FeSNT - A Finite Element Neutron Transport Solver

Andrew Johnson

May 2, 2018

1 Introduction

The goal of this work is to introduce and explain the code written for the MATH6641 project. The time-dependent one-dimensional neutron transport equation was solved using the discrete ordinates formulation [1] using discontinuous-Galerkin in space, and explicit Euler for time. The project was written in Python and works best with 3.5[1] and is included with this report in `fesnt.py`. This report, and the code included, are covered with the GNU Public License, found in `LICENSE`. The all additional information for this work can be found at `https://github.com/drewejohnson/fesnt`

1.1 Problem Statement

The discrete-ordinates formulation of the neutron transport equation chooses to solve for the angular neutron flux, $\psi(x, \mu, t)$ along a discrete set of N angles $\vec{\mu}$. A quadrature set of weights and angles is chosen to satisfy some physical constraints, briefly summarized here:

$$\mu_m = -\mu_{N-m+1} \tag{1a}$$

$$w_m = w_{N-m+1} \tag{1b}$$

$$\sum_m w_m = 2 \tag{1c}$$

$$\int_{-1}^{1} f(x, \mu) d\mu \approx \sum_m w_m f(x, \mu_m) \tag{1d}$$

The weights and angles are, for this work, chosen from a Guass-Legendre polynomial set. Select sets of weights and angles are listed in section 6.1

This project seeks to solve the following initial-boundary value problem:

$$\frac{1}{\nu}\frac{\partial \psi_m}{\partial t} + \mu_m \frac{\partial \psi_m}{\partial x} + \sigma_t(x)\psi(x,t) = \sigma_S(x)\phi(x,t), \text{ for } a < x < b, t > 0, \mu_m \in \vec{\mu} \tag{2}$$

where

$$\phi(x,t) = \sum_m w_m \psi_m(x,t) \tag{3}$$

and $\sigma_S(x) = \frac{1}{2}\left(\sigma_s(x) + \chi\nu\sigma_f(x)\right)$.

Equation (2) is subject to initial and boundary conditions

$$\psi_m(x, 0) = \psi_{m,0}(x), \text{ for } a < x < b, \mu_m \in \vec{\mu} \tag{4a}$$

$$\psi_m(a, t) = \Gamma_a(\mu_m, t) \text{ for } \mu_m > 0, t > 0 \tag{4b}$$

$$\psi_m(b, t) = \Gamma_b(\mu_m, t) \text{ for } \mu_m < 0, t > 0 \tag{4c}$$

The report is laid out as follows. Section 2 derives the numerical scheme implemented for this project. Section 3 gives some insight into the actual implementation and usage needed to reproduce the results. Section 4 presents results from three test cases of increasing difficulty.

[1] not tested on other versions

2 Derivation of the Method

The author set out to apply a finite-element scheme for the spatial domain, with an implicit Euler scheme in time. This section shall detail the mathematical foundation for the project, and present the numeric scheme that was ultimately implemented for this work.

2.1 Notation

Following the work of Reed and Hill [2], the author set out to apply a Discontinuous-Galerkin (DG) method for the spatial domain. This required subdividing the domain $a < x < b$ into non-overlapping elements τ. The finite-element space $\mathcal{V}_h$ was chosen to be piece-wise quadratic polynomials that are continuous within a single element.

2.1.1 Finite-Element Spaces

The following finite-element space was used for the trial functions that approximated $\psi(x, \mu, t)$ within an element τ. Let $\mathcal{V}_h$ be the space

$$\begin{aligned} \mathcal{V}_h = \{\tilde{\psi}_m | \tilde{\psi}_m &\text{ is piece-wise quadratic,} \\ \tilde{\psi}_m(a, \mu, t) &= \begin{cases} \Gamma_a(\mu, t), & \text{if } \mu > 0 \\ 0, & \text{otherwise} \end{cases} \quad , \\ \tilde{\psi}_m(b, \mu, t) &= 0\} \end{aligned} \tag{5}$$

For this work, the test functions used to create the finite-element method were of the piece-wise quadratic within each mesh and taken to be

$$\eta_i(x) = x^i \tag{6}$$

which, for $i \leq 2$, is contained within $\mathcal{V}_h$.

2.1.2 Quadratic Flux

Within each element τ, the flux was represented as a quadratic function using the element boundaries and midpoint of the element. Specifically, the angular flux was represented using Lagrange interpolating polynomials, as

$$\tilde{\psi}_m(x) = \sum_{j=0}^{2} \tilde{\psi}_{m,j} \sum_{l=0}^{2} \alpha_{j,l} x^l = \sum_{j=0}^{2} \tilde{\psi}_{m,j} \prod_{l=0, l\neq j}^{2} \frac{x - x_l}{x_j - x_l} \tag{7}$$

This way, given three internal points of element τ, the Lagrange coefficients $\alpha_{j,l}$ could be determined, meaning that amplitudes $\tilde{\psi}_{m,j}$ remained to be solved. Using this notation, the spatial derivative of the flux is expressed as

$$\frac{\partial \tilde{\psi}_m}{\partial x} = \sum_{j=0}^{2} \tilde{\psi}_{m,j} \sum_{l=1}^{2} l \alpha_{j,l} x^{l-1} \tag{8}$$

The benefit of using Lagrange polynomials is that, at a specific lattice site x_l, the angular flux evaluates to the specific polynomial amplitude, i.e.

$$\tilde{\psi}_m(x_j) = \tilde{\psi}_{m,j} \tag{9}$$

Using a quadratic representation of the flux within an element allows the use of Simpson's integration without introducing additional error, as such a method is exact for polynomials of order 2 or less. According to Simpson's rule, the spatial integral of a function can be expressed as

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \left(f(a) + 4f\left(\frac{b-a}{2}\right) + f(b) \right) \tag{10}$$

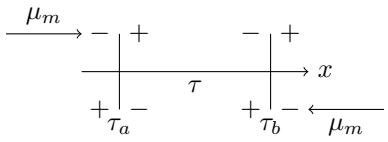

Figure 1: Notation used to denote upwind and downwind elements given flux angle μ

2.1.3 Upwind/Downwind Elements

The scheme implemented iterates through all angles in the quadrature set, and marches through elements in a manner akin to following the wind-direction. When mu is less than zero, this translates to moving left along the x axis, and the first mesh is taken to be the mesh with an edge at $x = b$. Such a scheme requires distinctions to be made between upwind and downwind sides, as will be utilized heavily in later sections. Figure 1 describes the upwind side of the edges of element τ as $-$ sides, and downwind sides as $+$.

Using this distinction, the jump of a function across an element edge is defined with

$$V^+ = \lim_{s \to 0^+} V(x + s\mu) \tag{11a}$$

$$V^- = \lim_{s \to 0^-} V(x + s\mu) \tag{11b}$$

$$[V(x)] = V^+ - V^- \tag{11c}$$

2.2 Finite-Element Method

Given these definitions, the process for deriving the finite-element method can begin. To start, multiply eq. (2) by a test function $\eta_i \in \mathcal{V}_h$ and integrating over the spatial domain $\Omega = [a, b]$. If the spatial discretization $\mathcal{T}_h$ does not cross material boundaries, then cross sections σ_S are constant within an element. The finite element will seek to approximate

$$\sum_{\tau \in \mathcal{T}_h} \left[\frac{1}{v} \left(\frac{\partial \psi_m}{\partial t}, \eta \right)_\tau + \mu_m \left(\frac{\partial \psi_m}{\partial x}, \eta \right)_\tau + \sigma_{t,\tau} \left(\psi_m, \eta \right)_\tau \right] = \sum_{\tau \in \mathcal{T}_h} \sigma_S \left(\phi, \eta \right)_\tau, \forall \eta \in \mathcal{V}_h \tag{12}$$

By integrating the spatial derivative term by parts, the derivative can be shifted off the true function ψ[2].

$$\left(\frac{\partial \psi_m}{\partial t}, \eta \right)_\tau = \left[\overset{*}{\psi}_m \, \eta|_\tau \right] \Big|_{\tau_a}^{\tau_b} - \left(\psi_m, \frac{\partial \, \eta|_\tau}{\partial x} \right)_\tau \tag{13}$$

Here, caution must be taken to ensure that the test function is evaluated *inside* the element τ. Since ψ_m is sufficiently smooth, we can take the upwind information $\overset{*}{\psi}_m = \tilde{\psi}_m^-$, but choosing $\eta|_\tau$ to always be inside the element. Another integration by parts, replacing $\psi_m \to \tilde{\psi}_m$ as the derivative has been removed

$$\left(\tilde{\psi}_m, \frac{\partial \, \eta|_\tau}{\partial x} \right)_\tau = \left(\overset{**}{\psi}_m \, \eta|_\tau \right) \Big|_{\tau_a}^{\tau_b} - \left(\left. \frac{\partial \tilde{\psi}_m}{\partial x} \right|_\tau, \eta \right)_\tau \tag{14}$$

Here, $\overset{**}{\psi}_m$ must be taken inside τ, yielding

$$\left(\frac{\partial \psi_m}{\partial x}, \eta \right)_\tau \Rightarrow \left(\left. \frac{\partial \tilde{\psi}_m}{\partial x} \right|_\tau, \eta \right)_\tau + \left(\left(\overset{*}{\psi}_m - \overset{**}{\psi}_m \right) \eta|_\tau \right) \Big|_{\tau_a}^{\tau_b} \tag{15}$$

For positive or negative values of μ, different terms in eq. (15) will cancel. For $\mu > 0$,

$$\overset{**}{\tilde{\psi}}_m = \begin{cases} \tilde{\psi}_m^-, \text{ on } \tau_b \\ \tilde{\psi}_m^+, \text{ on } \tau_a \end{cases} \tag{16}$$

[2] assumed to be sufficiently smooth

Similarly, for $\mu < 0$,

$$\overset{**}{\tilde{\psi}}_m = \begin{cases} \tilde{\psi}_m^+, \text{ on } \tau_b \\ \tilde{\psi}_m^-, \text{ on } \tau_a \end{cases} \tag{17}$$

Using these definitions, the generalized version of eq. (15) approximates to

$$\mu_m \left(\frac{\partial \psi_m}{\partial x}, \eta \right)_\tau \Rightarrow \mu_m \left(\frac{\partial \tilde{\psi}_m}{\partial x}, \eta \right)_\tau + |\mu_m| \left(\left[\tilde{\psi}_m \right] \eta|_\tau \right) \Big|_{\tau^{upw}} \tag{18}$$

where τ^{upw} is taken to be the upwind edge of element τ.

For this project, the time derivative was approximated with an implicit Euler scheme,

$$\left(\frac{\partial \psi_m}{\partial t}, \eta \right)_\tau \approx \frac{\tilde{\psi}_m^{n+1} - \tilde{\psi}_m^n}{\Delta t_n} \tag{19}$$

and the final finite-element method can be written as follows: For all angles $\mu_m \in \vec{\mu}$, find $\tilde{\psi}_m$ such that

$$\begin{aligned} \sum_{\tau \in \mathcal{T}_h} \left[\left(\sigma_t^\tau + \frac{1}{v^\tau \Delta t_n} \right) \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau + \mu_m \left(\frac{\partial \tilde{\psi}_m^{n+1}}{\partial x}, \eta \right)_\tau + |\mu_m| \left(\left[\tilde{\psi}_m^{n+1} \right] \eta|_\tau \right) \Big|_{\tau^{upw}} \right] \\ = \sum_{\tau \in \mathcal{T}_h} \left[\frac{1}{v^\tau \Delta t_n} \left(\tilde{\psi}_m^n, \eta \right)_\tau + \sigma_S \left(\tilde{\phi}^{n+1}, \eta \right)_\tau \right], \forall \eta \in \mathcal{V}_h \end{aligned} \tag{20}$$

where $\tilde{\phi} \equiv \sum_m w_m \tilde{\psi}_m$

3 Implementation

As discussed in section 4, the scheme presented in eq. (20) is applied to various one-dimensional problems. A traditional method for solving the discrete-ordinates form of the transport equation is to march through meshes along specific angles μ starting at an incident boundary condition [1]. For this approach, each mesh is solved individually using information taken from either the upwind mesh, in the form of jump terms, or the relevant boundary conditions.

3.1 Matrix Formulation

Equation (20) is recast into a linear system that will be solved iteratively, as the scalar flux $\tilde{\phi}$ must be evaluated at the same time level as the angular flux $\tilde{\psi}_m$.

$$\left[\mu_m \mathbf{A} + |\mu_m| \mathbf{B} + \mathbf{C} + \frac{1}{\Delta t_n} \mathbf{D} \right] \left\{ \tilde{\psi}_m^{n+1} \right\} = \frac{1}{\Delta t_n} \mathbf{D} \left\{ \tilde{\psi}_m^n \right\} + \left\{ S^{n+1} \right\} \tag{21}$$

The where matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ contain coefficients from the following bilinear operators:

$$a \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau = \left(\frac{\partial \tilde{\psi}_m^{n+1}}{\partial x}, \eta \right)_\tau \Rightarrow \{a_{ij}\} \tag{22a}$$

$$b \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau = \left(\left[\tilde{\psi}_m^{n+1} \right] \eta|_\tau \right) \Big|_{\tau^{upw}} \Rightarrow \{b_{ij}\} \tag{22b}$$

$$c \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau = \sigma_t \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau \Rightarrow \{c_{ij}\} \tag{22c}$$

$$d \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau = \frac{1}{v \Delta t_n} \left(\tilde{\psi}_m^{n+1}, \eta \right)_\tau \Rightarrow \{d_{ij}\} \tag{22d}$$

In each of the following examples, the variable ξ_j refers to the vector of coefficients from Simpson's integration: $\{\xi\} \equiv [0.5, 2.0, 0.5]$.

The integration $\left(\tilde{\psi}_m^{n+1}, \eta\right)_\tau$ is straightforward if the Kronecker-delta property of the Lagrange polynomials is utilized. By Simpson's integration:

$$\int_\tau \tilde{\psi}_m \eta dx \approx \frac{h_\tau}{2}\left(\tilde{\psi}_{m,0}\eta(x_0) + 4\tilde{\psi}_{m,1}\eta(x_1) + \tilde{\psi}_{m,1}\eta(x_2)\right) \tag{23}$$

The coefficients for the flux at the j-th node multiplied by η_i for the kernel of eqs. (22c) and (22d) can be expressed as

$$\left(\tilde{\psi}_m, \eta_i\right)_\tau = \{h_\tau \eta_i(x_j)\xi_j\} \tag{24}$$

Then the coefficients of $\mathbf{C}$ and $\mathbf{D}$ are

$$c_{i,j} = \sigma_t h_\tau \eta_i(x_j)\xi_j \tag{25a}$$

$$d_{i,j} = \frac{1}{v\Delta t_n} h_\tau \eta_i(x_j)\xi_j \tag{25b}$$

For the $\mathbf{A}$ coefficients, the integration requires a little more work

$$\begin{aligned}\int_\tau \frac{\partial\tilde{\psi}}{\partial x}\eta(x)dx \approx \frac{h_\tau}{2}\Bigg[&\eta(x_0)\sum_{j=0}^{2}\tilde{\psi}_{m,j}\sum_{l=1}^{2} l\alpha_{j,l}x_0^{j-1} \\ &+4\eta(x_1)\sum_{j=0}^{2}\tilde{\psi}_{m,j}\sum_{l=1}^{2} l\alpha_{j,l}x_1^{j-1} + \eta(x_2)\sum_{j=0}^{2}\tilde{\psi}_{m,j}\sum_{l=1}^{2} l\alpha_{j,l}x_2^{j-1}\Bigg]\end{aligned} \tag{26}$$

$$a_{i,j} = h_\tau \sum_{l=1}^{2} l\alpha_{j,l}\sum_{p=0}^{2}\xi_p\eta_i(x_p)x_p^{l-1} \tag{27}$$

Since the jump values in eq. (22b) introduce values from the upstream mesh, those must be included in the source matrix $\left\{S^{n+1}\right\}$. For the mesh of interest, the $\mathbf{B}$ matrix depends upon the value of μ_m.

$$b_{i,j} = \begin{cases}\eta_i(x_0)\delta_{j,0}, & \text{for } \mu_m > 0 \\ \eta_i(x_2)\delta_{j,2}, & \text{for } \mu_m < 0\end{cases} \tag{28}$$

Using the coefficients defined in eqs. (25a), (25b), (27) and (28) and the test functions defined in eq. (6), all the material is in place in order to solve the linear system[3] defined in eq. (21).

3.2 Basic Iteration Scheme

Algorithm 1 gives a brief look into the iterations that take for every time-level n. The special routines that are not detailed in this document, but are listed in Algorithm 1 are:

1. *updateSouce* - line 4: Update the scalar flux $\tilde{\phi}^{(k)}$ using the most recent angular flux coefficients $\tilde{\psi}_{m,j}^{(k-1)}$
2. *innerSolve* - line 13: Solve the linear system from eq. (21) at a specfic angle using $\tilde{\phi}^{(k)}$
3. *convergenceCheck* - line 16: Sweep over all meshes and check if the coefficients have converged to a reasonable degree $\Delta \approx O(1E-6)$

3.3 Using `fesnt.py`

Present in the repository are a collection of yaml input files that were used to generate results for this section. In particular, there are a few gifs that display the time-evolution of the angular fluxes for the test problems. Due to the non-linearity of eq. (2), an analytic can be found only for the most trivial of problems. Therefore, more in-depth analysis will be applied to the pure-absorber problem, with select snapshots and discussion for two more complicated problems: a single-media problem that includes scattering and a mulit-region problem that includes scattering and fission. For all problems, the S_2 and S_4 quadrature sets were used, except for case 3, which includes the S_8 quadrature set exclusively.

[3] iteratively, as discussed in section 3.2

Algorithm 1 Inner iteration at time level n - Python-like

```
isConverged = False
while innerIndex ≤ innerLimit and not isConverged do
    for mesh in meshes do
        mesh.updateSource(innerIndex)
    end for
    for μ_m in μ⃗ do
        if μ_m > 0 then
            sweep ← meshes
        else
            sweep ← meshes[:: −1]                    ▷ Reversed sweep order
        end if
        for mesh in sweep do
            mesh.innerSolve(μ_m, innerIndex, Δt_n)
        end for
    end for
    isConverged ← convergenceCheck()
    innerIndex ← innerIndex + 1
end while
```

4 Results

4.1 Analytic Solution

An analytic solution to eq. (2) was found for the pure absorber case with $\Gamma_b = 0$, $\psi_{m,0} = 0$.

$$\psi(x, mu, t) = \begin{cases} e^{-x\sigma_t/v\mu}\Gamma_a(\mu, t - x/v\mu)u(t - x/v\mu), \text{ for } \mu > 0 \\ 0, \text{ for } \mu < 0 \end{cases} \tag{29}$$

where $u(t - x/v\mu)$ is a shifted step function

$$u(t - a) = \begin{cases} 0, t < a \\ 1, t > a \end{cases} \tag{30}$$

The boundary conditions and domain were chosen as to observe wave propagation through the media. This required also reducing the neutron velocity v by a factor of $1E4$, as higher speeds required very small space-time discretization and induced oscillations. The boundary condition was defined to be

$$\Gamma_a(\mu, t) = \mu sin^2(1000t) \tag{31}$$

which yields an analytic solution plotted in fig. 2 that contains some degree of wave propagation over space and time.

The physical domain was taken to be $x \in \Omega \equiv [0, 1]$, $t \in \mathcal{T} \equiv [0, 0.01]$ with 100 elements per dimension.

4.2 Numeric Results

5 Conclusion

References

[1] E. Lewis and W. Miller, *Computational methods of neutron transport.* Wiley, 1984.

[2] W. H. Reed and T. R. Hill, "Triangular mesh methods for the neutron transport equation," in *National topical meeting on mathematical methods and computational techniques for analysis of nuclear systems*, Ann Arbor, Michigan, USA, 8 April 1973.

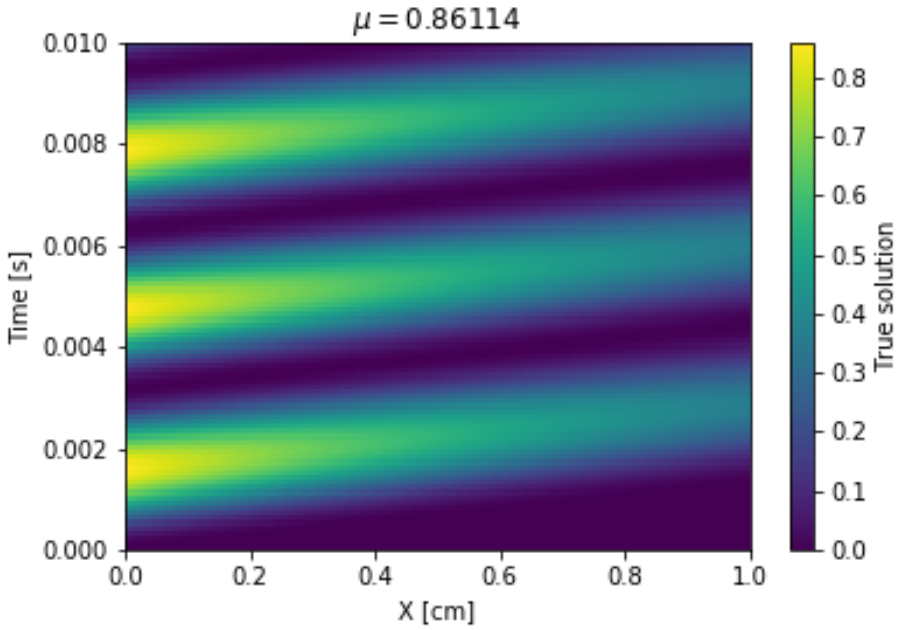

Figure 2: Analytic solution for the pure absorber with S_4 quadrature

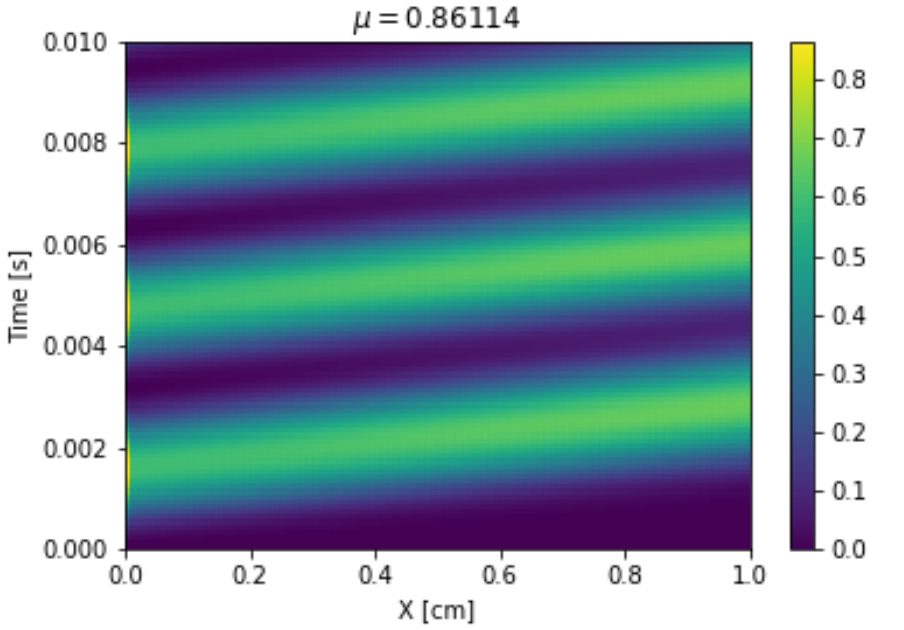

Figure 3: Numerical solution for case 1 using S_4 quadrature

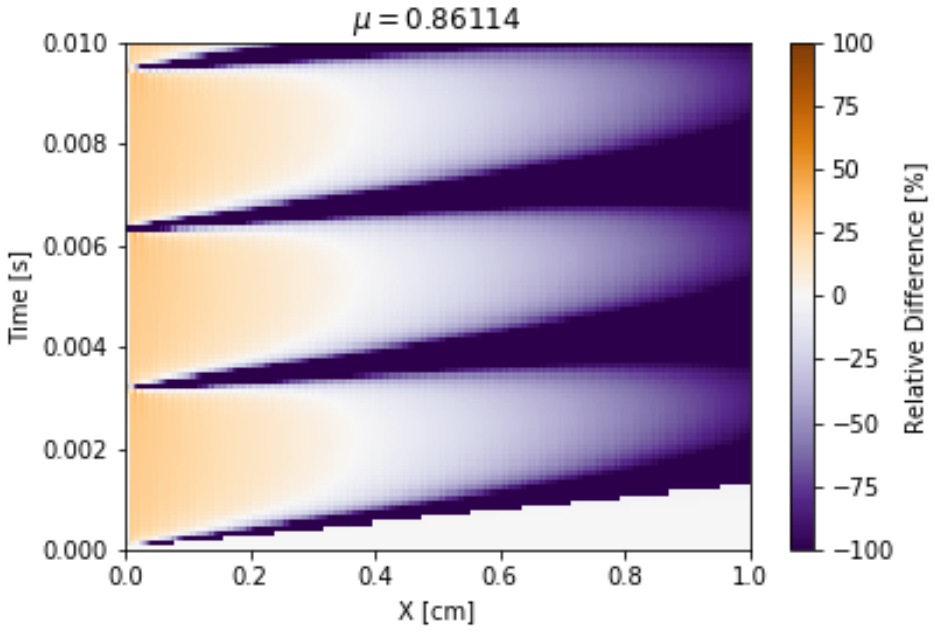

Figure 4: Relative difference between numeric and analytic solution

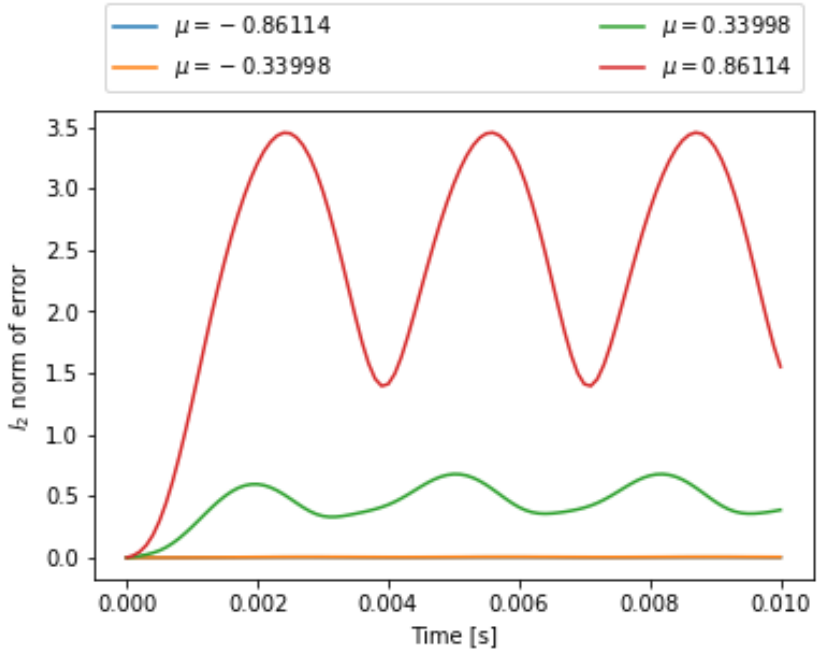

Figure 5: l_2 norm of error over time

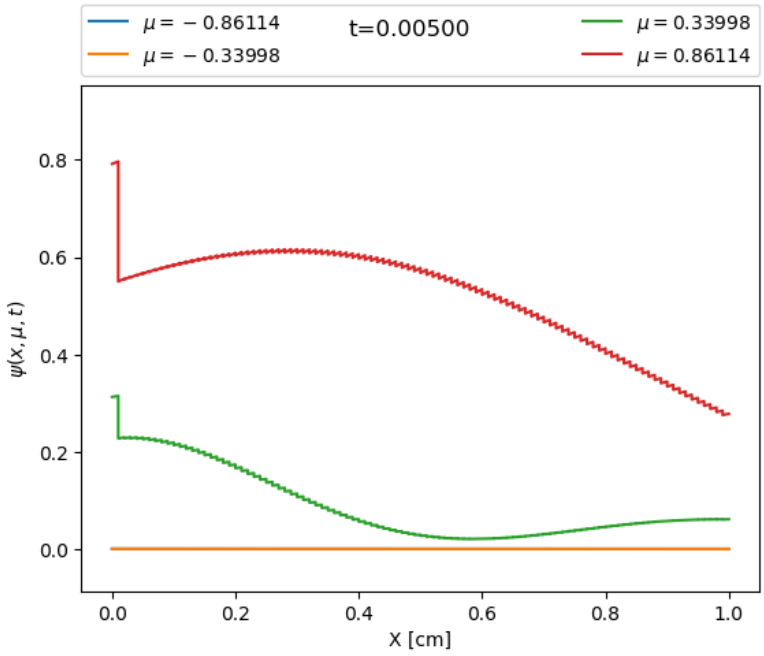

Figure 6: Angular fluxes for Case 2 using S_4 quadrature

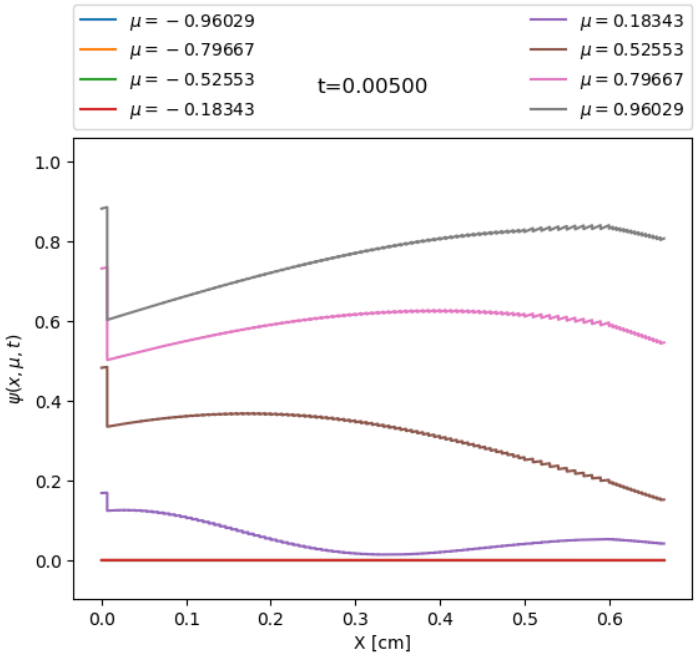

Figure 7: Angular fluxes for the multi-regional problem

6 Appendix

6.1 Quadrature Sets

The following tables contain the quadrature sets that were used in this work and can be found in [1]. For each set, only half of the values are presented, as symmetry rules from eq. (1) can be used to construct the full set.

Table 1: S-2 Weights and Angles

μ_m	w_m
0.5773502691	1.0

Table 2: S-4 Weights and Angles

μ_m	w_m
0.3399810435	0.6521451549
0.8611363115	0.3478548451

Table 3: S-8 Weights and Angles

μ_m	w_m
0.1834346424	0.3626837834
0.5255324099	0.3137066459
0.7966664774	0.2223810344
0.9602898564	0.1012285363