Boltzmann-Fokker-Planck Equation Solver for Charged Particle Transport Using the Continuous Slowing Down Approximation

Melek Derman [1,*]

[1]Oregon State University, Corvallis, Oregon

ABSTRACT

In this report, I introduce a Python-based software package that I developed to solve the Boltzmann transport equation using the Continuous Slowing Down Approximation (CSDA). To solve this problem, I employ a linear discontinuous Galerkin finite element approach with coupled discretizations in both space and energy domains, which enables an accurate representation of continuous energy loss phenomena. My ultimate goal is to extend the model by incorporating the scattering process through the addition of the Fokker-Planck scattering term. Given the limited existing research on coupled space–energy discretizations, I aim to contribute to the literature in charged particle transport theory through this study.

Keywords: Charged Particle Transport, Discontinuous Galerkin Finite Element Method, Discrete Ordinates, Boltzmann Fokker-Plank Equation

1. INTRODUCTION

The Boltzmann-Fokker-Planck (BFP) equation, an adapted form of the linear Boltzmann transport equation, effectively describes charged particle transport scenarios characterized by highly forward-peaked scattering and continuous energy loss. The BFP formulation simplified differential operators instead of complex integral collision kernels, significantly streamlining numerical solutions without compromising physical accuracy [1].

The other critical aspect of this work is the Continuous Slowing Down Approximation (CSDA), which assumes that charged particles experience continuous and deterministic energy loss in the medium, based on the macroscopic stopping power of the medium. Under this approximation, stochastic fluctuations in energy loss (energy straggling) are neglected, considering only the mean energy loss per unit distance, which greatly simplifies the energy transport modeling [2]. CSDA accurately represents the reduction of particle energy in media characterized by numerous small-angle collisions, providing a robust estimate of the depth of particle penetration and the energy deposited in practical applications, such as radiation therapy dose calculations [3].

Furthermore, angular scattering is efficiently handled using the Fokker–Planck angular diffusion operator. This operator captures the cumulative effect of numerous small-angle deflections by retaining only the first two angular moments of the collision term, converting what would otherwise be a computationally challenging integral operator into a simpler partial differential form. This approach is advantageous for modeling multiple Coulomb scattering in charged particle transport, since it significantly reduces numerical challenges such as oscillations from anisotropic scattering [4].

*dermanm@oregonstate.edu

Despite these advantages, it is crucial to acknowledge the limitations in these approximations. The angular diffusion approximation is strictly valid only for very small-angle, forward-peaked scattering and might not accurately describe scenarios involving significant broad-angle scattering [4]. Nevertheless, the BFP equation with CSDA represents an optimal trade-off between computational efficiency and physical accuracy, widely adopted for deterministic modeling of charged-particle transport in dense media.

In this project I developed the mathematical formulation of the BFP equation with CSDA, describes numerical methods for solving this equation, including Discontinuous Galerkin and discrete ordinates (S_N) methods, and details implementation practices employing the MFEM finite element library.

2. METHODS

In this section, I detail the mathematical formulation and numerical strategies used to solve the Boltzmann–Fokker–Planck (BFP) equation under the Continuous Slowing Down Approximation (CSDA). The formulation is presented step by step, including the derivation of the collision operators, the weak formulation, and the discrete ordinates (S_N) approach for angular discretizations.

2.1. Governing Equations and CSDA

The starting point is the general Boltzmann transport equation for the particle distribution function f:

$$\frac{\partial f}{\partial t} + v\,\nabla_{\mathbf{r}} f + \frac{q}{m}\Big(E + \mathbf{V}\times\mathbf{B}\Big)\nabla_{\mathbf{v}} f = \left(\frac{\partial f}{\partial t}\right)_{\text{coll}} + Q, \tag{1}$$

where v is the particle speed, q/m the charge-to-mass ratio, E and $\mathbf{B}$ the electric and magnetic fields, respectively, and Q an external source term.

For practical transport simulations, it is common to work with the angular flux $\psi(\mathbf{r}, E, \Omega, t)$, which satisfies:

$$\frac{\partial \psi(\mathbf{r}, E, \Omega, t)}{\partial t} + \Omega\cdot\nabla_{\mathbf{r}}\psi(\mathbf{r}, E, \Omega, t) + \sigma_t(\mathbf{r}, E)\,\psi(\mathbf{r}, E, \Omega, t) = Q(\mathbf{r}, E, \Omega, t) + \left(\frac{\partial \psi}{\partial t}\right)_{\text{coll}}. \tag{2}$$

Here, σ_t denotes the total interaction cross section and Ω is the unit vector representing the direction of particle motion.

Under the CSDA, the energy loss is assumed to be continuous and deterministic, characterized by the stopping power $S(E)$, which represents the mean energy loss per unit distance. This approximation neglects stochastic fluctuations in energy loss (energy straggling) and simplifies the treatment of charged particle transport.

2.2. Collision Operator Decomposition

The collision term in Eq. (2) is decomposed into energy and angular components:

- **Energy-Dependent Collision Term:** The continuous energy loss is approximated by

$$\left(\frac{\partial \psi}{\partial t}\right)_{\text{coll, energy}} \approx \frac{\partial}{\partial E}\Big[S(E)\,\psi(\mathbf{r}, E, \Omega, t)\Big], \tag{3}$$

  where $S(E)$ is the stopping power.

- **Angular Collision Term:** The cumulative effect of multiple small-angle scatterings is modeled via an angular diffusion operator:

$$\left(\frac{\partial \psi}{\partial t}\right)_{\text{coll, angular}} \approx D\, \nabla_{\Omega}^2 \psi(\mathbf{r}, E, \Omega, t), \tag{4}$$

with D being the angular diffusion coefficient. The angular Laplacian ∇_{Ω}^2 can be expressed in two common forms. In terms of the cosine of the polar angle μ:

$$D\, \nabla_{\Omega}^2 \psi \approx \frac{\alpha}{2} \frac{\partial}{\partial \mu}\left[(1-\mu^2)\frac{\partial \psi}{\partial \mu}\right], \tag{5}$$

or, alternatively, in a coordinate system with ξ and the azimuthal angle ϕ:

$$D\, \nabla_{\Omega}^2 \psi \approx -\frac{\alpha}{2}\left[\frac{\partial}{\partial \xi}\left((1-\xi^2)\frac{\partial \psi}{\partial \xi}\right) + \frac{1}{1-\xi^2}\frac{\partial^2 \psi}{\partial \phi^2}\right]. \tag{6}$$

Substituting the collision approximations from Eqs. (3) and (4) into Eq. (2) leads to the combined transport equation:

$$\frac{\partial \psi}{\partial t} + \Omega \cdot \nabla_{\mathbf{r}} \psi + \sigma_t\, \psi = Q + C\psi + D\, \nabla_{\Omega}^2 \psi, \tag{7}$$

where the energy collision operator C is defined as:

$$C\psi = \frac{\partial}{\partial E}\Big[S(E)\, \psi\Big]. \tag{8}$$

2.3. Discrete Ordinates (S_N) Method

The angular variable Ω is discretized using the discrete ordinates method (S_N). In this approach, the continuous angular domain is approximated by a finite set of directions $\{\Omega_i\}_{i=1}^{N_\Omega}$. The transport equation is then solved independently along each discrete direction:

$$\frac{\partial \psi_i}{\partial t} + \Omega_i \cdot \nabla_{\mathbf{r}} \psi_i + \sigma_t\, \psi_i = Q_i + C\psi_i + D\, \nabla_{\Omega}^2 \psi_i, \tag{9}$$

where ψ_i is the angular flux corresponding to the i^{th} direction. The scattering source, which couples different directions, is integrated over the discrete set, hence reducing the continuous angular integration to a weighted summation.

2.4. Weak Formulation

To solve Eq. (7) numerically, I adopt a weak formulation. Multiplying the equation by a test function $W(\mathbf{r}, E, \Omega)$ and integrating over the computational domain D yields:

$$\int_D \frac{\partial \psi}{\partial t}\, W\, dD + \int_D \Omega \cdot \nabla_{\mathbf{r}} \psi\, W\, dD + \int_D \sigma_t\, \psi\, W\, dD = \int_D Q\, W\, dD + \int_D C\psi\, W\, dD + \int_D D\, \nabla_{\Omega}^2 \psi\, W\, dD. \tag{10}$$

This weak formulation is essential for the stability and accuracy of the Discontinuous Galerkin (DG) method. In this formulation, boundary conditions and interior face fluxes are handled via appropriate numerical flux functions and trace integrators.

Finite Element Method and Discrete System

For the DG discretization, the weak form is further transformed by introducing a finite element expansion of the angular flux. In this work, I proceeded by assuming that the system is in steady state and that it does not include a scattering term (which will be added later). Under these assumptions, the following equations were obtained.

The integrated weak form after applying integration by parts and incorporating boundary terms is given by

$$\begin{aligned}
&\mu \int dE\, W_{ik}(x_R, E)\, \tilde{\psi}(x_R, E) - \mu \int dE\, W_{ik}(x_L, E)\, \tilde{\psi}(x_L, E) \\
&\quad - \mu \int dE \int dx\, \tilde{\psi}(x, E) \frac{\partial W_{ik}(x, E)}{\partial x} + \int dE \int dx\, W_{ik}(x, E)\, \sigma_t(E) \tilde{\psi}(x, E) \\
&\quad - \int dx\, W_{ik}(x, E_{\min})\, S(E_{\min}) \tilde{\psi}(x, E_{\min}) + \int dx\, W_{ik}(x, E_{\max})\, S(E_{\max}) \tilde{\psi}(x, E_{\max}) \\
&\quad + \int dx \int dE\, S(E) \tilde{\psi}(x, E) \frac{\partial W_{ik}(x, E)}{\partial E} = \int dE \int dx\, W_{ik}(x, E) Q.
\end{aligned} \tag{11}$$

The angular flux $\tilde{\psi}(x, E)$ was expanded in terms of finite element basis functions $B_j(x, E)$ as follows:

$$\tilde{\psi}(x, E) = \sum_{j=1}^{J_i} \psi_j\, B_j(x, E), \tag{12}$$

where ψ_j are the unknown coefficients and J_i is the number of basis functions in the i^{th} element.

Substituting the expansion from Eq. (12) into the weak form (Eq. (11)) leads to a system of algebraic equations:

$$
\begin{aligned}
&\mu \sum_{j=1}^{J_i} \int dE\, W_{ik}(x_R, E)\psi(x_R, E)B_j(x_R, E) - \mu \sum_{j=1}^{J_i} \int dE\, W_{ik}(x_L, E)\psi(x_L, E)B_j(x_L, E) \\
&\quad - \mu \sum_{j=1}^{J_i} \int dE \int dx\, \psi(x, E)B_j(x, E)\frac{\partial W_{ik}(x, E)}{\partial x} \\
&\quad + \sum_{j=1}^{J_i} \int dE \int dx\, W_{ik}(x, E)\, \sigma_t(E)\psi(x, E)B_j(x, E) \\
&\quad - \sum_{j=1}^{J_i} \int dx\, W_{ik}(x, E_{\min})S(E_{\min})\psi(x, E_{\min})B_j(x, E_{\min}) \\
&\quad + \sum_{j=1}^{J_i} \int dx\, W_{ik}(x, E_{\max})S(E_{\max})\psi(x, E_{\max})B_j(x, E_{\max}) \\
&\quad + \sum_{j=1}^{J_i} \int dx \int dE\, S(E)\psi(x, E)B_j(x, E)\frac{\partial W_{ik}(x, E)}{\partial E} \\
&\quad = \int dE \int dx\, W_{ik}(x, E)Q.
\end{aligned}
\tag{13}
$$

The inflow boundary conditions are imposed by specifying the incident angular flux:

$$
\psi_{0,E} = \psi_{\text{inc}}(0, E), \quad \Omega > 0, \qquad \psi_{L,E} = \psi_{\text{inc}}(X_L, E), \quad \Omega < 0. \tag{14}
$$

Equations (11)-(14) form the discrete weak formulation of the BFP equation under CSDA. Equation (11) represents the balance of flux contributions over each finite element, incorporating both the convective transport and the continuous energy loss effects. Equation (12) discretizes the angular flux into a series expansion over basis functions, while Equation (13) results in a system of algebraic equations for the flux coefficients, ensuring the conservation laws are satisfied in an averaged sense. Finally, Equation (14) enforces the physically meaningful inflow boundary conditions, ensuring that the incoming particle flux is correctly prescribed. Together, these equations capture the essential physics of charged-particle transport and enable a numerical solution using the DG method.

3. IMPLEMENTATION

In this work, my main goal is to develop a software package to solve the BFP equation using the CSDA. Currently, the scattering term is not integrated, and the software does not yet support parallel computation. These features will be incorporated in future versions. The recent version (v0.1.0) of BFP solves the steady-state Boltzmann transport equation using the CSDA.

3.1. Software Structure and Dependencies

The BFP solver used several software libraries. Specifically, I used the Modular Finite Element Method (MFEM) [5] library and its Python wrapper, PyMFEM, developed by Lawrence Livermore National Laboratory (LLNL). These libraries enabled efficient finite element calculations for particle transport simulations. For visualization, I integrated PyGLVis [6] to visualize meshes and solution fields.

Additionally, it utilized several Python libraries such as;

- NumPy [7] for numerical operations.
- Matplotlib [8] and Seaborn[9] for data visualization.
- Pandas [10] for data analysis.

The software installation process was simplified with a Python script `setup.py`. This script installed necessary Python dependencies, cloned,, and installed external repositories (PyMFEM, PyGLVis) and cleaned temporary files after completion.

The BFP contained five main directories:

- `.github/`: Manages GitHub workflows and automates tests via Git Actions.
- `bfp/`: Contains source code modules and directories for input and output files.
- `docs/`: Includes Sphinx documentation files and project reports.
- `tests/`: Provides unit tests which are used by Git Actions.
- `examples/`: Contains Jupyter examples that demonstrate the use of the software.

3.2. Modules and Functionality

The software consisted of two folders and six major modules in the source (`bfp/`) folder:

- `data/`: Contains `.h5` files.
- `mesh/`: Contains several example `.mesh` files.
- `assemble.py`: Assembles the equation from bilinear and linear forms.
- `coeff.py`: Converts coefficients for equation systems into forms compatible with PyMFEM's special `Eval` methods, allowing interfacing with PyMFEM's C++ backend.
- `input.py`: Defines problem setups and parameters.
- `mesh.py`: Transforms MFEM-defined unit meshes into the real coordinates for the actual problem setup.
- `solver.py`: Contains solver routines to solve the constructed equation systems.
- `utils.py`: Provides helper functions, such as reading HDF5 files and generating Gauss-Legendre quadratures.
- `vis.py`: Offers visualization tools for simulation results.

The BFP could be executed easily via command-line interfaces and configuration input files, allowing users flexibility to adjust parameters without modifying the source code.

3.3. Numerical Implementation via PyMFEM

I employed several MFEM integrators to discretize terms in the weak form of the BFP equation:

- `ConvectionIntegrator`: Calculated the convective term $\int_D \Omega \cdot \nabla_{\mathbf{r}} \psi W dD$.
- `MassIntegrator`: Implemented the mass term $\int_D \sigma_t \psi W dD$.
- `DomainLFIntegrator`: Managed source and diffusion terms for energy and angular collisions.
- `DGTraceIntegrator and BoundaryFlowIntegrator`: Computed numerical fluxes across element interfaces and enforced weak boundary conditions.

4. RESULTS

BFP was verified using one benchmark problem and four Model of Manufactured Solutions (MMS). These problems are summarized below:

Problem 1: Homogeneous Infinite Medium with CSDA

This problem solves the equation for a homogeneous infinite medium:

$$\mu \frac{\partial \psi(x, E, \mu)}{\partial x} + \frac{\partial}{\partial E} \left(S(E)\, \psi(x, E, \mu) \right) + \sigma_t\, \psi(x, E, \mu) = Q \tag{15}$$

where ψ is the angular flux, $S(E)$ is the stopping power, σ_t is the total cross-section, Q is an isotropic source, and μ is the directional cosine. Under these assumptions, the analytical solution is given in Eq. (16).

The GLVis representations of the solution obtained with $Q = 100$ and $\sigma_t = 5$ can be seen in Figure 1 in the Appendix A.

$$\psi = \frac{Q}{\sigma_t}, \quad \phi = \frac{2Q}{\sigma_t}. \tag{16}$$

Problem 2: Exponential Decrease with Inflow Boundary

This problem tests exponential attenuation with an inflow boundary condition. The analytical solution is:

$$\psi(x) = \frac{Q}{\sigma_t} + \psi_\ell\, e^{-\frac{\sigma_t}{\mu} x} \tag{17}$$

The GLVis representations of the solution obtained with $Q = 10$, $\sigma_t = 5$ and boundary inflow $\psi_\ell = 10$ can be seen in Figure 2 in the Appendix A.

Problem 3: Linear Spatial Dependence

For Problem 3, which considers linear spatial dependence, we used the angular flux value provided by the analytical solution.

$$\psi(x) = a + bx, \tag{18}$$

where constants a and b define the intercept and slope, respectively. With constant σ_t and inflow values, and for $a = 5$ and $b = 10$, the results are shown in Appendix A, Figure 3.

Problem 4: Linear Energy Dependence

Similarly to Problem 3, the analytical solution of the equation solved with linear energy dependence is given in Eq. (19).

$$\psi(E) = a + bE, \tag{19}$$

where a and b are constants, with $a = 5$ and $b = 10$; the solution for these values is shown in Appendix A, Figure 4.

Problem 5: Linear Spatial and Energy Dependence

Finally, the analytical solution of the simple MMS problem that verifies the linear spatial-energy dependence is given as follows:

$$\psi(x, E) = a + bxE. \tag{20}$$

For the constant $a = 5$ and $b = 10$; the solution is shown in Appendix A, Figure 4.

5. CONCLUSIONS AND FUTURE WORK

The main goal of this project was to solve the Boltzmann–Fokker–Planck (BFP) transport problem, which is commonly used for charged particle problems with forward-peaked scattering, under a 2D spatial–energy coupling approach using the finite element method. In version v0.1.0, the latest release of the BFP solver, I validated that the BFP can solve the Boltzmann equation under the CSDA appropriately, excluding the scattering term, which is one of the final goals of the project.

During this project, I experienced some difficulties related to visualization tools. I realized that GLVis, a commonly used visualization tool for MFEM, does not provide the features required for my needs. Moreover, I still lack a complete understanding of how to correctly assemble outputs from DG finite element discretizations, especially when the output data corresponds to multiple nodes per element. As a result, the figures could not be presented at the desired standard. Additionally, although analytical solutions and rough error estimates were obtained as part of the project, these were not included in this report.

This study has successfully addressed one of the more challenging steps toward its final goal. As future work, I will continue to improve my software by implementing a few more complex MMS test cases. Following this, the Fokker–Planck scattering term will be added and tested across different solvers to evaluate and compare performance. Once the software tool is fully established, parallel computations and performance analysis will be conducted using the internal solvers provided by MFEM.

ACKNOWLEDGMENTS

This work was supported by the Center for Exascale Monte-Carlo Neutron Transport (CEMeNT) a PSAAP-III project funded by the Department of Energy, grant number: DE-NA003967.

REFERENCES

[1] J. Morel. "Multigroup Legendre Coefficients for the Diamond Difference Continuous Slowing Down Operator." *Applied Theoretical Physics Division, Los Alamos National Laboratory* (1984).

[2] M. Berger. "Stopping-power and range tables for electrons, protons, and helium ions." *National Institute of Standards and Technology* (1992). NISTIR 4999.

[3] I. R. 49. "Stopping Powers and Ranges for Protons and Alpha Particles." *International Commission on Radiation Units and Measurements* (1993).

[4] G. Pomraning. "The Fokker-Planck Operator as an Asymptotic Limit of the Boltzmann Collision Operator." *Transport Theory and Statistical Physics*, **volume 21**, pp. 393–413 (1992).

[5] R. Anderson, J. Andrej, A. Barker, et al. "MFEM: A modular finite element methods library." *Computers & Mathematics with Applications* (2020).

[6] T. Kolev, V. Dobrev, and USDOE. "GLVis: OpenGL Finite Element Visualization Tool." (2010).

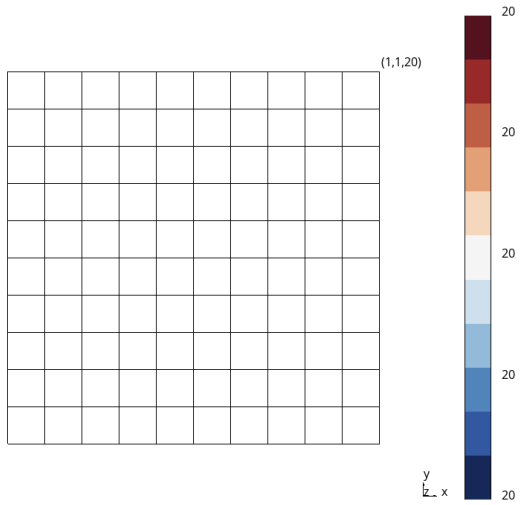
[7] C. R. Harris, K. J. Millman, S. J. van der Walt, et al. "Array programming with NumPy." *Nature*, **volume 585**, pp. 357–362 (2020).

[8] J. D. Hunter. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, **volume 9**(3), pp. 90–95 (2007).

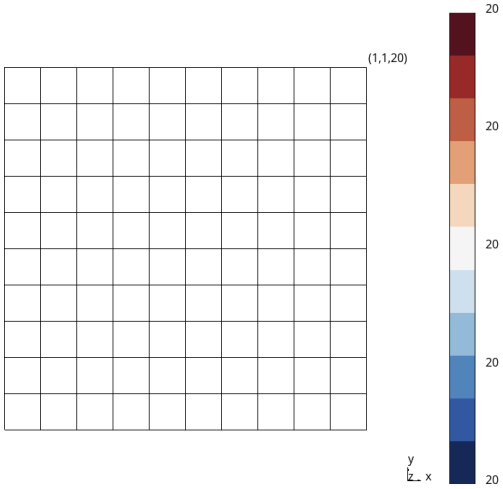
[9] M. L. Waskom. "seaborn: statistical data visualization." *Journal of Open-Source Software*, **volume 6**(60), p. 3021 (2021).

[10] T. pandas development team. "pandas-dev/pandas: Pandas." (2020). URL https://doi.org/10.5281/zenodo.3509134.

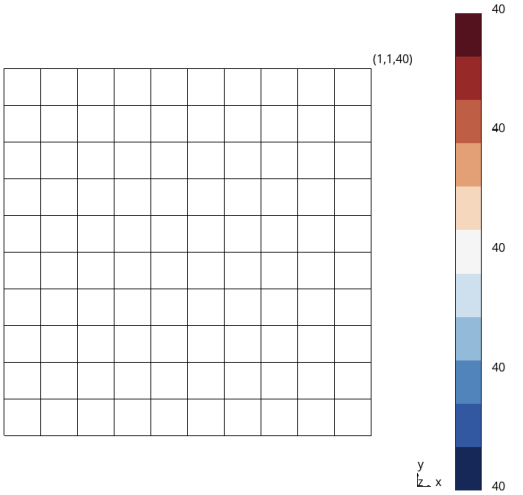
APPENDIX A.



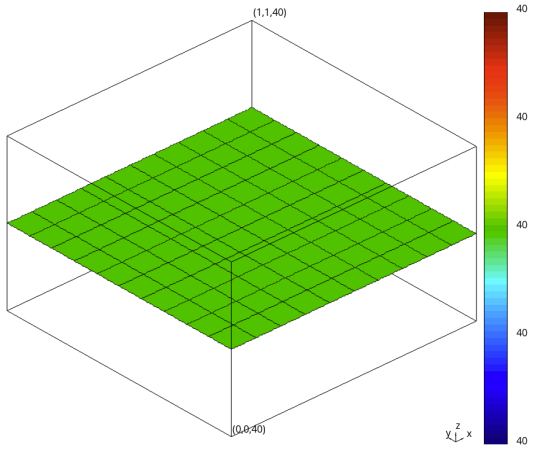
((a)) Angular flux for μ = -0.577.



((b)) Angular flux for μ = 0.577.

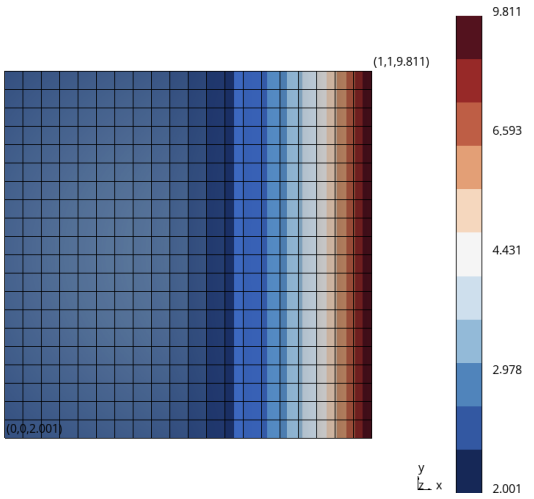


((c)) 2D representation of the scalar flux

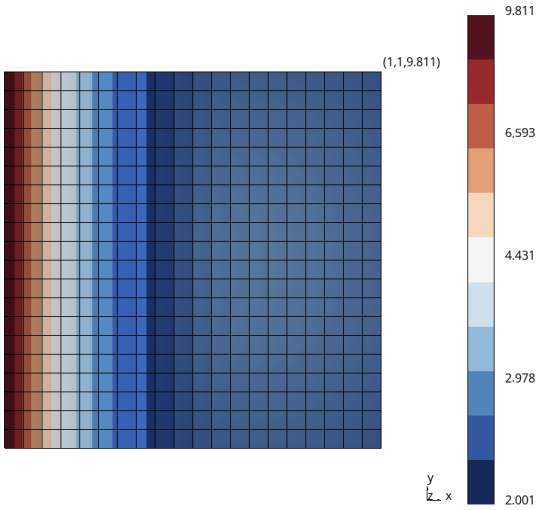


((d)) 3D representation of the scalar flux

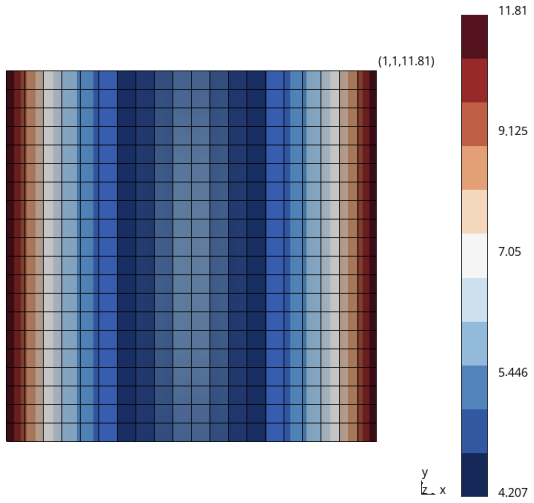
Figure 1. In this figure, (a) and (b) compare the angular flux at two directions for 2 directional S_N calculations of Problem 1, while (c) and (d) illustrate the scalar flux in both 2D and 3D. These visualizations demonstrate how the angular and scalar flux distribution changes based on spatial and energy directions and provide a understanding of the overall solution behavior.



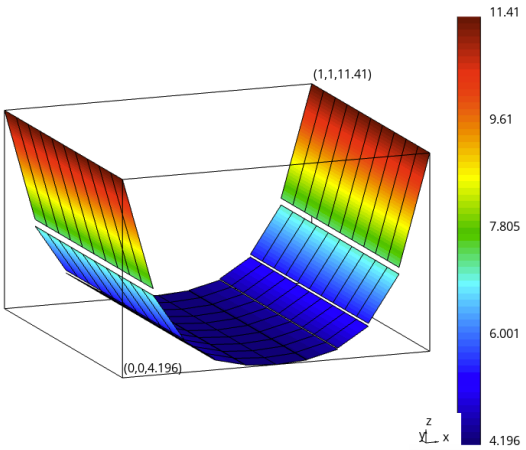
((a)) Angular flux for μ = -0.577.



((b)) Angular flux for μ = 0.577.

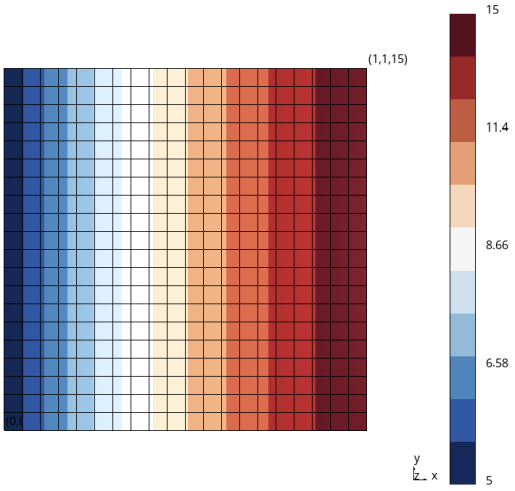


((c)) 2D representation of the scalar flux.

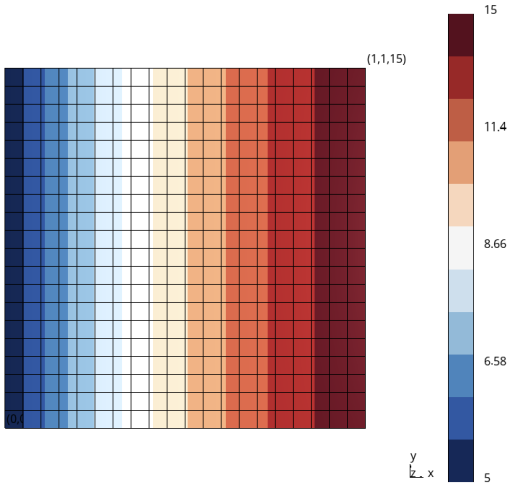


((d)) 3D representation of the scalar flux

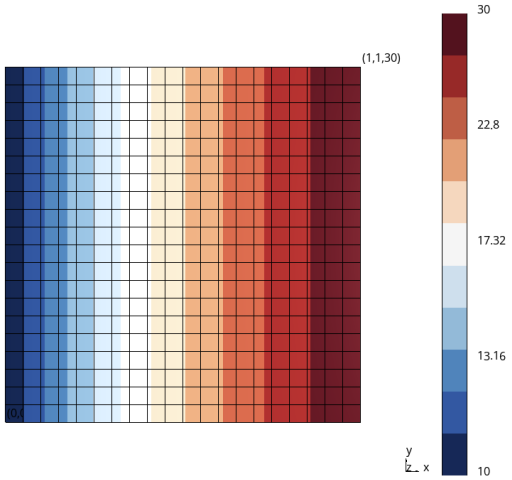
Figure 2. In this figure, (a) and (b) compare the angular flux at two directions for 2 directional S_N calculations of Problem 2, while (c) and (d) illustrate the scalar flux in both 2D and 3D. These visualizations demonstrate how the angular and scalar flux distribution changes based on spatial and energy directions and provide a understanding of the overall solution behavior.



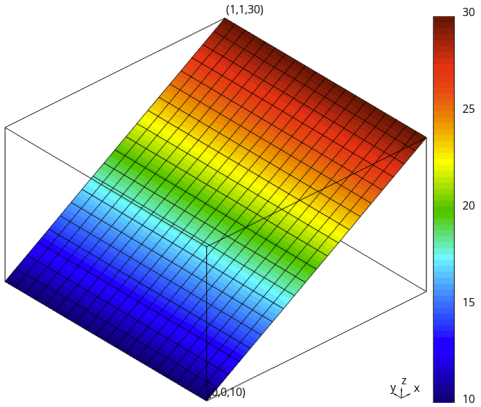
((a)) Angular flux for μ = -0.577.



((b)) Angular flux for μ = 0.577.

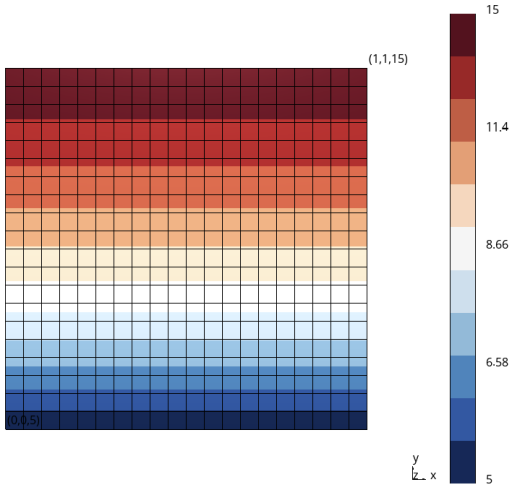


((c)) 2D representation of the scalar flux

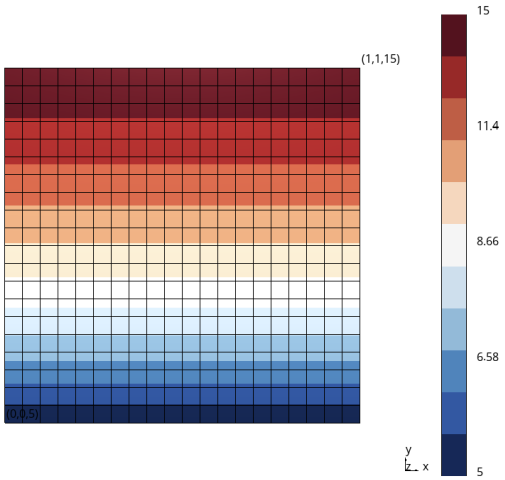


((d)) 3D representation of the scalar flux

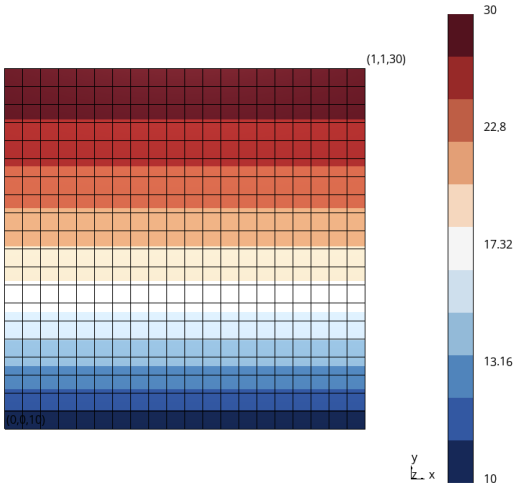
Figure 3. In this figure, (a) and (b) compare the angular flux at two directions for 2 directional S_N calculations of Problem 3, while (c) and (d) illustrate the scalar flux in both 2D and 3D. These visualizations demonstrate how the angular and scalar flux distribution changes based on spatial and energy directions and provide a understanding of the overall solution behavior.



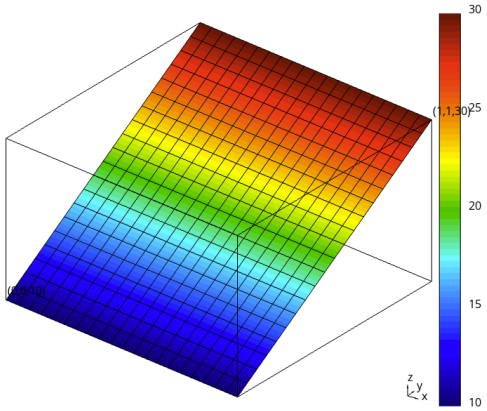
((a)) Angular flux for μ = -0.577.



((b)) Angular flux for μ = 0.577.

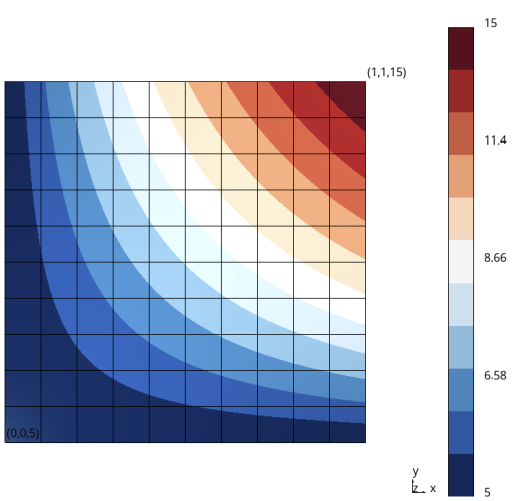


((c)) 2D representation of the scalar flux

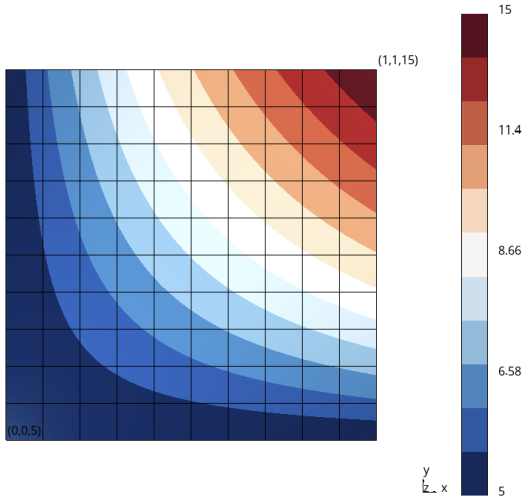


((d)) 3D representation of the scalar flux

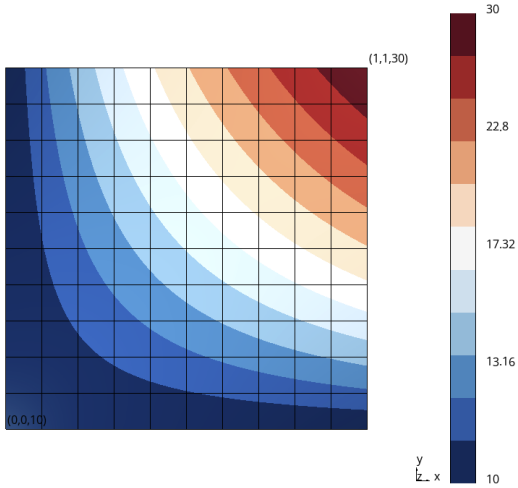
Figure 4. In this figure, (a) and (b) compare the angular flux at two directions for 2 directional S_N calculations of Problem 4, while (c) and (d) illustrate the scalar flux in both 2D and 3D. These visualizations demonstrate how the angular and scalar flux distribution changes based on spatial and energy directions and provide a understanding of the overall solution behavior.



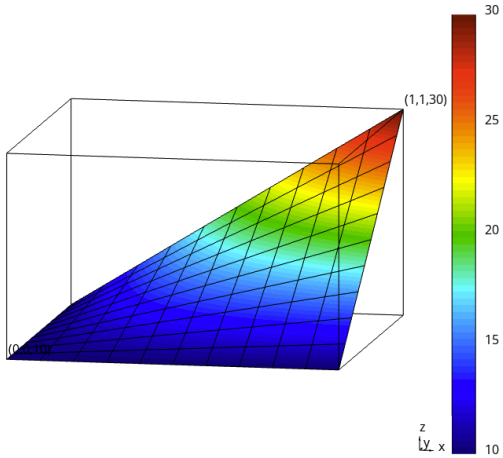
((a)) Angular flux for μ = -0.577



((b)) Angular flux for μ = 0.577



((c)) 2D representation of the scalar flux



((d)) 3D representation of the scalar flux

Figure 5. In this figure, (a) and (b) compare the angular flux at two directions for 2 directional S_N calculations of Problem 5, while (c) and (d) illustrate the scalar flux in both 2D and 3D. These visualizations demonstrate how the angular and scalar flux distribution changes based on spatial and energy directions and provide a understanding of the overall solution behavior.