast to discrete ordinates methods the P_N methods consist of expansion of the angular flux in the first $N+1$ Legendre polynomial. An extended references on these methods and other techniques to remedy the ray effects in the discrete ordinates collocation can be found in [18, 14, 4, 20].

Appendix B: A Fortran Code

For completeness, we include in this appendix a Fortran code for the SI algorithm to solve the neutron transport equation (5). The code links both the `algorithm 1` and `algorithm 2` to handled the linear system (15). We used the vacuum boundary conditions along the whole domain. However, these boundary conditions together with the scattering and the absorption coefficients can be changed very easily in their correspondent functions at the end of the listing code.

We want to point out that the code is not written to be a show of efficiency and optimized programming. It is simply to show for the reader how a source iteration code can be done.

```
      subroutine SI(nd,nx,ny,np,x,y,hx,hy,dx,dy,wg,aflux,
     &              sflux,q,work,tol,maxits,iout)
c
c This subroutine solve the Neutron transport problem (15)
c using the source iteration method. We assume the inflow
c boundary condition on all the bondaries of the computational
c domain. Discrete ordinates and Diamond differencing are used
c for angle and space discretizations.
c---------------------
c The input variables:
c---------------------
c  nd     = the number of directions in the unit sphere
c  nx     = the number of the space gridpoints in the x-direction
c  ny     = the number of the space gridpoints in the y-direction
c  np     = the total number of the space gridpoints. (np = nx x ny)
c  hx     = the space gridsize in the x-direction.(hx is a vector of size nx)
c  hy     = the space gridsize in the y-direction.(hy is a vector of size ny)
c  x      = the space gridpoints in the x-direction.
c           (x is a vector of size nx+1)
c  y      = the space gridpoints in the y-direction.
c           (y is a vector of size ny+1)
c  dx     = the angle direction in the x-direction.(dx is a vector of size nd)
c  dy     = the angle direction in the y-direction.(dy is a vector of size nd)
c  wg     = the weight associated to the directions dx and dy
c  aflux  = the angular flux is used as initial guess for the SI method
c           can be seted to zero unless for vacuum boundary conditions
c           are used. (aflux is a vector of size nd x nx+1 x ny+1)
c  q      = the source term. (q is a vector of size nd x nx+1 x ny+1)
c  work   = the work vector. (work is a vector of size nd)
c  tol    = the tolerance to stop the iterations.
c  maxits = the maximum number of iterations allowed
c  iout   = the output unit number for printing intermediate results
c           if(iout.le.0) no statiscts are printed
c---------------------
c The ouput variables:
c---------------------
c  aflux  = the current angular flux.
```

```
c           (aflux is a vector of size nd x nx+1 x ny+1)
c  sflux  = the scalar flux. (sflux is a vector of size np)
c------------------------
c The auxilary parameters:
c------------------------
c  wsflux          = the vector to store sclar fux at the previous iteration.
c                   (wsflux is a vector of size np)
c  s               = the vector to store source term.
c                   (s is a vector of size nx x ny)
c  bcl,bcr,bcb,bct = the vectors to store boundary conditions for the sweeping
c                    subroutine. (bcl,bcr are vectors of size ny+1 and
c                    bcb,bct are vectors of size nx+1)
c--------------------
c The functions used:
c--------------------
c  xkappa(x,y)   = the absorption function
c  sigma(x,y)    = the scattering function
c  bcleft(y)     = the left boundary function
c  bcright(y)    = the righ boundary function
c  bcbottom(x)   = the bottom boundary function
c  bctop(x)      = the top boundary function
c  xint(nd,wg,w) = the function to compute the weighted integrals
c---------------------
c The subroutine used:
c---------------------
c                      the sweeping subroutine
c  sweeping(nd,nx,ny,np,x,y,hx,hy,dx,dy,s,bcl,bcr,bcb,bct,aflux)
c=====================================================================
      implicit real*8(a-h,o-z)
      real*8   hx(1),hy(1),x(1),y(1),dx(1),dy(1),wg(1),sflux(1),work(1)
      real*8   aflux(nd,nx+1,ny+1),wsflux(np),q(nd,nx+1,ny+1)
      real*8   bcl(ny+1),bcr(ny+1),bcb(nx+1),bct(nx+1),s(nx,ny)
      data     pi/3.14159265358979d0/
c--------------------------------
c The initial scalar flux guess
c--------------------------------
      do 10 i=1,nx
       do 10 j=1,ny
        do 5 l=1,nd
         work(l) = 0.25d0*(aflux(l,i+1,j)+aflux(l,i,j)+
     &             aflux(l,i,j+1)+aflux(l,i+1,j+1))
 5      continue
         wsflux((i-1)*ny+j) = xint(nd,wg,work)
 10   continue
c----------------------------
c Strat the iterative process
c----------------------------
```

```
      do 85 its=1,maxits
c
c Compute the right hand side
c
       do 60 l=1,nd
        do 15 i=1,nx
         do 15 j=1,ny
          ct     = sigma(x(i),y(j))/(4d0*pi)
          s(i,j) = ct*wsflux((i-1)*ny+j)+q(l,i,j)
 15     continue
        if(dx(l).lt.0d0.and.dy(l).lt.0d0) then
         do 20 j=1,ny+1
          bcr(j) = bcright(y(j))+q(l,nx+1,j)
 20      continue
         do 25 i=1,nx+1
          bct(i) = bctop(x(i))+q(l,i,ny+1)
 25      continue
        elseif(dx(l).lt.0d0.and.dy(l).gt.0d0) then
         do 30 j=1,ny+1
          bcr(j) = bcright(y(j))+q(l,nx+1,j)
 30      continue
         do 35 i=1,nx+1
          bcb(i) = bcbottom(x(i))+q(l,i,1)
 35      continue
        elseif(dx(l).gt.0d0.and.dy(l).lt.0d0) then
         do 40 j=1,ny+1
          bcl(j) = bcleft(y(j))+q(l,1,j)
 40      continue
         do 45 i=1,nx+1
          bct(i) = bctop(x(i))+q(l,i,ny+1)
 45      continue
        else
         do 50 j=1,ny+1
          bcl(j) = bcleft(y(j))+q(l,1,j)
 50      continue
         do 55 i=1,nx+1
          bcb(i) = bcbottom(x(i))+q(l,i,1)
 55      continue
        endif
 60    continue
c---------------------------------------------------
c Call the sweeping to compute the new angular flux
c---------------------------------------------------
       call sweeping(nd,nx,ny,np,x,y,hx,hy,dx,dy,s,bcl,
     &               bcr,bcb,bct,aflux)
c----------------------------
c Compute the new scalar flux
```

```
c-----------------------------
      do 70 i=1,nx
       do 70 j=1,ny
        do 65 l=1,nd
         work(l) = 0.25d0*(aflux(l,i+1,j)+aflux(l,i,j)+
     &            aflux(l,i,j+1)+aflux(l,i+1,j+1))
 65     continue
        sflux((i-1)*ny+j) = xint(nd,wg,work)
 70    continue
c----------------------
c Compute the residual
c----------------------
      res = 0.0d0
      do 75 k=1,np
       res = res+dabs(sflux(k)-wsflux(k))**2
 75    continue
      res = dsqrt(res)
      if(its.eq.1) res0 = res
c----------------------
c Convergence creteria
c----------------------
      err = res/res0
      print*, its-1,err
      if(iout.gt.0) write(iout,'(i4,2x,d20.6)') its-1,err
      if(err.le.tol.or.its.eq.maxits) return
      do 80 k=1,np
       wsflux(k) = sflux(k)
 80    continue
c
 85   continue
c
      return
      end
c=======================================================
      subroutine sweeping(nd,nx,ny,np,x,y,hx,hy,dx,dy,f,
     &                   bcl,bcr,bcb,bct,u)
c
c This subroutine invert the matrix H in (15) using the sweeping procedure
c
      implicit       real*8(a-h,o-z)
      dimension      hx(1),hy(1),dx(1),dy(1),x(1),y(1)
      dimension      bcl(1),bcr(1),bcb(1),bct(1)
      dimension      f(nx,ny),u(nd,nx+1,ny+1)
c
c Initialize the angular flux u
c
      do 5 l=1,nd
```

```
       do 5 i=1,nx+1
        do 5 j=1,ny+1
         u(l,i,j) = 0d0
 5    continue
c
c Strat the sweeping
c
      do 70 l=1,nd
       if(dx(l).lt.0d0.and.dy(l).lt.0d0) then
c-------------------
c The first sweeping
c-------------------
        do 10 j=1,ny+1
         u(l,nx+1,j) = bcr(j)
 10      continue
        do 15 i=1,nx+1
         u(l,i,ny+1) = bct(i)
 15     continue
        do 20 i=nx,1,-1
         do 20 j=ny,1,-1
          d  = dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e1 = dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e2 = -dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e3 = -dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
c
          u(l,i,j) = (f(i,j)-e2*u(l,i+1,j)-e3*u(l,i+1,j+1)-
     &                 e1*u(l,i,j+1))/d
 20     continue
       elseif(dx(l).lt.0d0.and.dy(l).gt.0d0) then
c--------------------
c The second sweeping
c--------------------
        do 25 j=1,ny+1
         u(l,nx+1,j) = bcr(j)
 25     continue
        do 30 i=1,nx+1
         u(l,i,1) = bcb(i)
 30     continue
        do 35 i=nx,1,-1
         do 35 j=1,ny
          d  = dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e1 = dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
```

```
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e2 = -dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e3 = -dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
c
          u(l,i,j+1) = (f(i,j)-e3*u(l,i+1,j)-e1*u(l,i,j)-
     &                   e2*u(l,i+1,j+1))/d
 35     continue
       elseif(dx(l).gt.0d0.and.dy(l).lt.0d0) then
c--------------------
c The third sweeping
c--------------------
        do 40 j=1,ny+1
         u(l,1,j) = bcl(j)
 40     continue
        do 45 i=1,nx+1
         u(l,i,ny+1) = bct(i)
 45     continue
        do 50 i=1,nx
         do 50 j=ny,1,-1
          d  = dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e1 = dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e2 = -dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e3 = -dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
c
          u(l,i+1,j) = (f(i,j)-e2*u(l,i,j)-e1*u(l,i+1,j+1)-
     &                   e3*u(l,i,j+1))/d
 50     continue
       else
c---------------------
c The fourth sweeping
c---------------------
        do 55 j=1,ny+1
         u(l,1,j) = bcl(j)
 55     continue
        do 60 i=1,nx+1
         u(l,i,1) = bcb(i)
 60     continue
        do 65 i=1,nx
         do 65 j=1,ny
          d  = dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &          0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
```

```
          e1 = dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e2 = -dabs(dx(l))/(2d0*hx(i))+dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
          e3 = -dabs(dx(l))/(2d0*hx(i))-dabs(dy(l))/(2d0*hy(j))+
     &         0.25d0*(sigma(x(i),y(j))+xkappa(x(i),y(j)))
c
          u(l,i+1,j+1) = (f(i,j)-e1*u(l,i+1,j)-e3*u(l,i,j)-
     &                    e2*u(l,i,j+1))/d
 65     continue
       endif
 70   continue
c
      return
      end
c============================
      function xint(nd,wg,u)
c
c This function compute the weighted
c integral operator applied to u.
c
      implicit  real*8(a-h,o-z)
      dimension u(1),wg(1)
c
      xint = 0d0
      do 5 l=1,nd
       xint = xint+wg(l)*u(l)
 5    continue
c
      return
      end
c==========================
      function xkappa(x,y)
c
c This function set the absorption coefficient.
c As default is set to 0 here, but can be changed
c by the user.
c
      implicit real*8(a-h,o-z)
c
      xkappa = 0d0
c
      return
      end
c=========================
      function sigma(x,y)
c
```

```
c This function set the scattering coefficient.
c As default is set to 1 here, but can be changed
c by the user.
c
      implicit real*8(a-h,o-z)
c
      sigma = 1d0
c
      return
      end
c=======================
      function bcleft(y)
c
c This function set the inflow boundary
c on the left boundary. As default is set
c to 0 here, but can be changed by the user.
c
      implicit real*8(a-h,o-z)
c
      bcleftl = 0d0
c
      return
      end
c========================
      function bcright(y)
c
c This function set the inflow boundary
c on the right boundary. As default is set
c to 0 here, but can be changed by the user.
c
      implicit real*8(a-h,o-z)
c
      bcright = 0d0
c
      return
      end
c=========================
      function bcbottom(x)
c
c This function set the inflow boundary
c on the bottom boundary. As default is set
c to 0 here, but can be changed by the user.
c
      implicit real*8(a-h,o-z)
c
      bcbottom = 0d0
c
```

```
      return
      end
c=====================
      function bctop(x)
c
c This function set the inflow boundary
c on the top boundary. As default is set
c to 0 here, but can be changed by the user.
c
      implicit real*8(a-h,o-z)
c
      bctop = 0d0
c
      return
      end
```