CHALMERS | GÖTEBORG UNIVERSITY

Master's Thesis in Engineering Mathematics

Numerical Algorithms for Electron Beams

Samir Naqos

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg Sweden 2005

Thesis for the degree of M.Sc. in Engineering Mathematics
Examensarbete för civilingenjörsexamen i teknisk matematik

Numerical Algorithms for Electron Beams

Samir Naqos

Supervisor
Dr. Mohammad Asadzadeh

CHALMERS | GÖTEBORG UNIVERSITY

Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, 2005

Abstract

In this thesis we consider the deterministic numerical algorithms for electron beams represented by the Fermi-pencil-beam equation. We consider two solvers based on a fully discrete scheme using the Standard Galerkin (SG) and a Semi Streamline Diffusion (SSD) for discretization in the transversal variable combined with a backward Euler for discretization in the penetration variable, and two solvers based on characteristic schemes, namely the Characteristic Galerkin (CG), and a stabilized Galerkin finite element method referred to as the Characteristic Streamline Diffusion (CSD). A common feature of the mentioned algorithms is exact transport + projection.

Acknowledgements

I am grateful to my supervisor Mohammad Asadzadeh from whom I have learned a lot about mathematics. He introduced me to this subject and suggested the algorithms in the thesis. His encouragements, support, and valuable remarks were essential in my work.

I would like to thank Johan Hoffman and Johan Jansson for their useful suggestions in the code.

I wish to thank my teachers at Chalmers University of Technology and Göteborg University. The quality of the mathematics and the nice atmosphere at the Mathematics Centrum were very important.

Last but not least, I thank my family for their love and support.

Preface

Scope of the document

This report is written for fulfillment of the requirements of the International master programme in Engineering Mathematics at Chalmers University of Technology.

Structure of the document

This thesis report is based on algorithms developed in the papers

- *On the Stability of Characteristic Schemes for the Fermi Equation* [1].
- *On Fully Discrete Schemes for the Fermi Pencil-Beam Equation* [2].

Focus is on implementation and presentation of the results from the different algorithms.
Implementation of the algorithms has been carried out in the open source package *Dynamic Object-oriented Library for FINite element computation; DOLFIN*, a C++ interface of the free software for the Automation of Computational Mathematical Modeling *FEniCS*[1]. Visualization of the results is obtained using Matlab.

This report begins with an introduction to the concepts of radiation oncology, and physical properties of electron beams, then it proceeds with a derivation of the Fermi Pencil-Beam equation and a model problem. Exposition of the numerical algorithms based on Galerkin Finite Element Methods are detailed in chapter 2. Chapter 3 is devoted to a discussion of the implementation. Results are presented and discussed in chapter 4.

[1]www.fenics.org

Contents

Chapter 1

Radiation Oncology

1.1 Introduction

The use of radiation in cancer therapy dates back to the end of the 19th century. Shortly after the discovery of X-rays in 1895, researchers used them for diagnostic and therapeutic treatment of cancer. Since then, radiation in the form of high-energy beams of electrons, photons (X-ray), protons, neutrons and other particles have been used for the same purposes.

Early attempts at radiation were far from optimal and were marked by inaccuracy and failure. Gradually, as medicine took advantage of new advances in different scientific and technological fields, radiation therapy become much more sophisticated. Among the advances, we cite, the availability of new and better computer controlled accelerators for clinical work, the substantial understanding of the underlying physics, the emerging of computational algorithms, the development of Computerized Tomography (CT), Magnetic Resonance Imaging (MRI) and computer graphics software.

The major goal of radiation cancer therapy has been "to maximize the probability of local tumor control with minimal damage in the neighboring healthy tissue". This is an optimization problem which is solved by determining a set of beams while providing the necessary dose to each point in the tumor, minimizes the risk of complication to the neighboring tissue. this problem is solved iteratively by a sequence of dose calculations obtained through a series of computed transport calculations that simulate the effects of beams penetrating the human tissue.

The choice of the form of radiation is subject to clinical considerations.

Electron therapy is used primarily to treat superficial diseases such as skin tumors, providing a unique option in this sense. The hardware used to produce such high-energy electron beams are the so called *linear accelerators* often referred to as *linacs*. They typically deliver 4- to 25 MeV electron beams. 25 MeV is 25 million *electron volts* (eV) (1 eV is the energy needed to move one electron through a potential of one volt).

Passage of radiation through a portion of the body sets in motion a long series of phenomena which result in biological effects. The question of calculating the *dose*, or energy deposited per unit mass, to a patient is a crucial part in the treatment process. These calculations enable the oncologist to decide whether a certain set of beams is optimal.

Nowadays, about half of all cancer patients receive radiation therapy [8], most of the time in combination with other forms of treatment (such as surgery, chemotherapy, hyperthermia. . .). Surely, any improvements to radio therapy, even small improvements, will benefit a great number of people.

1.2 Radiation Therapy Planning

First, the radiation treatment of a tumor begins with the creation of a three dimensional image of the tumor and surrounding healthy organs. Using Computed Tomography or MRI and some suitable computer graphics packages.

Next, the oncologist, being able to visualize the geometry and position of the tumor and neighboring regions on the computer, selects a set of beams that maximize the control of tumor without affecting the surrounding healthy tissue. It is worth mentioning here that the oncologist rely on experience and intuition while performing this step.

After choosing the set of beams, the oncologist performs a 3-D particle transport calculation in which the *dose*, or energy deposited per unit mass, is calculated within each *voxel*, which represents approximately 1 mm^3 in volume. This step is referred to as *dose calculations*. Generally, clinical dose calculations must be performed in less than ten minutes [15].

The dose calculations are then plotted into a graph typically consisting of dose-volume histograms. The histogram allows the oncologist to investigate which fraction of the volume of an affected organ receives more than a given percent of the total dose.

Using this information the oncologist decides whether the original configuration of beams is optimal or not. If not, the oncologist chooses another configuration and a second dose calculation is performed. This step is carried out iteratively (from two to six times in a typical treatment) until the oncologist decides that further iterations will not be needed.

As mentioned earlier, the oncologist decides on a configuration of beams based on experience and intuition. Usually taking certain conflicting factors into consideration, to name a few

- There may be geometrical uncertainties. For instance, the boundaries of the tumor are most of the time not easy to observe and determine. This is mainly due to the difficulties that may arise in distinguishing between healthy and cancerous tissue. Another factor to consider is *microscopic invasions* by the cancer in the healthy tissue. Other sources that may lead to geometrical uncertainties are patient breathing and motion, which may lead to inaccurate CT or MRI images.

- Biological uncertainties may be a source of difficulty for the oncologist. An example of such uncertainty is the fact that individual organs have different tolerances for radiation. Also, individual patients have different tolerances for radiation.

- The probability of tumor control will be low if *cold spots*, that is voxels receiving less than the necessary dose, exist in the target volume.

- There should be no *hot spots* (voxels receiving more than the necessary dose) within the target volume.

- The oncologist must always keep in focus the purpose of the therapy. Sometimes the goal is to reduce the size of a tumor prior to surgery or to irradiate an area from which a tumor has been surgically removed; to destroy possible microscopic tumors that were not removed by surgery.

The above procedure, once performed, constitutes a radiation therapy plan. The patient receives radiation therapy accordingly. However, the treatment is *fractionated*, that is, delivered in N multiple sessions; often $N > 20$ (see [8]). This fractionated treatment is of great importance, healthy tissue repairs itself from the damage caused by low-level radiation doses more efficiently than cancerous tissue. And evidently, the cumulative damage is less in a fractionated treatment than in a single dose treatment.

To sum up, the radiation therapy is a complex process. The first stage consists of a treatment planning of radiation therapy, in which the oncologist performs a sequence of dose calculations using the computer. These calculations simulate 3-D, heterogeneous-media, highly anisotropic-scattering particle transport processes involving beams of high-energy photons and/or electrons.

1.3 Electron beams: Physical aspects

1.3.1 Physical Properties

The efficient use of radiation therapy requires a thorough understanding of the underlying physics. In this section, we outline some of the concepts in question regarding electron beams.

A beam of electrons is a concentrated and highly charged stream of electrons generated by the acceleration and conversion of electricity using linacs. Interactions occur while electron beams penetrate a background (a portion of a patients body and the surrounding air).

Made up of particles, electron beams are referred to as *radiation particles* and the tumor as *background particles.* Radiation particles interact with background particles by a variety of elastic and inelastic *Coulomb forces.* Consequently, the interactions result in setting up the background particles in rapid motion, hence becoming themselves radiation particles. The dose, or energy deposited per unit mass, is an outcome of excitation and ionization events.

Basically, energy deposition by electrons occurs through two mechanisms, namely, *collisional losses* and *radiative losses.* In collisional losses, electrons lose energy via interactions with orbital electrons of the atoms in the medium. This leads to excitations of the atoms or *ionization.*

The second form of energy deposition by electrons is through radiative losses or *bremsstrahlung*, which is an electromagnetic radiation that occurs when a charged particle undergoes a change in acceleration. The larger the change in acceleration, the more energetic the bremsstrahlung photon. The efficiency of bremsstrahlung in elements of different atomic number Z varies nearly as Z^2 (see [8] for a more detailed discussion).

As the electron beam traverses the patient, because of the *collisional* and *radiative losses*, its mean energy decreases and its angular spread increases.

Furthermore, electrons, being *light particles*, collide with particles of identical mass leading to large scattering angles. This results in a track that is very devious instead of a straight path as in the case of *heavy particles.*

Besides, electrons are less penetrating than heavy particles of similar energy. This is why electron beams offer a distinct clinical advantage in the treatment of superficial tumors. Actually, the lowest energy electron beams do not penetrate tissues while the higher strength beams do penetrate tissues to a moderate extent and then their energy drops off rapidly.

The two graphs below illustrate this physical feature of electron beams. The percent dose depth is the ratio of dose at a given point on the central axis of an electron beam to the maximum dose on the central axis multiplied by 100.

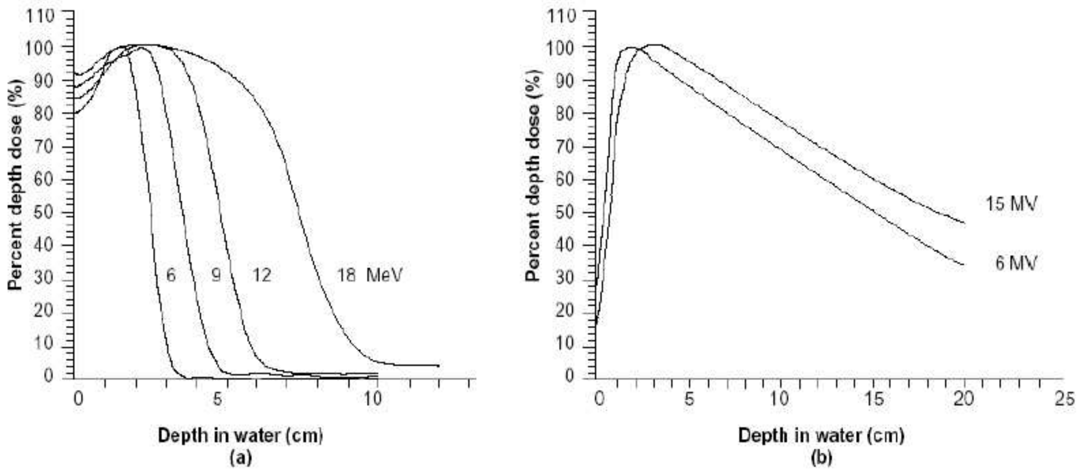

Figure 1.1: percentage depth dose curves in water (a) electron beams with energies of 6,9,12 and 18 MeV and (b) photon beams with energies of 6 MV and 15 MV. (Graphs courtesy of [16])

1.3.2 Mathematical models

The mathematical formulations that model the dose, or energy deposited per unit mass are variants of the *Boltzmann transport equation.*

The *multiple-scattering theory* developed by *Fermi* and others offers a fairly simple description of the penetration of a pencil beam of electrons through a background. Numerical algorithms resulting from this approach are suitable for implementation in treatment planning computer programmes.

The basic idea of This approach is the following, the incoming radiation is thought of as composed of a finite number of pencil beams, that is infinitesimally thin, mono-directional, mono-energetic beams. Mathematically, this means approximation of the initial data by a finite sum of *Dirac delta functions.*

In addition to the above mentioned approaches, there are plenty of *Monte Carlo* methods applied for the same purposes.

1.4 Algorithms used: Stochastic vs Deterministic

Motivated by accuracy and fast performance, the determination of doses is approached by two different families of algorithms. Mainly, *stochastic* and *deterministic.* While there are pros and cons of applying each set for a certain purpose. The determination of doses remains computationally challenging. In this section we state some of the used algorithms and we refer to the literature for more detailed approaches about the subject.

In current day clinical work, dose calculations are governed by using semi-empirical methods based on a combination of analysis and laboratory experiments. A typical procedure for radiotherapy planning based on this approach can be found in [13]. However, current efforts are being made to develop algorithms from both families.

To begin with, Monte Carlo methods are well suited in the presence of physical and geometrical complexity. With arbitrarily high accuracy, Monte Carlo methods have long been used for solving medical physics problems [15]. Some commercial Monte Carlo packages such as EGS, GEANT, MMC, VMC, and TIGER are used among others [15]. However, the use of these codes is mostly restricted for benchmarking and for determining dose deposition kernels. There is a severe limitation for their use clinically, mainly due to the fact that the solution of dose calculation problems with small statistical errors requires these codes to be run about 1000 times (while considering individual voxels) than the desired 10 minutes [17].

On the other hand, deterministic algorithms are fast. Recent publications in radiation oncology show that the future of the field will be governed by variants of deterministic algorithms. A mathematical study of the complexity of deterministic and Monte Carlo dose calculation methods advocating this can be found in [6].

A survey and a detailed discussion about the use of algorithms from both families can be found in [6], and [13].

Present day deterministic algorithms include finite difference (FD) or finite element (FE) discretization of one of the following models

- The Fermi pencil beam equation
- Generalizations of the pencil beam methods such as the *Fokker-Planck* equation
- A system of linear Boltzmann equations describing photon/electron/positron transport using six-dimensional phase space grid.
- Phase space time evolution or the PSTE method.

Again, there exist variants of the use of discretization techniques to handle each model, depending on the type of the numerical method used. These methods have had varying degrees of success.

Because of their deterministic nature, the above FD and FE methods are free of statistical errors. However, some of them are not used clinically on a wide scale mainly because of the computational cost.

The algorithms dealt with in this thesis are deterministic algorithms based on the variants of *General Galerkin* or the G2-method which uses piecewise polynomials in space-time. The algorithms are developed in two papers by Asadzadeh in [1], and Asadzadeh and Sopasakis in [2].

In [1] using *Characteristic Galerkin* and *Characteristic Streamline diffusion* methods. In [2] using Fully discrete schemes based on *Standard Galerkin* and a *Semi-Streamline Diffusion* in (y, z) combined with a *Backward Euler* in x.

According to [12], "G2 is regarded as *Eulerian* if the space-time mesh is oriented along the space and time coordinate axis, *Lagrangean* if the space-time mesh is oriented along particle paths in space-time, and *arbitrary Lagrangean-Eulerian* or ALE if the space-time mesh is oriented according to some other feature such as space-time gradients of the solution. We also refer to *Lagrangean* variants as *Characteristic Galerkin*, ALE-methods as *oriented Galerkin*, and *Eulerian* variants as SUPG and *Streamline Diffusion-methods.*"

Generally, the algorithms presented here can be classified as a combination of Eulerian, Lagrangean, and Eulerian-Lagrangean G2.

Chapter 2

Algorithms for Electron Beams

Clinical dose calculation algorithms for electron beams are usually based on a mathematical model, originally formulated by Fermi. Fermi derived the model in his research concerning cosmic rays [11].

Fermi model describes the broadening of a narrow *pencil beam* of particles. The particles in the beam undergo scattering events in which their directions change only slightly, then the directions of flight of the particles will slowly (with respect to a mean free path) "diffuse" away from the initial direction, and the width of the beam will slowly increase with depth. The mathematical problem is to determine *quantitatively* the broadening of the beam as it penetrates into the system.

2.1 Derivation of the Fermi Pencil Beam Equation

In this section, we sketch the derivation of the pencil beam equation. We start from the steady-state, monoenergetic transport equation in a homogeneous slab $\tilde{Q} := [0, L] \times \mathbf{R} \times \mathbf{R}$, given by

$$\omega \cdot \nabla_{\boldsymbol{x}} \psi(\boldsymbol{x}, \omega) + \sigma_t(\boldsymbol{x}) \psi(\boldsymbol{x}, \omega) = \int_{S^2} \sigma_s(\boldsymbol{x}, \omega \cdot \omega') \psi(\boldsymbol{x}, \omega) \, d\omega' \quad \text{in} \quad \tilde{Q} \times S^2, \tag{2.1}$$

and associated with the boundary conditions

$$\begin{cases} \psi(L,y,z,\omega)=0 & \text{if } \xi<0, \\ \psi(0,y,z,\omega)=\frac{1}{2\pi}\delta(1-\xi)\delta(y)\delta(z) & \text{if } \xi>0, \end{cases} \tag{2.2}$$

with $\boldsymbol{x}=(x,y,z)\in\tilde{Q}, \omega=(\xi,\eta,\zeta)\in S^2$, describing the spreading of a pencil beam of particles normally incident at the boundary $(0,y,z)$ of the slab $\tilde{Q}$. Here ψ is the density of particles at the point $\boldsymbol{x}$ moving in the direction of ω. σ_t is the total cross section, whereas σ_s is the scattering cross section. Assuming forward peaked scattering, the transport equation (2.1) is asymptotically approximated by the following Fokker-Planck equation

$$\omega\cdot\nabla_{\boldsymbol{x}}\psi^{FP}=\sigma\left[\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial}{\partial\xi}+\frac{1}{1-\xi^2}\frac{\partial^2}{\partial\vartheta^2}\right]\psi^{FP}, \tag{2.3}$$

where ϑ is the azimuthal angle with respect to the z-axis and

$$\sigma\equiv\frac{1}{2}\sigma_{tr}(\boldsymbol{x})=\pi\int_{-1}^{1}(1-\xi)\sigma_s(\boldsymbol{x},\xi)\,d\xi, \tag{2.4}$$

is the transport cross-section for a purely scattering medium.
Assuming further simplifications, and approximations which can be found in detail in [7]. The following Fermi equation is derived from (2.3):

$$\begin{cases} \omega_0\cdot\nabla_{\mathrm{X}}\psi^F=\sigma\Delta_{\eta\zeta}\psi^F, & \\ \psi^F(0,y,z,\eta,\zeta)=\delta(y)\delta(z)\delta(\eta)\delta(\zeta), & \text{if } \xi>0, \\ \psi^F(L,y,z,\eta,\zeta)=0, & \text{if } \xi<0, \end{cases} \tag{2.5}$$

here $\omega_0=(1,\eta,\zeta)$, where $(\eta,\zeta)\in\mathbf{R}\times\mathbf{R}$ and $\Delta_{\eta,\zeta}=\partial^2/\partial\eta^2+\partial^2/\partial\zeta$.
The operator L_F on the right hand side of (2.5) is related to the Fokker-Planck operator L_{FP} on the right hand side of (2.3). Geometrically, the equation (2.5) corresponds to projecting $\omega\in S^2$ in the equation (2.3), along $\omega=(\xi,\eta,\zeta)$, on the tangent plane to S^2 at the point $(1,0,0)$. In this way, if one applies the Laplacian operator (multiplied by σ) in the spherical coordinates (r,η,ϑ) to a function that is independent of r and then sets $r=1$, the resulting expression reduces exactly to L_F. Hence, except for the factor σ, the *Fokker-Planck* operator is the Laplacian operator defined on the unit sphere, and the Fermi operator L_F is just the Laplacian operator defined on the tangent plane to the unit sphere at the point $\xi=1,\eta=\zeta=0$. Thus, L_F should be a good approximation to L_{FP} for small η and ζ.

2.2 A model Problem

The equations (2.3) and (2.5) are formulated for the flux ψ, a measure of interest to nuclear engineers. Medical physicists are mainly concerned with the dose (energy deposited per unit mass). The dose is related to the current function

$$j = \xi\psi. \tag{2.6}$$

Now, we consider a two dimensional version of (2.1)-(2.5) leading to the *Fokker-Planck* problem below. Detailed treatment of this equation can be found in [3] and [4].

For $0 < x < L$ and $-\infty < y < \infty$, find $\psi_{FP} \equiv \Psi_{FP}(x, y, \theta)$ such that

$$\begin{cases} \omega \cdot \nabla_{\mathrm{X}}\psi^{FP} = \sigma\psi^{FP}_{\theta\theta}, & \theta \in (-\pi/2, \pi/2), \\ \psi^{FP}(0, y, \theta) = \frac{1}{2\pi}\delta(1 - cos(\theta))\delta(y), & \theta \in S^1_+, \\ \psi^{FP}(L, y, \theta) = 0, & \theta \in S^1_-, \end{cases} \tag{2.7}$$

where $\omega := (\xi, \eta) \equiv (cos(\theta), sin(\theta))$, $S^1_+ = \omega \in S^1 : \xi > 0$ and $S^1_- = S^1 \setminus S^1_+$.

Using the scaling substitution

$$z = tan(\theta), \qquad \theta \in (-\pi/2, \pi/2), \tag{2.8}$$

now, we introduce the scaled current function J as

$$J(x, y, z) \equiv \frac{j(x, y, tan^{-1}z)}{(1 + z^2)}. \tag{2.9}$$

Note that z corresponds now to the angular variable θ. Below, we shall keep θ away from the poles $\pm\pi/2$, and correspondingly formulate a problem for the current function J, in the bounded domain $Q \equiv I_x \times I_y \times I_z = [0, L] \times [-y_0, y_0] \times [-z_0, z_0]$:

$$\begin{cases} J_x + zJ_y = \epsilon AJ, & (x, x_\perp) \in Q, \\ J_z(x, y, \pm z_0) = 0, & \text{for}(x, y) \in I_x \times I_y, \\ J(x, \pm y_0, z) = 0, & \Gamma^-_{\tilde{\beta}} \setminus \{suppf\}, \\ J(0, x_\perp) = f(x_\perp), & \end{cases} \tag{2.10}$$

where $\Gamma^-_{\tilde{\beta}} := \{(x, x_\perp) \in \partial Q : \tilde{\beta} \cdot \boldsymbol{n} < 0\}$, $\tilde{\beta} = (1, z, 0)$, $x_\perp \equiv (y, z)$ is the transversal variable and $\boldsymbol{n} := n(x, x_\perp)$ is the outward unit normal to Γ at

$(x, x_\perp) \in \Gamma$. We have also $2\epsilon = \sigma_{tr}(x, y)$, $\sigma_{tr}$is the transport cross section, a small positive decreasing function of (x, y) indicating energy deposit due to particle collisions. Further, we have replaced the product of δ-functions (the source term) at the boundary by a smoother L_2-function f. The diffusion operator in (2.10) is

$$A = \partial^2/\partial z^2, \qquad \text{Fermi,} \tag{2.11}$$

$$A\cdot = \partial/\partial z[a(z)\partial/\partial z(b(z)\cdot)], \qquad \text{Fokker-Planck,} \tag{2.12}$$

where $a(z) = 1 + z^2$ and $b(z) = (1 + z^2)^{3/2}$.

In this thesis, we study the Fermi equation. Detailed studies of both equations using the streamline diffusion method can be found in a series of papers, notably in [3] and [4].

The corresponding Fermi equation is modeling the penetration (in the direction of the x-axis) of a narrowly focused pencil beam incident at the transversal boundary of an *isotropic* slab entering to the domain at the point $(x, y, z) = (0, 0, 0)$. While considering an isotropic background media, we may assume that all involved functions are symmetric, i.e, even in y and z.

Our model problem corresponds to a forward-backward (z changes the sign), convection dominated (ϵ is small), convection-diffusion equation of degenerate type (convection in (x, y) and diffusion in z).

In section (2.3) we deal with the discretization of this equation using fully discrete schemes. In section (2.4) we introduce a corresponding non-degenerate equation and apply characteristic schemes. Throughout the report, C will denote an absolute constant not necessarily the same at each occurrence, unless otherwise explicitly stated.

2.3 The fully discrete scheme

We now consider the semidiscrete schemes where the model problem is discretized in the transversal variable $x_\perp := (y, z)$ using the *Standard Galerkin* (SG) and the *Semi-Streamline Diffusion* (SSD) finite element methods. Because of the structure of the equation, the penetrating variable x is interpreted as a time variable and treated by usual time discretization such as *backward Euler*, *Crank-Nicolson*..etc. leading to a fully discrete scheme. It should be noted here that the SSD method is performed only on the $x_\perp$ variable, whereas the usual streamline diffusion (SD) finite element method is applied also on the x variable. The full SD is considered in [3].

We recall the model problem

$$\begin{cases} u_x + zu_y = \epsilon u_{zz}, & (x, x_\perp) \in Q, \\ u_z(x, y, \pm z_0) = 0, & \text{for}(x, y) \in I_x \times I_y, \\ u(x, \pm y_0, z) = 0, & \Gamma^-_{\tilde{\beta}} \setminus \{suppf\}, \\ u(0, x_\perp) = f(x_\perp), \end{cases} \tag{2.13}$$

where $\Gamma^-_{\tilde{\beta}} := \{(x, x_\perp) \in \partial Q : \tilde{\beta} \cdot \boldsymbol{n} < 0\}$, $\tilde{\beta} = (1, z, 0)$, and $x_\perp \equiv (y, z)$ is the transversal variable. Further, $\boldsymbol{n} := n(x, x_\perp)$ is the outward unit normal to Γ at $(x, x_\perp) \in \Gamma$.

2.3.1 Standard Galerkin

Discretization

First, we introduce a slab of thickness L, $x \in I_x := [0, L]$, with a symmetric cross section $I_\perp := I_y \times I_z := [-y_0, y_0] \times [-z_0, z_0]$, for $(y_0, z_0) \in \mathbf{R}^2_+$. The physical domain $I_x \times I_\perp$ is now three dimensional. Next, we discretize in $x_\perp = (y, z)$ using a finite element approximation based on a quasi-uniform triangulation of the rectangular domain $I_\perp = I_y \times I_z$ with a mesh size h. We also consider adaptive meshes with refinements in the center. We let $\beta = (z, 0)$ and define the inflow (outflow) boundary as

$$\Gamma^{-(+)}_{\beta} := \{x_\perp \in \Gamma := \partial I_\perp : \boldsymbol{n(x_\perp)} \cdot \beta < 0 (> 0)\}, \tag{2.14}$$

where $\boldsymbol{n}(x_\perp)$ is the outward unit normal to the boundary Γ at $x_\perp \in \Gamma$. Now, we introduce a discrete, finite dimensional, function space $V_{h,\beta} \subset H^1_\beta(I_\perp)$ with,

$$H^1_\beta(I_\perp) = \{v \in H^1(I_\perp) : v_z(\pm z_0) = 0 \quad \text{and} \quad v = 0 \quad \text{on} \quad \Gamma^-_\beta\}, \tag{2.15}$$

such that, $\forall v \in H^1_\beta(I_\perp) \cap H^r(I_\perp)$,

$$\inf_{\chi \in V_{h,\beta}} \|v - \chi\|_j \le Ch^{\alpha - j} \|v\|_\alpha, \qquad j = 0, 1 \qquad \text{and} \qquad 1 \le \alpha \le r, \tag{2.16}$$

$\|\cdot\|_s$, with s being a positive integer, denotes the L_2 based *Sobolev* norm of functions that are square integrable along with all their partial derivatives of order $\le s$. An example of such $V_{h,\beta}$ is the set of sufficiently smooth piecewise polynomials $P(x_\perp)$ of degree $\le r$, satisfying the boundary conditions given

in (2.13).
We then seek $u_h \in V_{h,\beta}$, such that

$$\begin{cases} (u_{h,x}, \chi)_\perp + (zu_{h,y}, \chi) + (\epsilon u_{h,z}, \chi_z)_\perp = 0, & \forall \chi \in V_{h,\beta}, \\ u_h(0, x_\perp) = f_h(x_\perp), \end{cases} \tag{2.17}$$

where f_h is a finite element approximation of f. The mesh size h is related to ϵ according to:

$$h^2 \leq \epsilon \leq h. \tag{2.18}$$

Here,

$$(u, v)_\perp = \int_{I_\perp} u(x_\perp) v(x_\perp)\, \mathrm{d}x_\perp, \qquad \text{and} \qquad \|u\|_{L_2(I_\perp)} = (u, u)_\perp^{1/2}.$$

Stability

In this part, we prove a stability estimate in the inner product $(\cdot, \cdot)_\perp$. We restrict the scope to the case of the semi-discrete SG case. more detailed stability analysis can be found in [2].

Lemma 1 *For $u_h \in V_{h,\beta}$, satisfying (2.17) we have*

$$\sup_{x \in I_x} \|u_h(x, \cdot)\|_{L_2(I_\perp)} \leq \|f\|_{L_2(I_\perp)}. \tag{2.19}$$

Proof: If we choose $\chi = u_h$, in the first equation of (2.17), we get

$$\frac{1}{2} \frac{d}{dx} \|u_h(x, \cdot)\|^2_{L_2(I_\perp)} + (zu_{h,y}, u_h)_\perp + \|\epsilon^{1/2} u_{h,z}\|^2_{L_2(I_\perp)} = 0. \tag{2.20}$$

Integration by parts in y yields

$$(zu_{h,y}, u_h)_\perp = \frac{1}{2} \int_{I_x} z(u_h^2(y_0) - u_h^2(-y_0))\, dz = \frac{1}{2} \int_{\Gamma_\beta^+} (n \cdot \beta) u_h^2\, d\Gamma. \tag{2.21}$$

Inserting (2.21) in (2.20), we get

$$\frac{1}{2} \frac{d}{dx} \|u_h(x, \cdot)\|^2_{L_2(I_\perp)} + \frac{1}{2} \int_{\Gamma_\beta^+} (n \cdot \beta) u_h^2\, d\Gamma + \|\epsilon^{1/2} u_{h,z}\|^2_{L_2(I_\perp)} = 0. \tag{2.22}$$

Now, since $\frac{1}{2}\int_{\Gamma_\beta^+}(n\cdot\beta)u_h^2\,d\Gamma \geq 0$, then by (2.22), $\frac{1}{2}\frac{d}{dx}\|u_h\|^2_{L_2(I_\perp)} \leq 0$. Therefore $\|u_h\|^2_{L_2(I_\perp)}$ is decreasing in x and hence,

$$\|u_h(x,\cdot)\|^2_{L_2(I_\perp)} \leq \|f\|^2_{L_2(I_\perp)}, \qquad \forall x \in [0,L]. \tag{2.23}$$

The next estimate, which we state without proof, concerns the continuous version of *lemma 1.* The proof follows the same setup. The starting point is to consider the continuous variational formulation.

Corollary 1 *The solution u of (2.13) satisfies the stability relation*

$$\sup_{x\in I_x}\|u(x,\cdot)\|_{L_2(I_\perp)} \leq \|f\|_{L_2(I_\perp)}. \tag{2.24}$$

Convergence

For convection dominated problems, being hyperbolic, the standard finite element schemes would have convergence rate of the same order as the assertion in the theorem below, which is the known optimal convergence rate for purely hyperbolic problems. However, this convergence rate is not obvious when degeneracy and type change are included. We state the theorem and a sketch of the proof.

Theorem 1 *For $u \in H^r(\Omega)$, satisfying (2.13) and with u_h being the solution of (2.17), there is a constant $C = C(\Omega, f)$ such that*

$$\|u-u_h\|_{L_2(\Omega)} \leq Ch^{r-1}\|u\|_r. \tag{2.25}$$

proof: This follows trivially from applying *Poincare's* inequality,

$$\|u-u_h\|_{L_2} \leq C\|(u-u_h)_z\|_{L_2}. \tag{2.26}$$

Using the following result from [2]

$$\|\epsilon^{1/2}(u-u_h)_z\|_{L_2} \leq Ch^{r-1/2}\|u\|_r, \tag{2.27}$$

and since $\epsilon \sim h$, the result follows.

2.3.2 A Semi Streamline Diffusion Method

Basically, the Streamline Diffusion Method is obtained by multiplication with *test* functions belonging to a space which is different from the space of *trial* functions where the discrete solution is sought. Such a method, where the *test* functions are different from the *trial* functions is referred to as a *Petrov-Galerkin* method. More generally, The Streamline Diffusion is also considered as an *Eulerian G2* method, see [12].

Below, we introduce a semi Streamline Diffusion approach with diffusion generating test functions in the y, z directions. This scheme is strongly stable (see [14]). Its smoothing properties are seen in the numerical implementation in section 4. The convergence rates are at least as good as those of the SG method, we refer to [3] for the complete analysis in the full SD case.

It is worth mentioning here that by using the SSD, we obtain a non-degenerate type convection dominated, convection-diffusion equation, with somewhat improved regularity. Yet, our test functions have the form $v+\delta v_\beta$. Such test functions automatically add up the extra diffusion term $\delta(v_\beta, v_\beta)$ to the variational formulation, which combined with $(v, -\epsilon v_{zz}) = (\epsilon v_z, v_z)$ leads to a non-degenerate fully diffusive equation (x is interpreted as a time variable). If $\delta \geq \epsilon$ the diffusion term is of order ϵ. We assume that $\delta \sim h$.

Discretization

The *test* function has the form $v + \delta v_\beta$ with $\delta \geq \epsilon$, $\beta = (z, 0)$, $v_\beta = \beta \cdot \nabla_\perp v$ and $\nabla_\perp = (\partial/\partial y, \partial/\partial z)$, and v satisfies the boundary conditions in (2.13). Multiplying the differential equation in (2.13) by $v+\delta v_\beta$ and integrating over $I_\perp$ yields,

$$\begin{aligned}(u_x + u_\beta - \epsilon u_{zz}, v + \delta v_\beta)_\perp = (u_x, v)_\perp + \delta(u_x, v_\beta)_\perp + (u_\beta, v)_\perp \\ +\delta(u_\beta, v_\beta)_\perp + (\epsilon u_z, v_z)_\perp + \delta(\epsilon u_z, (v_\beta)_z)_\perp = 0. \qquad (2.28)\end{aligned}$$

Stability

Our aim is to derive a stability estimate, we let $v = u$ in (2.28). This gives

$$\begin{aligned}\frac{1}{2}\frac{d}{dx}\|u\|^2_{I_\perp} + \delta(u_x, u_\beta)_{I_\perp} + \frac{1}{2}\int_{\Gamma^+_\beta}(n\cdot\beta)u^2\, d\Gamma \\ +\delta\|u_\beta\|^2_{I_\perp} + \|\epsilon^{1/2}u_z\|^2_{I_\perp} + \delta(\epsilon u_z, (u_\beta)_z)_{I_\perp} = 0.\end{aligned}$$

The inner product in the last term can be expressed as

$$\begin{aligned}(\epsilon u_z, (zu_y)_z)_{I_\perp} &= (\epsilon u_z, zu_{yz})_{I_\perp} + (\epsilon u_z, u_y)_{I_\perp} \\ &= \frac{1}{2}\frac{d}{dy}(\int_{I_\perp} \epsilon z u_z^2\, dy\, dz) - \frac{1}{2}\frac{d}{dy}(\int_{I_\perp} \epsilon_y z u_z^2\, dy\, dz) + (\epsilon u_z, u_y)_{I_\perp}.\end{aligned} \tag{2.29}$$

Now, using the symmetry assumption, u is even in y and z, and so is u_z^2. Therefore the integrands above are odd functions in z. Hence, their integral over the symmetric interval are identically zero. (2.29) is reformulated as

$$\begin{aligned}\frac{1}{2}\frac{d}{dx}\|u\|_{I_\perp}^2 + \delta(u_x, u_\beta)_{I_\perp} + \frac{1}{2}\int_{\Gamma_\beta^+}(n\cdot\beta)u^2\, d\Gamma \\ +\delta\|u_\beta\|_{I_\perp}^2 + \|\epsilon^{1/2}u_z\|_{I_\perp}^2 + (\epsilon u_z, u_y)_{I_\perp} = 0.\end{aligned} \tag{2.30}$$

We multiply the differential equation in (2.10) by δu_x, integrate over $I_\perp$ and perform an integration by parts to get

$$\delta\|u_x\|_{I_\perp}^2 + \delta(u_x, u_\beta)_{I_\perp} + \delta(\epsilon u_z, u_{xz})_{I_\perp} = 0. \tag{2.31}$$

Note that

$$(\epsilon u_z, u_{xz})_{I_\perp} = \frac{1}{2}\frac{d}{dx}\int_{I_\perp} \epsilon u_z^2\, dx_\perp - \frac{1}{2}\int_{I_\perp} \epsilon_x u_z^2\, dx_\perp. \tag{2.32}$$

Adding (2.30) and (2.31) and using (2.32) we get,

$$\begin{aligned}\frac{1}{2}\frac{d}{dx}\|u\|_{I_\perp}^2 &+ \delta\|u_x + u_\beta\|_{I_\perp} + \frac{1}{2}\int_{\Gamma_\beta^+}(n\cdot\beta)u^2\, d\Gamma + \|\epsilon^{1/2}u_z\|_{I_\perp}^2 + (\epsilon u_z, u_y)_{I_\perp} \\ &+ \frac{\delta}{2}\frac{d}{dx}\int_{I_\perp}\epsilon u_z^2\, dx_\perp - \frac{\delta}{2}\frac{d}{dx}\int_{I_\perp}\epsilon_x u_z^2\, dx_\perp = 0.\end{aligned} \tag{2.33}$$

Using the following trivial inequality

$$(\epsilon u_z, u_y)_{I_\perp} \le \frac{1}{2}\|\epsilon^{1/2}u_z\|_{I_\perp}^2 + \frac{1}{2}\|\epsilon^{1/2}u_y\|_{I_\perp}^2. \tag{2.34}$$

We make an additional symmetry assumption on the transversal plane viz,

$$\|\epsilon^{1/2}u_z\|_{I_\perp} \sim \|\epsilon^{1/2}u_y\|_{I_\perp}, \tag{2.35}$$

$\epsilon = \frac{1}{2}\sigma_{tr}(x,y) \sim 1/l$, where the mean free path l is an increasing function of x and y. This agrees with the physical fact. Actually, our model starts with dense collisions which gradually, towards the penetration direction x, transfers to a particle distribution with rarefied character on leaving the physical domain, see [2]. Thus ϵ is decreasing and $\epsilon_x \le 0$, therefore

$$\int_{I_\perp} \epsilon_x u_z^2 \, dx_\perp \le 0. \tag{2.36}$$

Inserting (2.34-2.36) in (2.33), we get,

$$\begin{aligned}\frac{1}{2}\frac{d}{dx}(\|u\|_{I_\perp}^2 + \delta \int_{I_\perp} \epsilon_x u_z^2 \, dx_\perp) + \frac{1}{2}\int_{\Gamma_\beta^+} (n\cdot\beta) u^2 \, d\Gamma \\ +\delta \|u_x + u_\beta\|_{I_\perp}^2 + (1-\delta)\|\epsilon^{1/2} u_z\|_{I_\perp}^2 \le 0. \end{aligned}\tag{2.37}$$

Thus for sufficiently small $\delta(\sim \sqrt{\epsilon})$

$$\frac{d}{dx}(\|u\|_{I_\perp}^2 + \delta \int_{I_\perp} \epsilon_x u_z^2 \, dx_\perp) < 0, \tag{2.38}$$

and hence $(\|u\|_{I_\perp}^2 + \delta \int_{I_\perp} \epsilon_x u_z^2 \, dx_\perp)$ is strictly decreasing in x. Consequently, we have $\forall x' \in [0, L]$,

$$\|u(x',\cdot)\|_{L_2(I_\perp)}^2 + \delta\|\epsilon^{1/2} u_z(x',\cdot)\|_{L_2(I_\perp)}^2 \le \|u(0,\cdot)\|_{L_2(I_\perp)}^2 + \delta\|\epsilon^{1/2} u_z(x',\cdot)\|_{L_2(I_\perp)}^2 \tag{2.39}$$

and in particular, we have the following result, which we state as a lemma.

Lemma 2 *Assuming (2.35) and with $\delta > \epsilon$ we have the stability estimate*

$$\|u(L,\cdot)\|_{L_2(I_\perp)}^2 + \delta\|\epsilon^{1/2} u_z(L,\cdot)\|_{L_2(I_\perp)}^2 \le \|f\|_{L_2(I_\perp)}^2 + \delta\|\epsilon^{1/2} f_z\|_{L_2(I_\perp)}^2. \tag{2.40}$$

2.3.3 The Fully Discrete Problem

In this section we derive the algorithms corresponding to the SSD scheme for $I_\perp$ combined with backward Euler(BE) method for the penetration interval I_x. The motivation behind treating the discretization in the x variable separately is to efficiently determine the beam intensity at different cross sections. In this way, we consider the penetration variable x as a time variable in similar time dependent problems.
We split the SSD variational formulation (2.28) as follows:

$$a(u,v) = (u_\beta, v)_{I_\perp} + \delta(\epsilon u_\beta, u_\beta)_{I_\perp} + (\epsilon u_z, v_z)_{I_\perp} + \delta(\epsilon u_z, (v_\beta)_z)_{I_\perp}, \tag{2.41}$$

$$b(u,v) = \delta(u, v_\beta)_{I_\perp} + (u,v)_{I_\perp}, \tag{2.42}$$

and rewrite the problem as

$$\begin{cases} \text{find a solution } u \in H^1_\beta(I_\perp) \text{ such that} \\ b(u_x, v) + a(u, v) = 0, \qquad \forall v \in H^1_\beta(I_\perp). \end{cases} \tag{2.43}$$

We use the finite dimensional subspace, $V_{h,\beta}$ of $H^1_\beta(I_\perp)$ and replace the discrete solution u_h with the "space-time-discrete" ansatz

$$u_h(x, y, z) = \sum_{i=1}^{M} \xi_i(x)\phi_i(y, z), \tag{2.44}$$

where $M \sim 1/h$. We replace v by ϕ_j for $j = 1, \cdots, M$, and insert (2.44) into the semidiscrete counterpart of (2.28). This gives the discretization method

$$\sum_{i=1}^{M} \xi_i'(x) b(\phi_i, \phi_j) + \sum_{i=1}^{M} \xi_i(x) a(\phi_i, \phi_j), \qquad j = 1, \cdots, M.$$

Or in matrix form,

$$B\xi_i'(x) + A\xi_i(x) = 0. \tag{2.45}$$

where $B = (b_{ij})$ with entries $b_{ij} = b(\phi_i, \phi_j)$ and $A = (a_{ij})$ with entries $a_{ij} = a(\phi_i, \phi_j)$. At this stage, we also discretize in the x direction using backward Euler to get the fully discrete scheme,

$$B(U_h^n - U_h^{n-1}) + k_n A U_h^n = 0. \tag{2.46}$$

Other fully discrete schemes can be obtained depending on the choice of the discretization method in x, e.g; Crank Nicolson, or discontinuous Galerkin.

2.4 Characteristic Schemes

The main feature in this section is the idea of *exact transport + projection.* To illustrate, we consider a homogeneous infinite slab, ($y, z \in \mathbf{R}$, $\tilde{Q} = (x, y, z)$ of thickness L, ($0 < x < L$). Let x be the penetration direction of a charged particle beam, $\{x_n\}$ an increasing sequence of discrete points indicating collision sites and $\{\mathcal{V}_n\}$ a corresponding sequence of piecewise polynomial spaces on space meshes $\{\mathcal{T}_n\}$ on the transversal variable $x_\perp = (y, z)$. Given the approximate solution (current) $J^{h,n} \in \mathcal{V}_n$ at the collision site x_n, solve the pencil beam equation exactly on the collision free interval (x_n, x_{n+1}) with the data $J^{h,n}$ to give the solution $J_-^{h,n+1}$ at the next collision site x_{n+1}, before the collision. This is an exact transport procedure. Now, one may compute $J^{h,n+1} = \mathcal{P}_{n+1} J_- h, n+1$, with $\mathcal{P}_{n+1}$ being a projection into $\mathcal{V}_{n+1}$. $J^{h,n+1}$ is the post collision solution at the (other face of collision) x_{n+1}. In this way, we have an algorithm of type *exact transport + projection.*

The domain $Q := I_x \times I_y \times I_z$ subdivided into slabs $S_n := I_x^n \times I_y \times I_z$, with $I_x^n := (x_{n-1}, x_n]$, $n = 1, 2 \dots, N$, corresponding to collision-free paths in the x-direction and I_y and I_z, bounded symmetric intervals representing the transversal domain of $x_\perp$. Each slab S_n has its own incident-transversal finite element mesh $\hat{\mathcal{T}_n}$. Consequently, at each collision site x_n we have two transversal meshes $\hat{\mathcal{T}_n^-} = \hat{\mathcal{T}_n} \mid x_n$ and $\hat{\mathcal{T}_n^+} = \widehat{\mathcal{T}_{n+1}} \mid x_n$, respectively. In general $\hat{\mathcal{T}_n^-} \neq \hat{\mathcal{T}_n^+}$ and the passage of information from one slab to the next is performed through a modified (built-in) L_2-projection.

Again, and as in the previous section, x is treated as a time variable, the transversal variable mesh is dealt with as a space mesh as in similar time-dependent problems

To proceed, we recall the model problem

$$\begin{cases} J_x + zJ_y = \epsilon J_{zz}, & (x, x_\perp) \in Q, \\ J_z(x, y, \pm z_0) = 0, & \text{for}(x, y) \in I_x \times I_y, \\ J(x, \pm y_0, z) = 0, & \Gamma_{\tilde{\beta}}^- \setminus \{suppf\}, \\ J(0, x_\perp) = f(x_\perp), & \end{cases} \tag{2.47}$$

where Q is a bounded domain $Q \equiv I_x \times I_y \times I_z = [0, L] \times [-y_0, y_0] \times [-z_0, z_0]$, $\Gamma_{\tilde{\beta}}^- := \{(x, x_\perp) \in \partial Q : \tilde{\beta} \cdot \boldsymbol{n} < 0\}$, $\tilde{\beta} = (1, z, 0)$, $x_\perp \equiv (y, z)$ is the transversal variable and $\boldsymbol{n} := \boldsymbol{n}(x, x_\perp)$ is the outward unit normal to Γ at $(x, x_\perp) \in \Gamma$.

As mentioned earlier, our model problem is a forward-backward, convection dominated convection-diffusion equation of degenerate type. A corresponding non-degenerate equation reads

$$\mathcal{L}(J) := J_x + \beta \cdot \nabla_\perp J - \epsilon \Delta_\perp J = 0, \tag{2.48}$$

where $\epsilon \approx C\sigma$. $\Delta_\perp := \partial^2/\partial y^2 + \partial^2/\partial z^2$ is the transversal Laplacian operator, and from now on $\beta \equiv (z, 0)$.
we introduce the change of coordinates $(x, \bar{x}_\perp) = (x, x_\perp - x\beta)$. If we set $\bar{J}(x, \bar{x}_\perp) = J(x, x_\perp)$, we reformulate (2.48) as

$$\bar{J}_x - \epsilon \Delta_\perp \bar{J} = 0, \qquad \text{in} \quad [0, L] \times I_y \times I_z, \qquad \bar{J}(0, \bar{x}_\perp) = f(x_\perp). \tag{2.49}$$

Since $\frac{\partial \bar{J}}{\partial x} = \frac{\partial}{\partial x} J(x, \bar{x}_\perp + x\beta) = \frac{\partial J}{\partial x} + \beta \cdot \nabla_\perp J$. If $\epsilon = 0$, then the solution of (2.49) is given by

$$\bar{J}(x, \bar{x}_\perp) = f(x_\perp - x\beta).$$

The characteristics of equation (2.48), in the case of $\epsilon = 0$ are given by $x_\perp + x\beta > 0$, and in this case the solution $J(x, \bar{x}_\perp)$ is constant along the characteristics.

Once and for both methods below, let $\{x_n\}$, $n = 0, 1, \ldots, N$, be an increasing sequence of x values with $x_0 = 0$, and let for each $0 \leq n \leq N$, $\{x_n\}$, $\{\mathcal{T}_n\}$ be a corresponding sequence of triangulations $\mathcal{T}_n$ of $\{x_n\} \times I_y \times I_z$ into triangles K and let $\mathcal{V}_n$ be the space of continuous piecewise linear functions on $\mathcal{T}_n$, i.e. $\mathcal{V}_n = \{v \in \mathcal{C}(I_y \times I_z) : v \text{ is linear on } K,\ K \in \mathcal{T}_n\}$. $\mathcal{C}(\Omega)$ denotes the set of continuous functions on Ω.

2.4.1 Characteristic Galerkin

In the case of $\epsilon = 0$, the characteristic Galerkin (CG) is formulated as follows: For $n = 1, 2 \ldots, N$

$$\begin{cases} \text{find } J^{h,n} \in \mathcal{V}_n \text{ such that:} \\ \int_{I_y \times I_z} J^{h,n}(x_\perp) v(x_\perp)\, dx_\perp = \int_{I_y \times I_z} J^{h,n-1}(x_\perp - \hbar_n \beta) v(x_\perp)\, dx_\perp, \end{cases} \tag{2.50}$$

where $\hbar_n = x_n - x_{n-1}$ and $J^{h,0} = f$. In other words

$$J^{h,n} = \mathcal{P}_n T_n J^{h,n-1}, \tag{2.51}$$

where $\mathcal{P}_n : L_2(I_y \times I_z) \rightarrow \mathcal{V}_n$ is the L_2 projection defined by $(\mathcal{P}_n w, v) = (w, v), \quad \forall v \in \mathcal{V}_n$, where $(\cdot,\cdot)$ denotes the inner product in $L_2(I_y \times I_z)$, and $T_n v_\perp = v(x_\perp - \hbar_n \beta) v(x_\perp)$.

2.4.2 Characteristic Streamline Diffusion

In this part, we formulate the Streamline Diffusion (SD) method, and the Characteristic Streamline Diffusion (CSD) for (2.48). CSD is a special case of SD obtained with oriented phase-space mesh elements. For $n = 1, 2, \ldots, N$, let $\hat{\mathcal{T}}_n = \{\hat{K}\}$ be a finite element subdivision of the slab $S_n = I_x^n \times I_y \times I_z$, $I_x^n = (x_{n-1}, x_n)$, into elements $\hat{K}$ and let $\hat{\mathcal{V}}_n$ be a space of continuous piecewise polynomials on $\hat{\mathcal{T}}_n$ of degree at most k. For $k = 1$ and small ϵ, the SD-method may be formulated as follows: For $n = 1, 2, \ldots, N$, find $\hat{J}^h \equiv \hat{J}^h|S_n \in \hat{\mathcal{V}}_n$ such that

$$\begin{aligned} \int_{S_n} (\hat{J}^h_x + \beta \cdot \nabla_\perp \hat{J}^h)(v + \delta(v_x + \beta \cdot \nabla_\perp v))\, dx dx_\perp \\ + \int_{S_n} \hat{\epsilon} \nabla_\perp \hat{J}^h \cdot \nabla_\perp v \, dx dx_\perp \\ + \int_{I_\perp} \hat{J}^{h,n}_+ v^n_+ \, dx_\perp = \int_{I_\perp} \hat{J}^{h,n}_- v^n_+ \, dx_\perp, \qquad \forall v \in \hat{\mathcal{V}}_n, \end{aligned} \tag{2.52}$$

where $v^n_\pm(x_\perp) = \lim_{\Delta x \to 0} v(x \pm \Delta x, x_\perp)$, $\hat{\epsilon} = max(\epsilon, \mathcal{F}(Ch^\alpha \mathcal{R}(\hat{J}^h))/M_n$, with

$$\mathcal{R}(\hat{J}^h) = |\hat{J}^h_x + \beta \cdot \nabla_\perp \hat{J}^h| + |[\hat{J}^h]|/\hbar.$$

$\mathcal{F}(v)$ is the element-wise average of v, $[v^n] = v^n_+ - v^n_-$, δ is a small parameter in general of order $\sim h$. $M_n = max_{x_\perp} |J^{h,n}_+(x_\perp)|$ is a normalization factor. The streamline diffusion modification is given by $\delta(v_x + \beta \cdot \nabla_\perp v)$ and the degenerate shock-capturing modification by $\hat{\epsilon}$. If β is approximated by piecewise constants on each slab, the streamline diffusion modification will disappear in the CSD-method.

The CSD-method is obtained through making a special choice of the finite element subdivision $\hat{\mathcal{T}}_n = \{\hat{K}\}$ of S_n and the corresponding finite element space $\mathcal{V}_n$. Let $\hat{\mathcal{T}}_n = \{\hat{K}\}$ be a subdivision of S_n given by the prismatic elements oriented along the characteristics

$$\hat{K}_n = \{(x, \bar{x}_\perp + (x - x_n)\beta) : \bar{x}_\perp \in K \in \mathcal{T}_n, x \in I^n_x\},$$

where $\mathcal{T}_n = \{K\}$ is a triangulation of $I_\perp$ given above, and let $\hat{\mathcal{V}}_n$ be defined by

$$\hat{\mathcal{V}}_n = \{\hat{v} \in \mathcal{C}(S_n) : \hat{v}(x, x\perp) = v(x\perp - (x - x_n)\beta), v \in \mathcal{V}_n\},$$

with $\mathcal{V}_n$ the space of continuous piecewise linear functions on $\mathcal{T}_n$ as above. So $\hat{\mathcal{V}}_n$ consists of the continuous functions $\hat{v}(x, x\perp)$ on S_n such that $\hat{v}$ is constant along characteristics $x_\perp = \bar{x}_\perp + x\beta$ parallel to the sides of the prismatic elements $\hat{K}_n$. With this choice, we notice that if $\hat{v} \in \hat{\mathcal{V}}_n$, then $\frac{\partial \hat{v}}{\partial x} + \beta \cdot \nabla_\perp \hat{v} = 0$. The SD-method is then reduced to the following: For $n = 1, 2, \ldots, N$, find $\hat{J}^h \in \hat{\mathcal{V}}_n$ such that

$$\int_{S_n} \hat{\epsilon} \nabla_\perp \hat{J}^h \cdot \nabla_\perp v \, dx dx_\perp + \int_{I_\perp} \hat{J}^{h,n}_+ v^n_+ \, dx_\perp = \int_{I_\perp} \hat{J}^{h,n}_- v^n_+ \, dx_\perp, \qquad \forall \hat{v} \in \hat{\mathcal{V}}_n, \tag{2.53}$$

where $\hat{\epsilon} = max(\epsilon, \mathcal{F}(Ch^{\alpha}(|[\hat{J}^h]|/\hbar_n))/M_n$, and $h(x, x_\perp) = h_n(x_\perp - (x - x_n)\beta)$; $h_n(x_\perp)$ gives the local element size of $\mathcal{T}_n$. If ϵ is small, then (2.53) can be stated as:

$$\int_{I_\perp} \hat{\epsilon}\nabla_\perp \hat{J}_+^{h,n} \cdot \nabla_\perp v \, dx dx_\perp + \int_{I_\perp} \hat{J}_+^{h,n} v \, dx_\perp = \int_{I_\perp} \hat{J}_-^{h,n} v \, dx_\perp, \qquad \forall \hat{v} \in \mathcal{V}_n, \tag{2.54}$$

writing $\hat{J}_+^{h,n} = J^{h,n}$, since $\hat{J}_-^{h,n} = T_n J^{h,n-1}$we can restate (2.54) as follows: For $n = 1, 2, \dots, N$, find $J^{h,n} \in \mathcal{V}_n$ such that

$$\int_{I_\perp} \hat{\epsilon}\nabla_\perp J^{h,n} \cdot \nabla_\perp v \, dx dx_\perp + \int_{I_\perp} J^{h,n} v \, dx_\perp = \int_{I_\perp} T_n J^{h,n-1} v \, dx_\perp, \qquad \forall \hat{v} \in \mathcal{V}_n, \tag{2.55}$$

where $J^{h,0} = f$ and $\hat{\epsilon} = \mathcal{F}(Ch_n^{\alpha}|J^{h,n} - T_n J^{h,n-1}|)/M_n$. If we introduce the operator $\hat{\mathcal{P}}_n : L_2(I_\perp) \bigcap L_\infty(I_\perp) \to \mathcal{V}_n$ defined as

$$(\hat{\mathcal{P}}_n w, v) = (\hat{\epsilon}\nabla_\perp \hat{\mathcal{P}}_n w, \nabla_\perp v) = (w, v), \qquad \forall v \in \hat{\mathcal{P}}_n.$$

This time, $\hat{\epsilon} = \mathcal{F}(Ch_n^{\alpha}|\hat{\mathcal{P}}_n w - w|/max|\hat{\mathcal{P}}_n w|)$, and $(\cdot, \cdot)$ denotes the $L_2(I_\perp)^m$ inner product with $m = 1, 2$. We can reformulate (2.55) as

$$J^{h,n} = \hat{\mathcal{P}}_n T_n J^{h,n-1}.$$

$\hat{\mathcal{P}}_n$ may be viewed as a modification of the usual L_2 projection, obtained by adding the artificial viscosity term with coefficient $\hat{\epsilon}$ as defined above.

Chapter 3

Computational Implementations

As outlined in Chapter 2, we split the discretization procedure into two steps. For the fully discrete schemes SG and SSD, we first discretize the domain $I_\perp = I_y \times I_z$ using continuous Galerkin approximation with piecewise linears: $cG(1)$, and then step advance in x using the Backward Euler (BE) method. In the characteristics schemes CG and CSD, we discretize $I_\perp$ using $cG(1)$ and we step advance in x using discontinuous Galerkin approximation with piecewise constants: $dG(0)$. The $cG(1)$ basis functions have the form $1-y-z$.

3.1 Exact Solution

The model problem (2.10) has a closed form exact solution, in some special cases(for $\epsilon = \epsilon(x) := \sigma(x)/2$, see [13]), given by

$$J(x,y,z) = \frac{\sqrt{3}}{\pi \epsilon x^2} e^{-2(3(y/x)^2 - 3(y/2)z + z^2)/(\epsilon x)}. \tag{3.1}$$

This allows us to compare the computed solution with the exact one and derive errors in various norms. However, the comparisons are limited because of two reasons. First, the closed form exact solution is a limited case which displays singularities near the origin. Second, the initial conditions used in the numerical experiments are not the same ones considered in deriving (3.1). As a matter of fact, we can not numerically provide an initial data of the form of a Dirac δ function. For comparison purposes, we consider three types of initial conditions that approximate the δ function in the L_1 sense: *Maxwellian*, *Hyperbolic*, and *modified Dirac*.

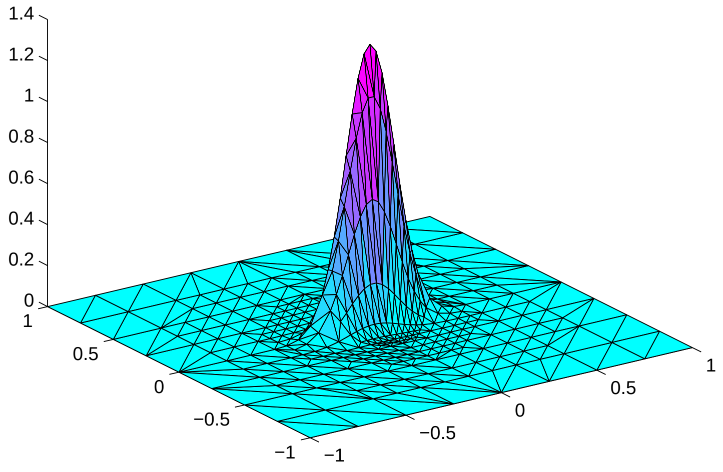

Figure 3.1: The closed form exact solution at x=20.

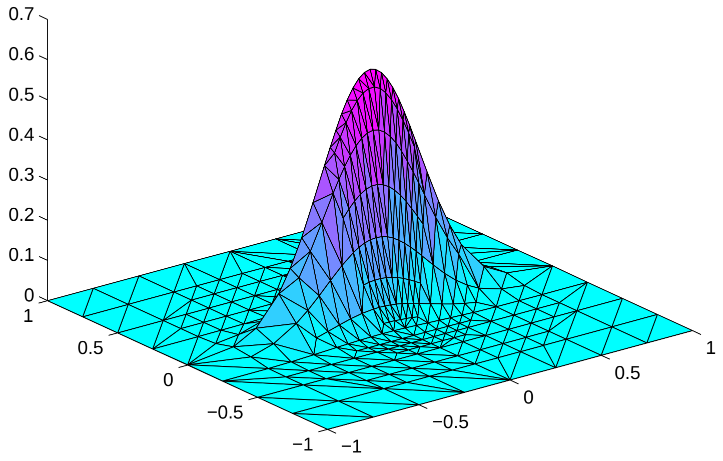

Figure 3.2: The closed form exact solution at x=30.

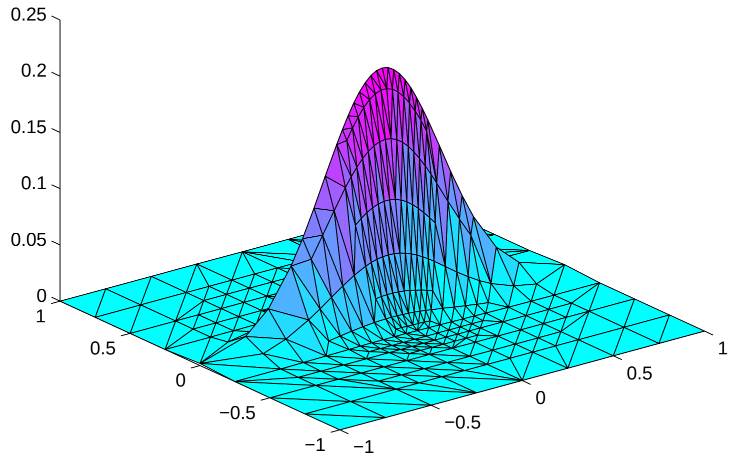

Figure 3.3: The closed form exact solution at x=50.

3.2 The Fermi pencil beam Solver

3.2.1 DOLFIN

The code included in Appendix A, and discussed below uses the free and open source library DOLFIN (Dynamic Object oriented Library for FINite element computation) a C++ interface of the free software for the Automation of Computational Mathematical Modeling FEniCS. DOLFIN is implemented in the department of computational mathematics at Chalmers University of Technology and Göteborg University.

DOLFIN is implemented in C++, and licensed under the GNU/GPL[1]. It is structured on Object-Orientedness notions and concepts. It can be used either as a stand-alone solver, or as a tool for development and implementation of new methods.

Dolfin is organized into three levels of abstraction, namely, the *user* level, the *module* level, and the *kernel* level. Through the *user* level, we specify the geometry, boundary conditions, and equations parameters.

[1]www.gnu.org

At *module* level, the developer has access to DOLFIN classes, objects, and building blocks. A typical DOLFIN module is composed of two classes: Problem.h, in which we specify the variational formulation, and a ProblemSolver.h. Both classes are used by ProblemSolver.cpp; the backbone of the solver which includes the algorithm for the problem.

The *kernel* space contains the code for all the basic tools that are available in the module level such as preconditioned iterative method, GMRES... etc. A more detailed discussion is available in [9]

3.2.2 The Fermi Pencil Beam Solver as a Module in DOLFIN

The solvers SG, SSD, CG, and CSD are named FermiSG, FermiSSD, FermiCG, and FermiCSD respectively. Each solver is composed of four files. Three in the *module level* and one in the *user level.* For instance Solver FermiX contains the files *FermiX.h*, *FermiXSolver.h*, and *FermiXSolver.cpp* (in the *module level*) and a main.cpp (in the *user level*).

- The class *FermiX.h* is derived from the abstract class *PDE.h*, it contains the variational formulation for the problem.
- The class *FermiXSolver.h* is derived from the abstract class *Solver.h.*
- The file *FermiXSolver.cpp* is the main part of the solver. In this file, we set the formulae for initial data, time stepping, assembling, etc.

The tasks performed in main.cpp are the following:

- Reading the geometry (mesh), which should be in a separate XML file.
- setting the functions for diffusion and convection coefficients ϵ and β, respectively.
- setting the boundary conditions.
- setting the time step, final time....

Machine used

The algorithms considered in this thesis were implemented and tested on a computer with the following characteristics: $PII, 130, 612KBRAM$ The operating system

```
Machine          :i686
Processor        :i686
HW-platform      :i386
Operating System :GNU/Linux
Version          :2.4.21-27.0.2.EL
Release          :1 Wed Jan 12 23:46:37 EST 2005
DOLFIN           :0.5.1
```

Using the benchmark in Dolfin

```
-Assembling 100 times                  2.76
-Matrix/vector multiplication 100 times 1.06
-LU factorization                      4.85
-Krylov solve                          2.43
Totaltime:                             12.52
```

Chapter 4

Results

4.1 Mesh

The meshes used in the numerical experiments are shown in Figure 4.1. The finest mesh used has a step size of $h = 0.0625$ in the y and z variables, and a step size $k = 0.005$ in the x variable, corresponding to 8198 nodes in two dimensions.

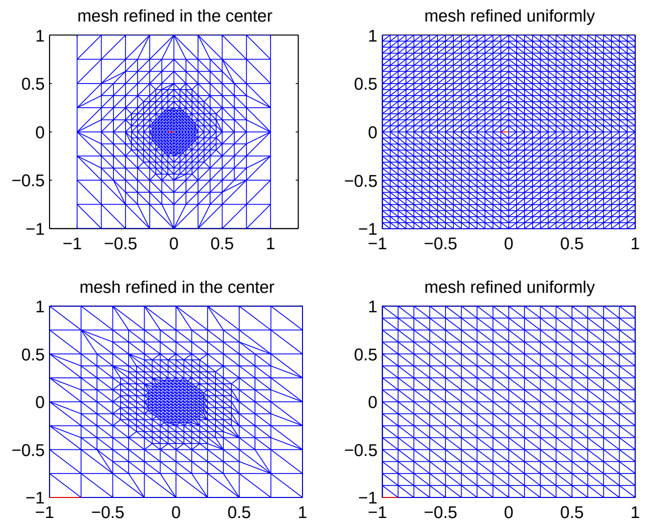

Figure 4.1: Meshes used: symmetric refinements (2 top figures). Non-symmetric refinements (2 down figures)

The refined meshes were obtained using DOLFIN.

4.2 Initial Data

We use three types of initial data to approximate the Dirac δ function; Maxwellian, hyperbolic and a modified Dirac (Figure 4.2)

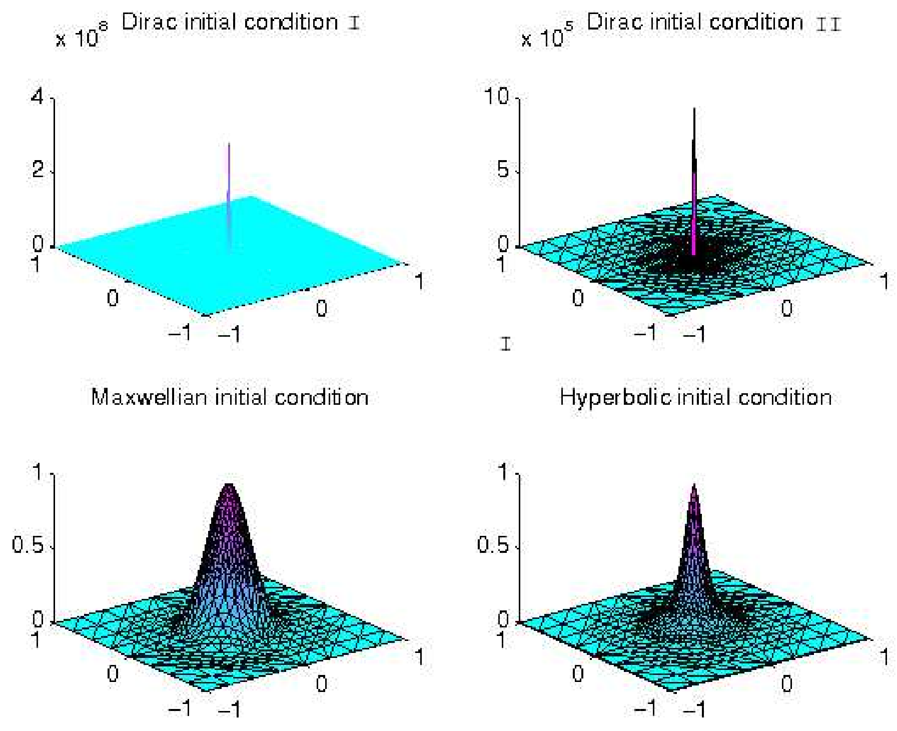

Figure 4.2: Types of initial conditions used.

4.3 Computed Solutions

The computational parameters used rely on the theoretical results discussed in chapter 2. For instance ϵ must be chosen small, and $\delta \sim h$. The computed solutions in the figures were computed using $\epsilon = 0.002$, and $h = 0.175$

($h = 0.0625$ in the finest mesh). The value for δ in SSD and CSD is taken as $\delta = h/2$. The value for x is set to $x = 1$, and the norms are calculated at this value.

At $x = 1$ the "exact" solution is shown in Figure 4.3.

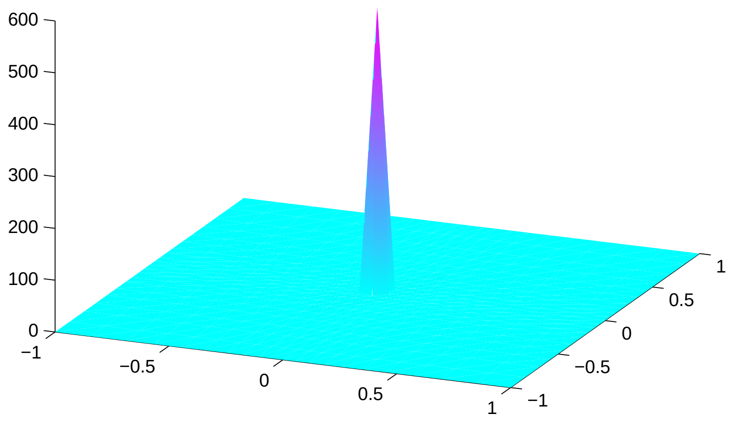

Figure 4.3: Closed form exact solution at x = 1.

The computed solutions for the Dirac initial conditions using CSD and SSD are shown in Figures 4.4 and 4.5 respectively.

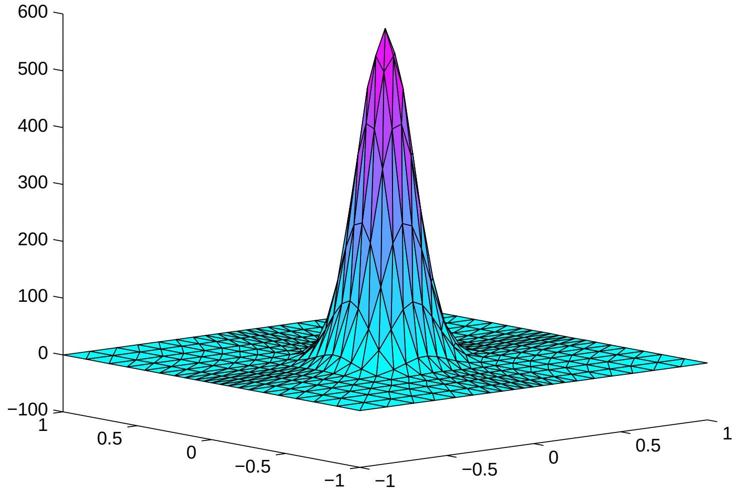

Figure 4.4: Computed solution at x = 1 with Dirac Init.cond. using CSD.

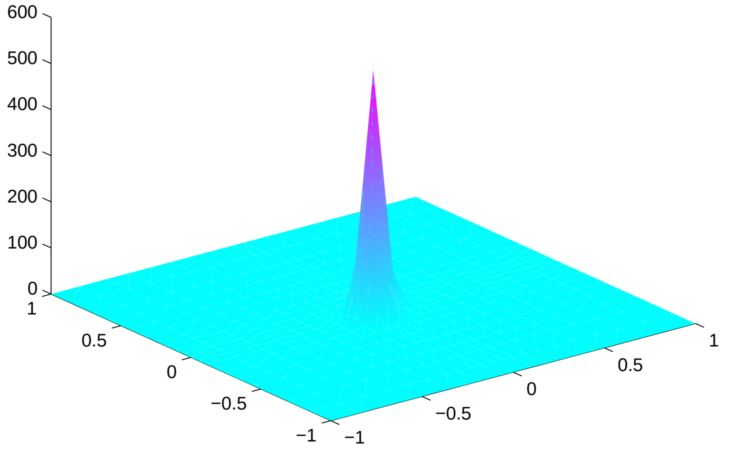

Figure 4.5: Computed solution at x = 1 with Dirac Init.cond. using SSD.

In the characteristic schemes, the time step (x is interpreted as time) was chosen as $k = 0.01$. While in the fully discrete schemes, it was set to $k = 0.005$.

Using SG and CG, and as we step advance in x, the computed solutions explode. In the case of the SG method, this happens through the formation of layers and in the CG method through an oscillatory behavior. On the other hand SSD and CSD are more stable. However, in some cases, the SSD and CSD lose the stability feature. We notice, in the SSD, that choosing large time steps results in the formation of sharp layers (as in the SG, see Figures 4.6 and 4.7). Whereas, some initial conditions (non smooth) in the CSD as well as the SSD scheme lead to an oscillatory behavior. This is more apparent in the case of a hyperbolic initial data (Figure 4.8).

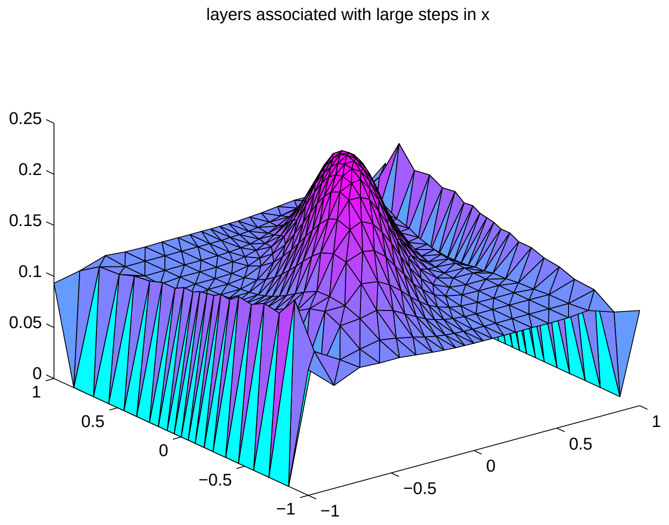

Figure 4.6: formation of layers in a solution with Maxwellian i.c at x = 0.66

This behavior can be eliminated by modifying the L_2 projection as mentioned in chapter 2.

For a smooth initial condition the usual L_2 projection would not produce such oscillations. In Figure 4.9, a solution with Maxwellian initial data is shown.

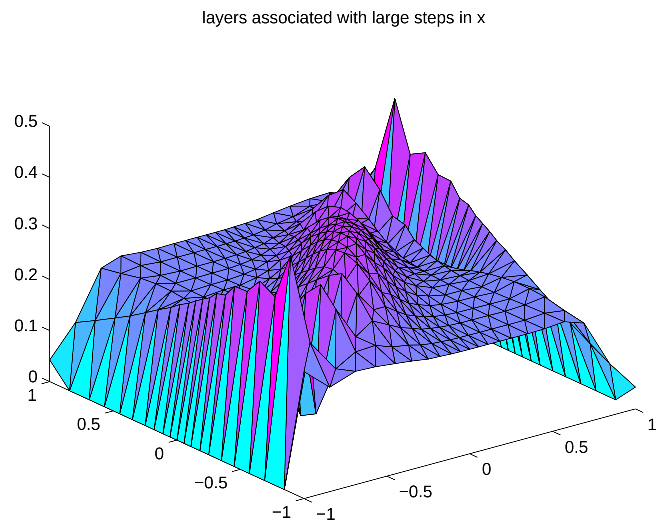

Figure 4.7: formation of layers in a solution with Maxwellian i.c at x = 0.85

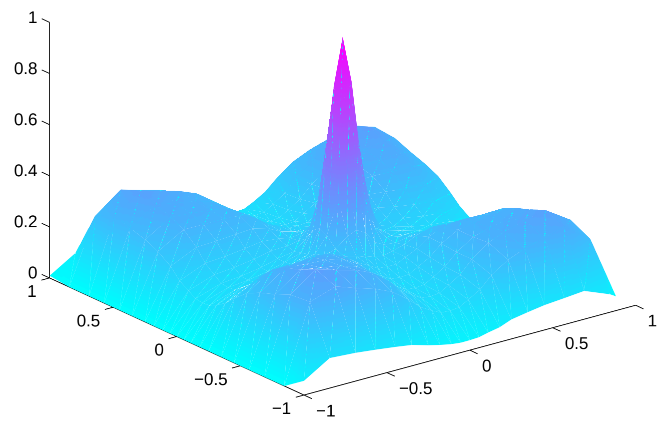

Figure 4.8: Oscillatory behavior of a solution with hyperbolic i.c

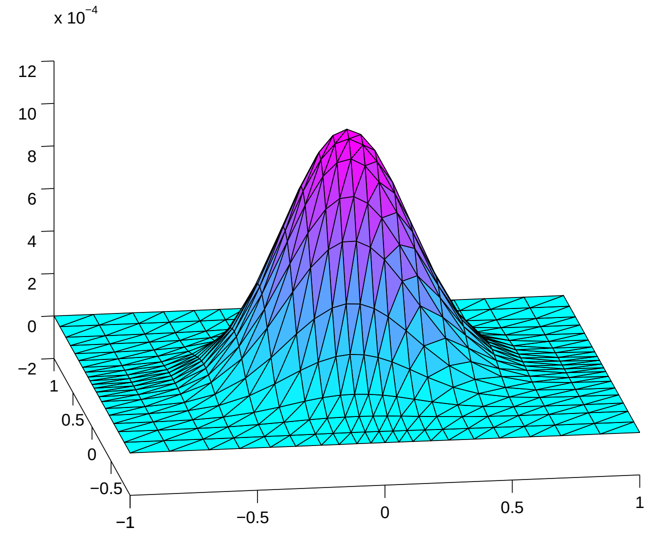

Figure 4.9: Computed solution with Maxwellian initial condition (three projections have been performed).

In the tables, 4.1, 4.2, and 4.3, we compute the L_∞, L_1, and L_2 norms of the errors between the "exact" and computed solutions for the three used initial conditions at $x = 1$.

In order to calculate the L_1 and L_2 errors we use the vertex quadrature of the form.

$$\int_K g(x)\,dx \approx \sum_{j=1}^{3} g(a_K^j)\frac{|K|}{3},$$

where a_K^j denotes the vertices of a triangle K, and we denote by $|K|$ the area of K. Therefore

$$\|w\|_{L_1} \approx \frac{1}{3}\sum_K |K| \sum_{j=1}^{3} w(a_K^j), \quad \text{and} \quad \|w\|_{L_2} \approx \left(\frac{1}{3}\sum_K |K| \sum_{j=1}^{3} (w(a_K^j))^2\right)^{1/2}.$$

L_∞	Dirac	Hyperbolic	Maxwellian
CSD	47.6325	0.6878	0.6521
SSD	48.2734	0.6135	0.6521
CG	47.2871	0.6046	0.623

Table 4.1: The L_∞ based norm for the errors between the "exact" and computed solutions

L_1	Dirac	Hyperbolic	Maxwellian
CSD	32.5463	0.1378	0.2912
SSD	33.4347	0.1535	0.2821
CG	36.1862	0.1546	0.3234

Table 4.2: The L_1 based norm for the errors between the "exact" and computed solutions

We also use the midpoint quadrature formula to get a weighed $\tilde{L}_2$ as follows:

$$\|w\|_{\tilde{L}_2} \approx \left(\frac{1}{3} \sum_K |K| \sum_{1<=i<j<=3} (w(a_K^{ij})^2) \right)^{1/2},$$

where a_K^{ij} denotes the midpoint of the side connecting the vertices a_K^i and a_K^j, and $|K|$ is the area of the triangle K.

Errors in the $\tilde{L}_2$ are shown in table 4.4

L_2	Dirac	Hyperbolic	Maxwellian
CSD	27.1325	0.1176	0.2031
SSD	27.1488	0.1135	0.2087
CG	31.9271	0.6036	0.2247

Table 4.3: The L_2 based norm for the errors between the "exact" and computed solutions

$\tilde{L}_2$	Dirac	Hyperbolic	Maxwellian
CSD	11.4339	0.1138	0.1342
SSD	11.2743	0.1155	0.1373
CG	12.7643	0.1264	0.1422

Table 4.4: The $\tilde{L}_2$ based norm for the errors between the "exact" and computed solutions

4.4 Conclusions

We have considered 4 deterministic algorithms for electron beams based on variants of General Galerkin or the $G2$-method , namely, the Standard Galerkin (SG), a Semi-Streamline Diffusion (SSD), a Characteristic Galerkin (CG), and a Characteristic Streamline Diffusion (CSD). We have developed the discretization schemes from a model problem, and showed stability for SG and SSG.

Computational Implementations were carried out to illustrate the applicability of the different algorithms for different types of initial data. To begin with, SSD and CSD are more stable and accurate than the SG and CG in all the three canonical forms of the initial conditions approximating the δ function. We have also noticed that differences in initial conditions affect the convergence estimates. In this sense, solutions with modified Dirac initial conditions are suited in both approaches CS and SSD. However, Maxwellian initial conditions produce accurate results in the CSD scheme, whereas the hyperbolic initial conditions produce more accurate results in the SSD scheme.

The resulting oscillatory behavior, while considering non smooth initial data, can be eliminated by modifying the L_2-projection. The formation of layers can be avoided by taking small steps in the penetration variable. However, a better approach to deal with this phenomenon is through adaptive refinement.

In general, for problems that are similar to our model problem (convection dominated convection-diffusion problems of degenerate type), streamline diffusion approaches such as SSD and CSD are more stable and accurate.

Bibliography

[1] M. Asadzadeh, *On the Stability of Characteristic Schemes for the Fermi Equation*, Computer Methods in Applied Mechanics and Engineering 191, (2002), 4641-4659.

[2] M. Asadzadeh, A. Sopasakis, *On Fully Discrete Schemes for the Fermi Pencil Beam Equation*, Appl. Comput. Math., 1 (2002). 158–174.

[3] M. Asadzadeh, *Streamline diffusion methods for the Fermi and Fokker-Planck Equations*, Transport Theory and Statistical Physics, (1997), 319-340.

[4] M. Asadzadeh, *A posteriori error estimates for the Fokker-Planck and Fermi pencil beam equations*, Mathematical Models and Methods in Applied Science, 10(2000), pp. 737–769.

[5] C. Börgers, *The Radiation Therapy Planning Problem*, in Computational Radiology and Imaging: Therapy and Diagnostics (Proceedings of IMA Workshop, March 1997), edited by C. Börgers and F. Natterer, IMA Volumes in Mathematics and its Applications 110, Springer-Verlag, 1999.

[6] C. Börgers, *Complexity of Monte Carlo and deterministic dose calculation methods*, Phys. Med. Biol. 43, (1998) pp. 517–528.

[7] C. Börgers, E. Larsen, *Asymptotic derivation of the Fermi pencil beam approximation*, Nuclear Science and Engineering (1996) vol.123 16 pp. 343–357.

[8] J. Coderre, *Lecture notes in Principles of Radiation interactions*, Department of Nuclear science and Engineering, MIT. Fall 2004.

[9] T. Dupont, J. Hoffman, C. Johnson, R. Kirby, M. Larson, A. Logg, R. ScottD.W.O. Rogers, A.F Bielaje, *The FEniCS project*, Preprint NO 2003:21, ISSN 1404-4382, Chalmers Finite Element Center. Chalmers University of Technology.

[10] K. Eriksson, D. Estep, C. Johnson, *Applied Mathematics: Body and Soul*, Vol. 3 (2004) Springer.

[11] E. Fermi, *Cosmic Ray theory*, Rev. Mod. Phy. 13 (1941) pp. 240–.

[12] J. Hoffman, C. Johnson, *Irreversibility in reversible systems I: the compressible Euler equations in 1d*, Preprint NO 2005:03, ISSN 1404-4382, Chalmers Finite Element Center. Chalmers University of Technology.

[13] D. Jette, *Electron dose calculations using multiple-scattering theory. A new theory of multiple scettering*, Journal of Medical Physicd. 23 (1996) pp. 459–476.

[14] C. Johnson, *Numerical solutions of partial differential equations by the finite element method*, Studentlitteratur (1987) Lund, Sweden.

[15] E. Larsen, *The nature of Transport Calculations in Radiation Oncology*, Transport Theory and Statistical Physics, 26 (1997) pp.739.

[16] M. Olivares, W. Parker, W. Strydom, *Electron Beams: Physical and Clinical Aspects* Review of Radiation Oncology Physics: A Handbook for Teachers and Students, edited E. B. Podgorsak Rev. Educational Reports Series, International Atomic Energy Agency. (2004) pp. 225–248.

[17] D.W.O. Rogers and A.F Bielaje, *Monte Carlo Techniques of Electron and Photon Transport for Radiation Dosimetry*, Dosimetry of Ionizing Radiation Ch 5 (2002) vol. III pp. 2–19.

Appendix A

Code

An example of a solver is presented in this appendix, namely, the Semi Streamline Diffusion solver. Comments are incorporated in the code.

A.1 FermiSSD

A.1.1 `main.cpp`

Below is the main programme.

```
// Copyright (C) 2005 Samir Naqos.
// Licensed under the GNU GPL Version 2.
#include <string>
#include <iostream>
#include <stdlib.h>
#include <dolfin.h>

using namespace dolfin;

//--Diffusion
real epsilon(real x, real y, real z, real t)
{
  return 0.002;
}
//--Convection
real beta(real x, real y, real z, real t, int i)
{
  if ( i == 0 )
    return y;
  else
    return 0.0;
}
//--Boundary conditions
void mybc(BoundaryCondition& bc)
{
  //--u = 0 in the inflow boundary
  if ( bc.coord().x == 0.0)
  {
   bc.set(BoundaryCondition::DIRICHLET,   0.0);
  }

  if ( bc.coord().y == 1.0)
    bc.set(BoundaryCondition::NEUMANN, 0.0);
  else if ( bc.coord().y == -1.0 )
    bc.set(BoundaryCondition::NEUMANN, 0.0);
  else
    bc.set(BoundaryCondition::DIRICHLET, 0.0);
}

int main(int argc, char **argv)
{
 //--Check arguments
  if ( argc != 2 )
 {
    dolfin_info("Usage: dolfin-FermiSSD n");
    dolfin_info("");
    dolfin_info("where n is one of");
    dolfin_info("");
    dolfin_info("  1 - the Maxwellian");
    dolfin_info("  2 - the Hyperbolic");
    dolfin_info("  3 - Modified Dirac I");
    dolfin_info("  2 - Modified Dirac II");
    return 1;
  }
  int n = atoi(argv[1]); // Get the number of the initial condition
  Mesh mesh("mesh2s.xml"); //--Read the mesh
  unsigned int refinements = 2; //--refine mesh
  for (unsigned int i = 0; i < refinements; i++)
  mesh.refineUniformly();
  //--Define and solve the problem
  Problem fermiSG("fermiSG", mesh);
  fermiSG.set("initial data", n);
  fermiSG.set("boundary condition", mybc);
  fermiSG.set("final time", 1.0);
  fermiSG.set("time step", 0.001);
  fermiSG.set("diffusivity", epsilon);
  fermiSG.set("convection", beta);
  fermiSG.solve();

  return 0;
}
```

In the file FermiSSD.h, we assemble the right hand side and the left hand side as specified in the variational formulation.

A.1.2 FermiSSD.h

```
#include <dolfin.h>
// Copyright (C) 2005 Samir Naqos.
// Licensed under the GNU GPL Version 2.
#ifndef __FERMISSD_H
#define __FERMISSD_H
#include <dolfin/PDE.h>

namespace dolfin {

  class FDS : public PDE {
  public:
        FDS(Function& uprevious,
        Function& diffusion,
        Function::Vector& convection) :
      PDE(2),beta(3)
      {
        add(up, uprevious);
        add(epsilon, diffusion);
        add(beta,  convection);
        d1 = h/2.0;
        d2 = h/2.0;
      }
   //--The left hand side of the variational formulation
   real lhs(const ShapeFunction& u,const ShapeFunction& v)
    {
      A = (beta,grad(u))*v*dx + d1*(beta,grad(u))*(beta,grad(v))*dx
          + epsilon*ddy(u)*ddy(v)*dx + d1*epsilon*ddy(u)*ddy((beta,grad(v)))*dx;
      B = d1*(beta,grad(v))*u*dx +u*v*dx;
      return (B + k*A);
    }
   //--The right hand side of the variational formulation
```

```
    real rhs(const ShapeFunction& v)
    {
      C=d2*(beta,grad(v))*up*dx + up*v*dx;
      return C;
    }
  private:
    ElementFunction::Vector beta;   // Convection
    ElementFunction epsilon;        // Diffusion
    ElementFunction up;  // value at left end-point
    real d1,d2,A,B,Abar,C;
  };
class Projectionfds: public PDE {
  public:
    Projectionfds(Function& actual) : PDE(2)
     {
      add(u1, actual);
     }
    real lhs(const ShapeFunction& u,const ShapeFunction& v)
     {
      return 0;
     }
    real rhs(const ShapeFunction& v)
     {
      return sqrt(u1*u1*v*dx);//--weighted L2
      return sqrt(u1*u1*dx); //--L2
     }
  private:
    ElementFunction u1;
  };
}
#endif
```

A.1.3 FermiSSDSolver.h

```
// Copyright (C) 2005 Samir Naqos.
// Licensed under the GNU GPL Version 2.
#ifndef __FERMISSD_SOLVER_H
#define __FERMISSD_SOLVER_H
#include <dolfin/Solver.h>

namespace dolfin {

  class FermiSSDSolver : public Solver {
  public:
    FermiSSDSolver(Mesh& mesh);
    const char* description();
    void solve();
    real l2error(Mesh &mesh, Vector &dofs);
  };
}
#endif
```

A.1.4 FermiSSDSolver.cpp

```
// Copyright (C) 2005 Samir Naqos.
// Licensed under the GNU GPL Version 2.
#include <fstream>
#include <string>
#include <iostream>
#include <stdlib.h>
#include "FermiSSDSolver.h"
#include "FermiSSD.h"

using namespace dolfin;
//-----------------------------------------------------------------------------
FermiSSDSolver::PoissonSolver(Mesh& mesh) : Solver(mesh)
{
  dolfin_parameter(Parameter::REAL,      "final time",  1.0);
  dolfin_parameter(Parameter::REAL,      "time step",   0.001);
  dolfin_parameter(Parameter::FUNCTION,  "diffusivity", 0.002);
  dolfin_parameter(Parameter::VFUNCTION, "convection",  0.1);
  dolfin_parameter(Parameter::INT,      "initial data",  0);
}
//-----------------------------------------------------------------------------
const char* FermiSSDSolver::description()
{
  return "FermiSSD";
}
//------------------------------------------------------------------------------
void FermiSSDSolver::solve()
{
  Matrix A;
  Vector x0, x1, b, bb, d, Xexact, diff;
  Function Uexact(mesh, Xexact);
  Function u0(mesh, x0);
  Function u1(mesh, x1);
  Function epsilon("diffusivity");
  Function::Vector beta("convection", 3);
  FDS       fds(u0,epsilon,beta);
  Projectionfds       projectionfds(u1);
  KrylovSolver solver;
  File         computed_file("fermi.m");
  File         exact_file("exact.m");
  std::ofstream file("norm.m");
  real t = 0.0, eee= 0.0;
  real T = dolfin_get("final time");
  real k = dolfin_get("time step");
  int c = dolfin_get("initial data");
//-----------------------------------
int counter = 0;
  switch (c) {
  case 1:
    dolfin_info("Solving with Maxwellian initial condition.");
      for(NodeIterator n(mesh); !n.end(); ++n)
       {
         x0(n->id()) =  exp(16*(-((n->coord().x)-k*(n->coord().y))
                         *((n->coord().x)-k*(n->coord().y))-(n->coord().y)
                         * (n->coord().y)));
       }

  break;
  case 2:
    dolfin_info("Solving with hyperbolic initial condition");
      for(NodeIterator n(mesh); !n.end(); ++n)
       {
         x0(n->id()) =  1/(100*((n->coord().x)*(n->coord().x)
                     + (k*k+1)*(n->coord().y)*(n->coord().y)
                     - 2*k*(n->coord().x)*(n->coord().y))+1) ;
       }
  break;
  case 3:
    dolfin_info("Solving with modified Dirac I");
      for(NodeIterator n(mesh); !n.end(); ++n)
       {
         x0(n->id()) = 1/(sqr(n->coord().x)*0.5+sqr(n->coord().y)*0.5+0.000001);
       }
  break;
  case 4:
   dolfin_info("Solving with modified Dirac II");
      for(NodeIterator n(mesh); !n.end(); ++n)
       {
         x0(n->id()) = 2.0e-9(1/sqrt((sqr(n->coord().x)+sqr(n->coord().y)+DOLFIN_EPS)));
       }
   break;
   default:
    dolfin_error("No such Initial conditions");
  }

 //--Save initial value
  u0.rename("u", "temperature");
  mfile << u0;
 //--Compute exact solution
   for(NodeIterator n(mesh); !n.end(); ++n)
     {
        Xexact(n->id()) = (2.0*sqrt(3)/(3.14*0.002*20.0*20.0))
                      *exp(-4.0*(3*(sqr(n->coord().x)/20.0)-3
                      *((n->coord().x)/20.0)*(n->coord().y)
                      +sqr(n->coord().y))/(0.002*20.0)) ;
     }
exact_file << Uexact; //--Save exact solution to the file
//--Assemble matrix
 fds.k=k;
 FEM::assemble(fds, mesh, A);
int i = 0;
 //--Start a progress session
  Progress p("Time-stepping");
  //--Start time-stepping
while ( t < T ){
    i++;
    //--Make time step
    t += k;
    //--Assemble load vector
    fds.k = k;
    fds.t = t;
    FEM::assemble(fds, mesh, b);
    solver.solve(A, x1, b);  //--Solve the linear system
         diff = x1;
         diff -= Xexact;
         file << diff.norm(0) << std::endl;
   if((counter % 11 == 0 ) && (counter <= 33))//--the projection step
   {
     projectionfds.k = k;
     projectionfds.t = t;
     FEM::assemble(projectionfds, mesh, bb);
     x1=bb;
   }
    x0=x1;
    u1.update(t); //--Save the solution
    computed_file << u1;
    p = t / T;    //--Update progress
    dolfin::cout <<"finished iteration"<< i <<dolfin::endl;
  }
}
```