On Fully Discrete Finite Element Schemes for the Fermi and Fokker-Planck Pencil-beam Equations

Muhammad Naseer

January 23, 2015

Contents

Acknowledgements

I can not find any proper words to show my deep thankfulness and respect for my research supervisor, Professor Mohammad Asadzadeh. He has helped me to become an independent researcher, his kind and patience attitude leads me to work in a total freedom of mind and heart. I would like to manifest my warmest gratitude to all other people who in one-way or the other have helped me in the completion of this thesis. They guided me throughout my research and supported me with their advices to rectify the troubles every time I faced.

Abstract

On fully discrete finite element schemes for the Fermi and Fokker-Planck pencil-beam equations.
This study concerns the Fokker-Planck equation and its asymptotic limit the Fermi pencil-beam equation in two-space dimensions (x, y), where x is aligned with the beams penetration direction and y together with the scaled angular variable z = tan μ correspond to a bounded and symmetric transversal cross-section.

We consider the forward-backward degenerate, convection dominated, convection-diffusion problem. For this problem we want to study some fully discrete numerical schemes using the standard Petrov-Galerkin finite element methods (for discretization of the transversal domain, combined with the backward Euler, Crank Nicolson) and discontinuous Galerkin methods (for discretization in the penetration variable).

In our numerical studies we show convergence results for the standard finite element method as a certain weighted L_2 norm which where obtained in [1]. Numerical implementations are presented for some examples with the data approximating Dirac delta-function to confirm the expected performance of the standard finite element scheme.

1 Introduction for model problem

We consider beam of electron or proton particles entering in an object from the origin $(x, y, z) = (0, 0, 0)$ in the direction of positive x axis. We study monoenergetic Fermi model problem with a closed form exact solution which is a model for the calculation of the dose (energy deposited per unit mass):

$$\begin{cases} W_x + zW_y = \epsilon W_{zz} & (x, y, z) \in \Omega = I_x \times I_\perp \\ W_z(x, y, \pm z_0) = 0 & for\ (x, y) \in I_x \times I_y, \\ W(0, x_\perp) = f(x_\perp) & for\ x_\perp \in I_\perp \\ W(x, x_\perp) = 0 & on\ \Gamma_{\beta}^{-}(0, x_\perp). \end{cases} \tag{1}$$

The problem is formulated for the flux W. Equation (1), known as pencil-beam equation, is an asymptotic expansion of the Fokker-Planck equation, which in turn is an asymptotic limit of the linear Boltzmann equation. Both equations are used in transport process and relay on forward peaked scattering [2]. In this thesis we have studied some approximations and fully discrete schemes, for numerical solutions of Fermi and Fokker-Planck equations in both two and three space dimensions, which was developed for a model for the transport of photon or electron particle beams used to treat cancer.

It is very important to calculate the proper amount of dose used in radiation therapy. High amount of dose can destroy healthy tissue and create undesired side effects. One may calculate the exact amount of dose of photon and electrons using a Monte-Carlo algorithm [3].

Pencil-beam models mostly use methods based on Fermi-Eyges equation in radiative transfer theory, see [4] and [5] for details. These models first used for electron radiation [6] and later on for proton particle beams, see [7] and [2].

We focus in our work on the numerical implementations (in using general finite element method) for the equation (1).

2 Radiation cancer therapy

2.1 Introduction to radiotherapy

Generally there are two types of radiotherapy: the external and the internal radiotherapy.

External radiotherapy:

In this way of treatment a certain machine gives high energy radiation such as x-rays to the cancer part of body and a small area of normal tissue surrounding it.

Internal radiotherapy:

In this type of radiation therapy the patients will obtain liquid source of radiation in their body through injection into a vein.

This thesis concerns the study related to the external radiotherapy, which from now on we will refer to as radiotherapy. Radiation therapy is used as one of the most common treatments for different types of cancer. Usually x-rays, γ-rays, electron beams, or protons are used in radiotherapy. Cancer cells are treated by sending high doses of radiation to them. Growth of cancer cells in body is faster than normal cells.
Radiotherapy helps to stop cancer cells from growing and dividing by affecting inside of cells. It makes small breaks in DNA of cancer cells. It could affect normal cells around them as well. But usually normal cells recover faster and are not damaged as much as cancer cells. Beam accelerator machine creates radiotherapy beams using electricity. Two features are important in radiotherapy of cancer treatment. One is the local treatment which means that it could be used for particular part of the body. The other aspect is being less harmfulness to the healthy tissue.

2.2 Brief history of radiotherapy

The Radiation-therapy was discovered within 3 years after the time a German physics professor, Wilhelm Rontgen, gave a lecture on X-rays in 1895 called "Concerning a New Type Of Ray". At the first time it was experienced as a cancer treatment by a doctoral student, Emil Grubbé, in Chicago. After three years two Swedish doctors used radiotherapy to treat different cancer types in neck and head which brought Nobel Prize for Röntgen in 1901. Besides the fast growth of radiotherapy field, in early 1900s cancer cure scientists came to the conclusion that radiotherapy could be dangerous and harmful because it could cause cancer itself. It was making some patients worse due to lack of knowledge in proper dose amount to skin so the medical society began considering it as a dangerous method of treatment. In that time X-ray was used by physicians around the world for diagnostics.

2.3 Mathematical view of radiotherapy

In mathematical modeling, an important aspect that should be discussed is calculation of radiation dose which gives information about energy absorbed per unit mass in tissue layers. As result of this calculation is the dose function helps radiation oncologists to know amount of radiation to give to tumor in order to destroy it.

3 Algorithms for Model Problem

3.1 Finite element spaces

We consider a radiation beam which is incidented at the center of the symmetric domain $D_\perp = D_y \times D_z = [-y_0, y_0] \times [-z_0, z_0]$ for $(y_0, z_0) \in \mathbb{R}^2_+$ and penetrating in the direction of the positive x-axis. Then the computational domain Ω of our study is a three dimensional slab with $(x, y, z) \in \Omega = I_x \times D_\perp$ where with $x \in I_x$ such that $I_x = [0, L]$. For the rectangular domain $D_\perp$ we take a mesh size h in y and z direction. We discretize $D_\perp$ with finite element approximation on a quasi-uniform triangulation. We let $\beta = (1, z, 0)$ and define the inflow and outflow boundaries as,

$$\Gamma^-_\beta := x_\perp \in \Gamma := \partial D_\perp : \mathbf{n}(x_\perp) \cdot \beta < 0\,, \tag{2}$$

and

$$\Gamma^+_\beta := x_\perp \in \Gamma := \partial D_\perp : \mathbf{n}(x_\perp) \cdot \beta > 0\,, \tag{3}$$

respectively. Here $\Gamma = \partial\Omega$ is boundary of the domain Ω , and $\mathbf{n}(x_\perp)$ is the outward unit normal to the boundary Γ at $x_\perp \in \Gamma$. We now introduce a discrete finite dimensional function space $V_{h,\beta} \subset H^1_\beta(D_\perp)$ with

$$H^1_\beta(D_\perp) = \left\{ v \in H^1(D_\perp) : v = 0 \; on \; \Gamma^-_\beta \; \{(0, x_\perp)\} \right\}, \tag{4}$$

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

$$\inf_{\chi \in V_h, \beta} \|v - \chi\|_j \leq C h^{\alpha - i} \|v\|_\alpha, j = 0, 1, \text{ and } 1 \leq \alpha \leq r. \tag{5}$$

Here r being an integer > 1 where j is the number of derivatives and $H^s(D_\perp)$ is the L_2 based Sobolev space with all its derivatives of order $\leq$ s being in L_2 [8]. An example of such a spaces $V_{h,\beta}$ is a set of sufficiently smooth piecewise polynomials $P(x_\perp)$ of degree $\leq r$, satisfying the boundary condition in our model problem

$$\begin{cases} u_x + z u_y = \epsilon u_{zz}, & (x, y, z) \in \Omega = D_x \times D_\perp \\ u_z(x, y, \pm z_0) = 0 & for\ (x, y) \in D_x \times D_y, \\ u(0, x_\perp) = f(x_\perp), & for\ x_\perp \\ u(x, x_\perp) = 0 & on\ \Gamma^-_\beta(0, x_\perp). \end{cases} \tag{6}$$

Proceeding further we introduce a bilinear form $A : H^1_\beta \times H^1_\beta$ as

$$A(u, v) = (u_x, v)_\perp + (z u_y, v)_\perp, \;\; \forall\, u, v \in H^1_\beta(I_\perp). \tag{7}$$

Now the continuous variational formulation for the problem is defined as: find a solution u of (6) such that,

$$\begin{cases} A(u, \chi)_\perp + (\epsilon u_z, \chi_z)_\chi = 0 \quad \forall \chi \in H^1_\beta(D_\perp), \\ u(0, \chi_\perp) = f(x_\perp). \end{cases} \tag{8}$$

Let $\tilde{u} \in V_{h,\beta}$ be an interpolant of the solution u of (6) defined by

$$A(u - \tilde{u}, \chi) = 0 \ \ \forall\, \chi \in V_{h,\beta}. \tag{9}$$

Now the discrete variational formulation problem is to find $u_h \in V_{h,\beta}$ such that,

$$\begin{cases} (u_{h,x}, \chi)_\perp + (z u_{h,y}, \chi)_\perp + (\epsilon u_{h,z}, \chi_z)_\perp = 0 \quad \forall \chi \in V_{h,\beta}, \\ u_h(0, x_\perp) = f_h(x_\perp), \end{cases} \tag{10}$$

where $f_h(x_\perp)$ is the finite element approximation of f. Here the scalar product $(u, v)_\perp$ is defined as $(u, v)_\perp = \int_{D_\perp} u(x_\perp) v(x_\perp) dx_\perp$ and the norm is defined as $\|u\|_{L_2(I_\perp)} = (u, u)_\perp^{1/2}$. Further the mesh size h should be chosen such that

$$h^2 \leq \epsilon \leq h. \tag{11}$$

3.2 The standard finite element discretization

In fully discrete scheme we derive algorithms for standard Galerkin and semi-streamline diffusion methods for discretization of $I_\perp$ combined with Discontinuous Galerkin and backward Euler methods discretizing I_x. We start introducing the bilinear form for model problem (6) with w = u

$$a(w, v) = (w_\beta, v)_\perp + \delta(w_\beta, v_\beta)_\perp + (\epsilon w_z, v_z)_\perp + \delta(\epsilon w_z, (v_\beta)_z)_\perp \tag{12}$$

$$b(w, v) = \delta(w, v_\beta)_\perp + (w, v)_\perp \tag{13}$$

and write the continuous problem: find the solution $w \in H^1_\beta(I_\perp)$ such that,

$$b(w_x, v) + a(w, v) = 0, \quad \forall\, v \in H^1_\beta(I_\perp). \tag{14}$$

Let $V_{h,\beta}$ represents the discrete solution by separation of variables

$$w_h(x, y, z) = \sum_{i=1}^{M} \xi_i(x) \phi_i(y, z). \tag{15}$$

Here $M \sim 1/h$. Letting $v = \phi_j$ for $j = 1, 2, \ldots, M$ and inserting w_h from (15) instead of w in (14) we get the discretization method,

$$\sum_{i=1}^{M} \xi_i'(x) b(\phi_i, \phi_j) + \sum_{i=1}^{M} \xi_i(x) a(\phi_i, \phi_j) = 0 \qquad j = 1, 2, \ldots, M \tag{16}$$

equation (16) in matrix form can be written as

$$B\xi'(x) + A\xi(x) = 0. \tag{17}$$

Here $B = (b_{ij})$ with $b_{ij} = b(\phi_i, \phi_j)$ and $A = (a_{ij})$ with $a_{ij} = a(\phi_i, \phi_j)$. Using the backward Euler we discretize ξ in the x direction to get the fully discrete scheme,

$$B(\xi_h^n - \xi_h^{n-1}) + k_n A \cdot \xi_h^n = 0. \tag{18}$$

For the problem (6) which can be recognized as backward Euler

$$[B + k_n A]\xi_h^n = B \cdot \xi_h^{n-1} \tag{19}$$

Other fully discrete schemes can be obtained depending on the choice of the discretization method in the direction of x.

3.2.1 Characteristic Galerkin

We start by letting $\{x_n\}, n = 0, 1, \ldots, N$, be an increasing sequence starting at initial point $x_0 = 0$, and for each n we have corresponding sequence of triangulations $\{\tau_n\}$ of $S_n := \{x_n\} \times I_y \times I_z$ into triangular elements. We let V_n being the space of continues piecewise linear functions on S_n.

By letting the convection cofficient $\epsilon = 0$ in the model problem (6) to solve characteristic Galerkin method can be formulated as:

$$\begin{cases} \text{find} \quad W^{h,n} \in V_n \; such \; that \quad for \quad n = 0, 1, \ldots, N \\ \int_{I_y \times I_z} W^{h,n}(x_\perp) v(x_\perp) dx_\perp = \int_{I_y \times I_z} W^{h,n-1}(x_\perp - I_x^n \beta) v(x_\perp) dx_\perp, \end{cases} \tag{20}$$

In (20) $I_x^n = x_n - x_{n-1}$; $W^{h,0} = f$, and $W^{h,n} = J_n k_n W^{h,n-1}$ such that $k_n W^{h,n-1}(x_\perp) = W^{h,n-1}(x_\perp - I_x^n \beta)$. Here $J_n : L_2(I_y \times I_z) \to V_n$ is L_2 projection, with $(J_n w, v) = (w, v), \forall\, v \in V_n$,

3.2.2 Characteristic Streamline Diffusion Method

We begin by constructing an oriented phase-space mesh to obtain streamline diffusion method: let $\{\hat{M}_n\} = \{\hat{K}\}$ for $n = 1, 2, ..., N$, be a subdivision of slab $S_n = I_x^n \times I_y \times I_z$ using a finite element mesh, in $I_x^n = (x_{n-1}, x_n)$. Further for $\hat{K}$ elements let $\hat{V}_n$ be a space of continuous piecewise polynomials of degree at most k. For $k = 1$ and for small ϵ the streamline diffusion method for the problem (6) can be formulated as:

$$\begin{cases} \text{find } \hat{W}_n^h \equiv \hat{W}^h | S_n \in \hat{\nu}_n \; such \; that \; for \; n = 1, 2, \ldots, N \\ \int_{S_n} (\hat{\beta} \cdot \nabla \hat{W}^h)(v + \delta(v_x + \beta \cdot \nabla_\perp v)) dx dx_\perp + \int_{S_n} \epsilon \nabla_\perp \hat{W}^h \cdot \nabla_\perp v dx dx_\perp \\ + \int_{I_\perp} \hat{W}_+^{h,n} v_+^n dx_\perp = \int_{I_\perp} \hat{W}_-^{h,n} v_+^n dx_\perp, \qquad \forall\, v \in \hat{V}_n \end{cases} \tag{21}$$

with $v_\pm^n(x_\perp) = lim_{\Delta x \to 0} v(x_n \pm \Delta x, x_\perp)$, where $\delta(v_x + \beta.\nabla_\perp v)$ is the modification in streamline diffusion, and $\delta \sim h$. In the presence of shock the term with ϵ is involved in order to capture the shock [1]. The Characteristic Streamline Diffusion method can be derived by making a special choice of finite element

subdivision where $\hat{M}_n = \{\hat{K}\}$ is given by tetrahedral element oriented along the characteristic streamline diffusion method.

$\hat{K} = \{(x, \bar{x}_\perp + (x - x_n)\beta) : x_\perp \ \in K \ \in M_n, x \in I_x^n\}$ space

$S_n = \{K\}$ is a triangulation as above and $\hat{V}_n$ is given as

$\hat{V}_n = \{\hat{v} \in C(S_n) : \hat{v}(x, x_\perp) = v(x_\perp + (x - x_n)\beta) \ v \in V_n\}$.
Here $\hat{V}_n$ consists on continuous functions $\hat{v}(x, x_\perp)$ on S_n with $\hat{v}$ be a constant along characteristics $x_\perp = \hat{x}_\perp + x\beta$ parallel to the sides of the tetrahedral elements $\hat{K}_n$. With this special choice we notice that if $\hat{v} \in \hat{V}_n$ then $\hat{v}_x + \beta.\nabla_\perp \hat{v} = 0$. Then the streamline diffusion method to solve problem (6) can be written in the reduced form

$$\begin{cases} find\ \hat{W}^h \in \hat{V}_n\ such\ that\ for\ n = 1, 2, \ldots, N, \\ \int_{L_n} \epsilon \nabla_\perp \hat{W}^h \cdot \nabla_\perp v \ dx dx_\perp + \int_{I_\perp} \hat{W}_+^{h,n} v_+^n \ dx_\perp = \int_{I_\perp} \hat{W}_-^{h,n} v_+^n \ dx_\perp \ , \forall \ \hat{v_n} \in \hat{V}_n. \end{cases} \tag{22}$$

4 Results

In this section we will discuss the numerical results using different numerical techniques described in section (3). We begin by discretizing the domain $I_\perp = I_y \times I_z$ using continuous piecewise linear continuous. Then we show the results for Standard Galerkin and Semi-Streamline Diffusion, and continuous Galerkin approximation. Then we use backward Euler Method (18) for the case of standard Galerkin approximation for I_x. Jump discontinuities are allowed in both Semi-Streamline diffusion(SSD) and SD characteristic schemes. $I_\perp$ is discretized by using continuous Galerkin method for the model problem (6). In x direction we discretized by using discontinuous Galekin approximation with piecewise constants denoted as dG(0).

4.1 Exact solution

The closed form of the solution of our model problem (6) is

$$w(x, y, z) = (\sqrt{3}/(\pi\varepsilon x^2)) \exp\left(-2(3(y/x)^2 - 3(y/2)z + z^2)/(\varepsilon x)\right). \tag{23}$$

The equation (23) gives us the exact solution to compare our computed solution and see how close we are to the exact solution. We computed the error $e = \|w_h - w\|$ using different norms $\|\cdot\|_{L_2(\Omega)}$, $\|\cdot\|_{L_1(\Omega)}$ and L_∞, and present results in section (4.4). This particular physical problem has applications in many fields of material science, astrophysics, electron microscopy and also radiation cancer therapy.

We start by presenting the meshing which we used here, then initial data, computed solution and norms of error which we have calculated for this thesis work.

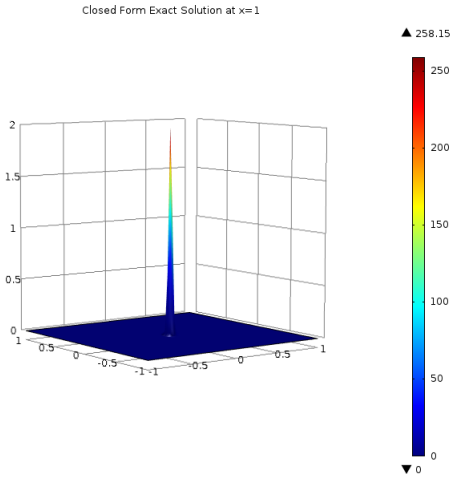


Figure 1: The exact solution of (23)Fermi pencil beam obtained for x =1.

Figure (1) shows that the closed form exact solution of (23) when $x = 1$ gives us the more strength of electron beam, but cover less area. Figure (3) show that when $x = 2$ this gives us the less strength of beam but with more area to be treated, we will see this in the next two figures.

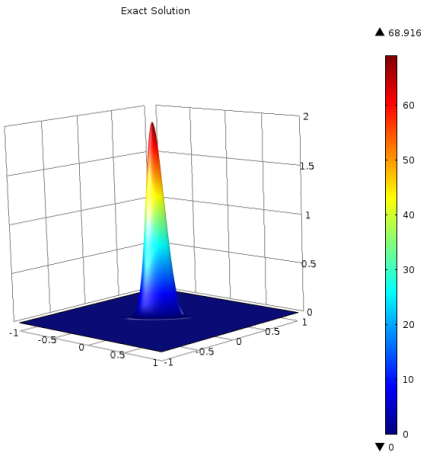


Figure 2: The exact solution of (23) Fermi pencil beam obtained at x =2.

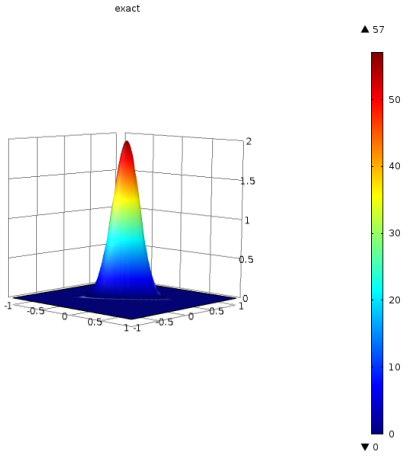


Figure 3: The exact solution of (23) Fermi pencil beam obtained at x =3.

The difference in solutions obtained at x = 1 and x = 2 in figures (1) and (3) respectively is that the area covered with the beam at $x = 2$, is comparatively much larger than the area covered at $x = 1$.
All these exact solutions were derived from the two dimensional models, but we get similar results in three dimensional models as well which we will see in section (4.4).

4.2 Geometry

Table 1: Geometry used in two dimensional model

Property	Value
Space dimension	2
Number of domains	1
Number of boundaries	4

Table 2: Geometry statistics

Name	Value
position	$0, 0$
Base	Center
Side length	2
Side length	2

Table 3: Position

Type	Units
Length unit	m
Angular unit	Radian

Table 4: Units

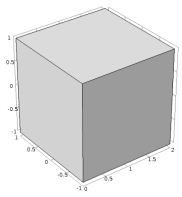


Figure 4: Example of a three dimension geometry which we used for getting our numerical results in 3d.

In 3D, we consider a cube of size $\{2, 2, 2\}$. The position of cube is $\{1, 0, 0\}$. The total number of boundaries in this case is 6 and domain is 1. Number of edges are 12 and the number of vertices 8. The length unit is meter and the angular unit is radian.

4.3 Meshing

Whenever we use the finite element method, it is important to keep in mind that the accuracy of solution is linked to the mesh size. As mesh size decreases towards zero (leading to a model of infinite size), we move toward the exact solution for the equations we are solving. However, since we are limited by finite computational resources, we have to rely on an approximation of the real solution. The goal of simulation, therefore, is to minimize the difference ("error") between the exact and the approximated solution, and to ensure that the error

is below some accepted tolerance level.

In order to apply the finite element method, we used here different types of meshing on COMSOL to get the desired numerical results: Uniform mesh and refinement of the mesh in the center. We choose the free type of meshing to create the triangular mesh. How we choose the geometric entity level and domain on which we want to apply the mesh is explained in tables following the mesh examples. Here are the few examples which we used for our model problems in Figure 6 and Figures 7

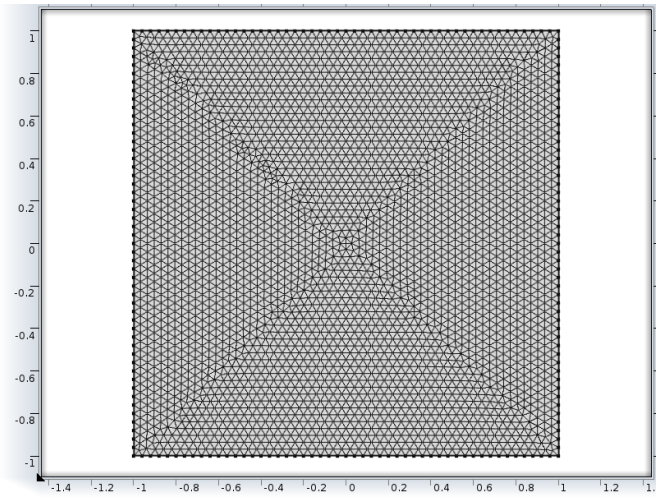


Figure 5: Uniform mesh on the whole domain D with built in extra-fine mesh size,

Property	Value
Minimum element quality	0.3414
Average element quality	0.9185
Triangular elements	26440
Edge elements	108
Vertex elements	4

Table 5: Mesh statistics for the mesh on figure (5).

When to form mesh elements (free triangles in two dimensions and tetrahedral in 3 dimensions) to form a piecewise linear approximation of the function the exactness of the approximation relay on the size and shapes of the element. In FEM the modification of the stiffness matrix also depends on the size and the shapes of the element. To compute the large sparse matrix system obtained by the finite element method LU factorization is used by Comsol. To get the

desired results the minimum element quality should be more than 0.1 and from the above table we can see our average element quality is very high and the minimum element is 0.3414. And we do not have any element which has quality less than 0.1.

Element quality

By default, poor quality elements are those elements having one or more of the following:

- Ratio of maximum side length to minimum side length is larger than 10.
- Minimum interior angle is smaller than 20 degree.
- Maximum interior angle is larger than 120 degree.

To measure the element quality following formula where used.
The element quality q for triangle is obtained by

$$q = \frac{4\sqrt{3}A}{h_1^2 + h_2^2 + h_3^2} \tag{24}$$

where A denote the area and h_1, h_2, h_3 the side-lengths.
For a tetrahedron the quality measures is evaluated using the formula

$$q = \frac{72\sqrt{3}V}{(h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 + h_6^2)^{\frac{3}{2}}} \tag{25}$$

where V denotes the volume, and the h:s are the edge lengths.

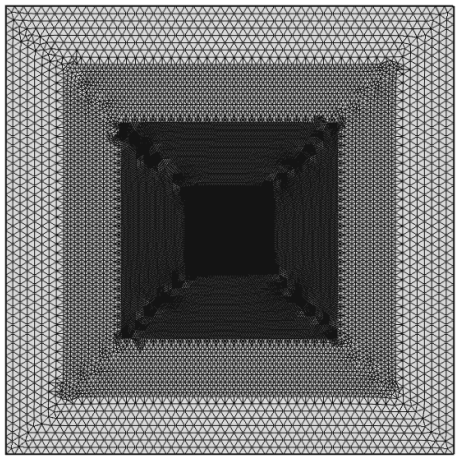


Figure 6: We used "fine" mesh size with one refinements in -0.25 to 0.25 and then another refinement from -0.5 to 0.5, finally a refinement from -0.75 to 0.75.

The original mesh of (6) is refined as follows: we have very small elements in the domain D_1 = [-0.5; 0.5] × [-0.5; 0.5] and large elements in the domain D_2 = [-0.75; 0.75] × [-0.75; 0.75] and rest of the domain $D/(D_1; D_2)$ is not refined.

Property	Value
Maximum element size	0.074
Minimum element size	2.5E-4
Resolution of curvature	0.25
Maximum element growth rate	1.25
Size Name	Finer

Table 6: Mesh statistics

Maximum element size — limits how big each mesh element can be.

Minimum element size — limits how small each mesh element can be.

Maximum element growth rate — limits the size difference of two adjacent mesh elements (lower value → finer mesh).
Curvature factor — limits how big a mesh element can be along a curved boundary (lower value → finer mesh).
Resolution of narrow regions — controls the number of layers of mesh elements in narrow regions (higher value → finer mesh).

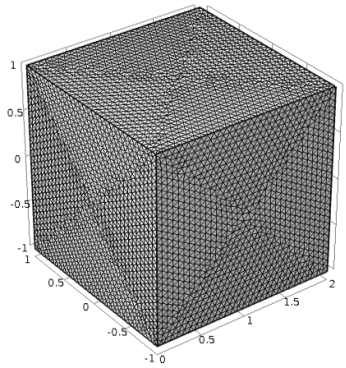


Figure 7: We did not use any refinement and use a uniform mesh on whole domain with built in fine mesh size and in three dimensions

In most of our implementations in 3d we have used the extra fine mesh size in which we have tetrahedral elements: 396552 and number of nodes are 8837. From figure (7) we can see that most of the low quality elements are located on the diagonal elements.

4.4 Numerical Implementation

We will justify what we defined in theory part, with our implementation which we did in COMSOL Multiphysics using LIVE LINK MATLAB. The parameters used in implementations are rely on the theory of chapter 3. We start with the discretization of the domain $I_\perp = I_y \times I_z$ using CG(1) method: continuous Galekin with piecewise linears combined with the backward Euler method in x. We choose small ϵ and $\delta \sim h$. Most of our graphical results are obtained using $\epsilon = 0.002$. Our mesh size h varies from 0.04 to 0.004. As we mentioned at meshing section, sometime we define different mesh size for different areas. Previous results in [9] , have an oscillatory behavior of computed solution for CG method, and layer behavior for SG method. To remove these oscillations one can use L_2 projection method is in [9] for current of pencil beam.

We have used the parameter α in initial conditions to avoid the singularities, e.g. at $(0, 0)$. Note also that the exact solution (23) has singularities at origin. This makes difficult to implement the numerical schemes as it is. The computed solution depends on the initial conditions, therefore it is not adequate to compare the computed results with (23), as the initial conditions were not exactly the same with ones we start with.

4.4.1 2D numerical examples using Maxwellian initial conditions

The FEM solution $w_h(x, y, z)$ with Maxwellian initial condition is shown in Figure (8). We used linear (p = 1) Lagrange elements to solve the problem, which means that the FEM solution $w_h(x, y, z)$ is a flat patch (linear functions) on each triangle of the mesh.

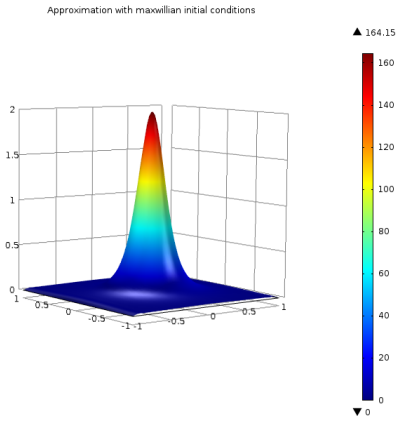


Figure 8: Computed FEM solution of model problem (6) with Maxwellian initial conditions

For the two dimensional model in COMSOL with Maxwellian initial condition we have used the function $f = \exp(-((x^2 + y^2) + \alpha))$. We used α in function is to control the formation of layer which appeared at point (x, y) = (0, 0). The numerical values for the parameters α= 0.19 and the time step was taken $k_n = 0.01$. The number of degree of freedom is 180481 (plus 804 internal DOFs). The diffusion coefficient is $\{\{0, 0\}, \{0, 0.002\}\}$ and the convection coefficient is taken as $\{0.1, 0\}$, damping or mass coefficient is zero. The physical memory used is : 1.06 GB and the virtual memory: 1.19 GB.

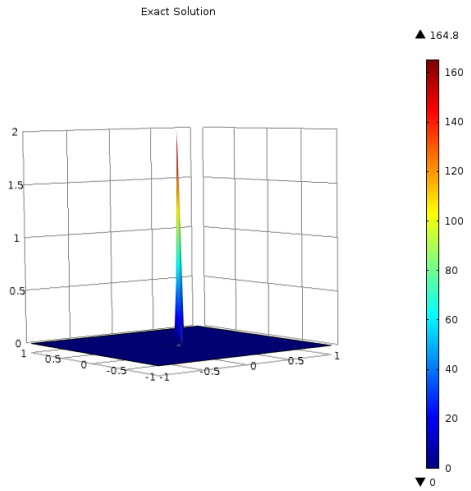


Figure 9: We have the exact solution of our model with same data we used to calculate the approximated solution

The mesh refinement is performed using free triangulation with the maximum element size $h = 0.134$ and the minimum element size 0.0006 consisting of three iterations in the domain from -0.2 to 0.2 in x-direction and -.2 to 0.2 in y-direction. Number of triangles are 7384 with number of nodes defined according to the formula for linear triangular elements. The minimum element quality is 0.3485 and the average element quality is 0.9224, which is much higher then the lower element quality 0.1 at which solution does not converge.

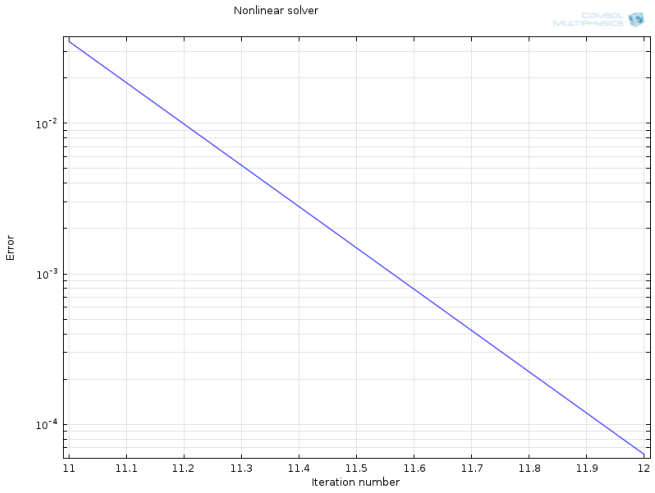


Figure 10: The above plot shows the behavior of error $e = \|w - w_h\|_{L_2}$ in this physics.

Convergence plot, which shows the error estimate decreasing between Newton-Raphson iterations.

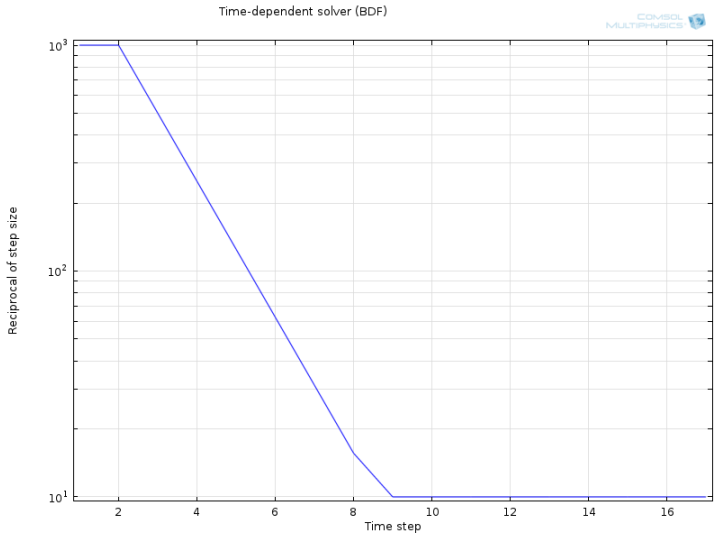


Figure 11: Convergence plot, which shows the relation between reciprocal of time step size vs time step

Time for computing the FEM solution in two dimensional model with Maxwellian initial conditions is 22 seconds.

3D numerical examples with maxellian initial conditions

Example 1

The FEM solution of our model problem calculated with three dimensional stationary model and the solution view in three dimensional slabs. The function used in three dimesional form is $f = \exp(1/(y^2 + z^2 + \alpha))$ and α used in this model is 0.16.

Linear Lagrange shape function represents to control the material density in our model, the solution contains the sensitivity of the objective function with respect to the discrete density value at each node point in the mesh. Because each node influences the density in a small surrounding region, the size of which varies from node to node, the individual sensitivities are not directly comparable to each others.

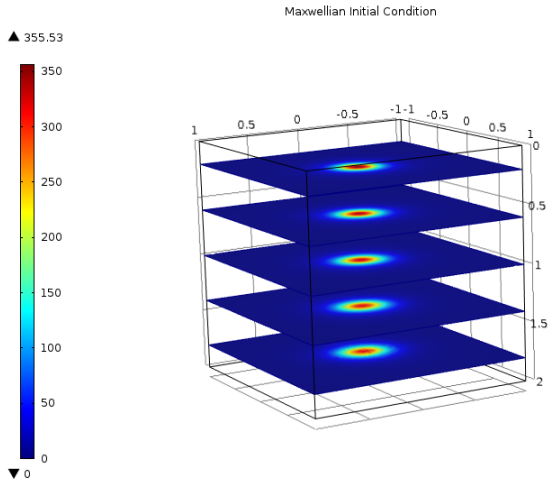


Figure 12: View of solution for model problem (6) with Maxwellian initial conditions 1. Here we have shape function being a Lagrangian and element order is quadratic.

Parameters used to obtain solution are as fellows. Number of nodes 541357 (with 24470 internal DOFs). The diffusion coefficient $\{\{0,0,0\},\{0,0,0\},\{0,0,0.002\}\}$ and the convection coefficient is taken as $\{1, 0.02 \times z, 0\}$, damping or mass coefficient is zero.

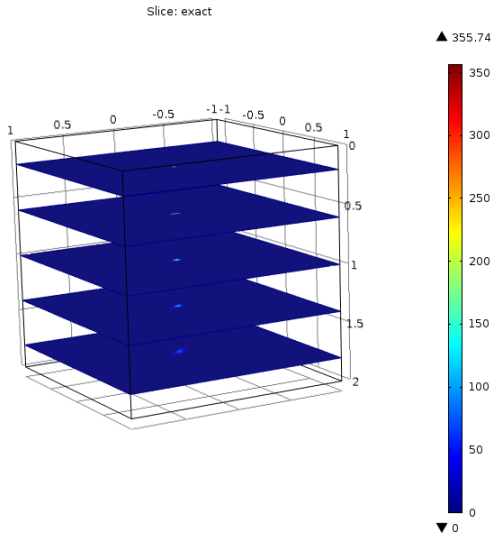


Figure 13: Exact solution of our model problem (23) view in 3d for comparing the results.

Time for generating solution is 100 s or (1 minute and 40 seconds), physical memory: 11.17 GB and virtual memory: 12.43 GB. The tetrahedral elements are 396552 triangular elements are 11884 and edge elements are 348. For the size of mesh here we used maximum element size 0.07 and minimum element size 0.003.

3D numerical examples with maxellian initial conditions.

Example 2
Numerical results obtained by Maxwellian initial condition in three dimensional model with height expression are shown here.

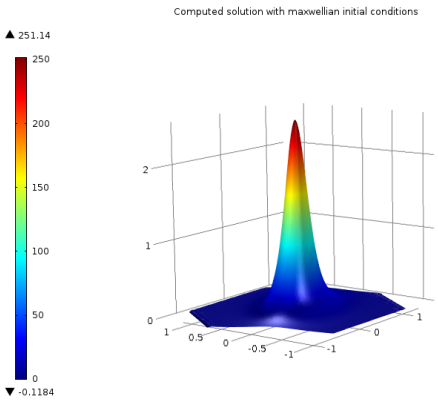


Figure 14: The solution for model problem (6) with Maxwellian initial conditions. Here we choose Lagrangian test function of the second order.

Number of degrees of freedom and the rest of the parameters (h, β, and α)are the same as in example (4.4.1). The time to generate the solution is 95 *second* or (1 minute and 35 seconds).

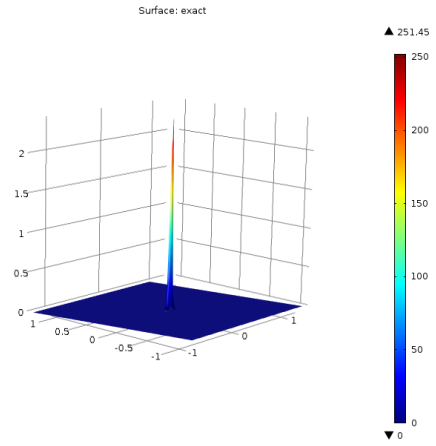


Figure 15: Exact solution of our model problem (23) in example 2.

4.4.2 Convergence study

In finite element method a finer mesh generally gives a more accurate solution. However, as a mesh is made finer, the computation time mostly increases. So one gets a mesh that satisfactorily balances accuracy and computing resources. One way to do this is to perform a mesh convergence study.

Here we denote our finite element solution by w_h and the exact solution by w. Then the error can be bounded by mesh size h. Having a bound for the norm of the error as $\|w - w_h\| \leq Ch^q$, where C is a constant independent of the mesh size, and q denotes the order of the convergence of the FEM. Generally we have $q = 1$ for the linear convergence, $q = 2$ for the quadratic convergence, and for faster convergence even higher values. We consider FEM error studies in the L_2 norm defined by

$$\|w\|_{L^2(\Omega)} = (\int_\Omega \{w(x)\}^2)^{1/2}. \tag{26}$$

For p=1 (the polynomial degree), i.e. the linear approximation we have a bound of the form $\|w - w_h\|_{L^2(\Omega)} \leq Ch^2$ known as a priori bound where w needed to be in $H^2(\Omega)$.

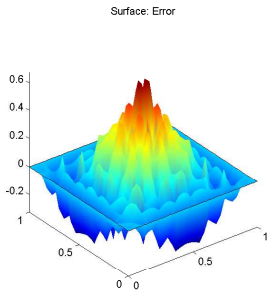
Because our PDE has a known solution w we may compute the norm of the error as

$$\|w - w_h\|_{L^2(\Omega)} \leq Ch^{p+1}. \tag{27}$$

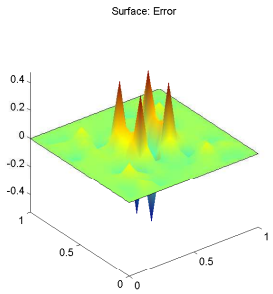
We start with initial mesh (*coarse*) and perform the refinements uniformly. Each refinement will subdivide the triangle of mesh into four triangles that mean h will be reduced to half in each refinement. We will denote the number of refinement by nr then the norm of error for each refinement will be $E_{nr} = \|u - u_h\|_{L^2(\Omega)}$. Now for each refinement we will define a table which has our computed values *number of elements* in the mesh, error of finite element method, square of the error, number of the degree of freedom

Number of refinement	Element	Vertices	DOF	$e_n = \|w - w_h\|_{L_2}$	e_n^2	e_n/e_{n+1}
0	272	157	157	8.278e-03	6.853e-05	0.00
1	1088	585	585	2.105e-03	4.431e-06	3.93
2	4352	2257	2257	5.290e-04	2.799e-07	3.98
3	17408	8865	8865	1.325e-04	1.754e-08	3.99
4	69632	35137	35137	3.313e-05	1.097e-09	4.00

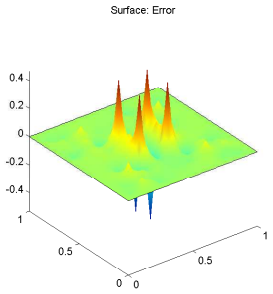
Table 7: Table for Lagrange elements when p=1



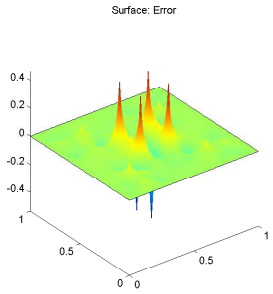
(a) Error view with no refinement



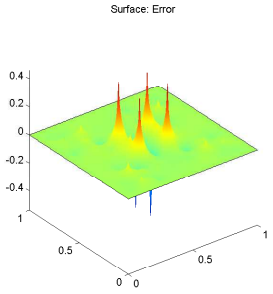
(b) error view with 1 refinement



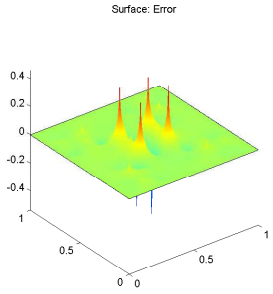
(a) Error view with 2 refinements



(b) error view with 3 refinements



(a) Error view with 4 refinements



(b) error view with 5 refinements

The figures above show how our error decreases graphically after each refinement. The results above are for the linear elements.

In the tables below we have calculated errors, i.e. the difference between all computed solutions which we get using all four initial conditions applying the considered finite element methods. In tables 13, 9 and 8 we calculated the errors

in L_1 , L_2 , L_∞ and $\tilde{L}_2$, norms where $\tilde{L}_2$ is the weighed L_2 norm, are define as

$$\|W\|_{\tilde{L}_2} = \left(\frac{1}{3} \sum_{M_i} |M_i| \sum_{j=1}^{3} (W(\zeta_j^i))^2 \right)^{1/2}, \tag{28}$$

where $|M_i|$ is the area of the mesh triangles and ζ_i^j are the middle points of edges of the mesh triangles M_i.

L_∞	Maxwellian	Hyperbolic	Dirac-I	DiracII
SSD	0.46	0.48	0.4	17
CSD	0.42	0.53	0.11	22
CG	0.33	0.52	0.28	16

Table 8: Error e for different initial conditions used for FEM approximation in the L_∞ - norm.

4.4.3 Hyperbolic initial conditions

Solving Fermi pencil-beam equation with hyperbolic initial data $f = \frac{1}{\sqrt{x^2+y^2+\alpha}}$. The function defined here is for two dimensional model and α= .19. Number of degree of freedom the model is solved for is: 180481 (plus 804 internal DOFs) and solution time is 22 s. The diffusion coefficient is $\{\{0,0\},\{0,0.002\}\}$ and the convection coefficient is taken as $\{0.2 \times y, 0\}$, the damping or mass coefficient is zero.

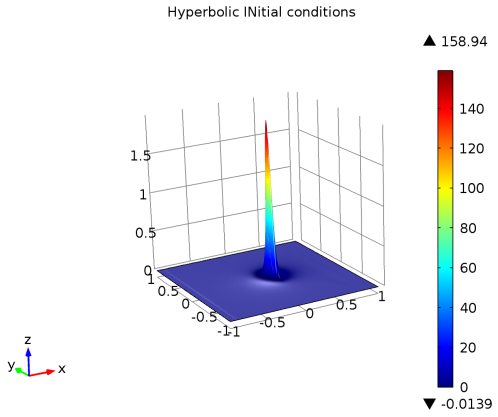


Figure 19: Above is the approximate solution with hyperbolic initial condition.

The mesh used here has maximum element size $h = 0.134$ and the minimum element size 0.0006 with three refinements in the domain from -0.2 to 0.2 in the x-direction and from -.2 to 0.2 in the y-direction. Number of triangles are

7384 with number of edge elements 120 and the vertex elements are 4. And the physical memory is 1.06 GB and virtual memory is: 1.19 GB.

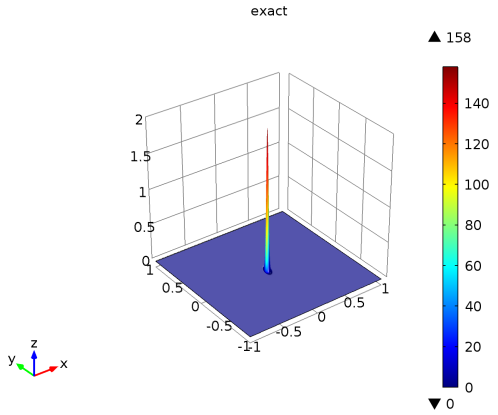


Figure 20: Here we have the exact solution of our model with the same data that we used to calculate the approximate solution

L_2	Maxwellian	Hyperbolic	Dirac-I	DiracII
SSD	0.24	0.18	0.19	18
CSD	0.21	0.10	0.11	9.6
CG	0.19	0.22	0.21	14

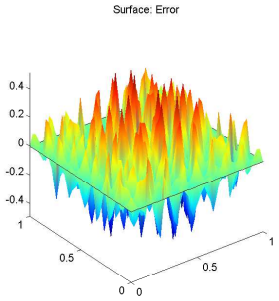
Table 9: Error for different initial conditions used for FEM, to compute the error in L_2.

Finite Element convergence study for $p = 2$

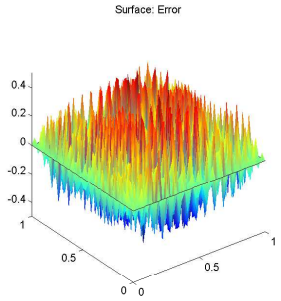
Number of refinement	Element	Vertices	DOF	$e_n = \|w - w_h\|_{L_2}$	e_n^2	e_n/e_{n+1}
0	272	157	585	2.706e-04	7.322e-08	0.00
1	1088	585	2257	3.427e-05	1.175e-09	7.90
2	4352	2257	8865	4.307e-06	1.855e-11	7.96
3	17408	8865	35137	5.398e-07	2.913e-13	7.98
4	69632	35137	139905	6.755e-08	4.564e-15	7.99

Table 10: Table for Lagrange elements when p=2

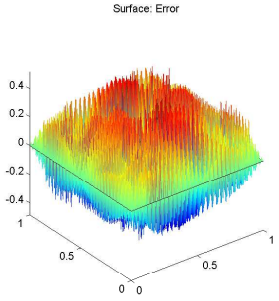
If we compare the tables (7) and (10) we can see that Lagrange elements of quadratic order (p=2) give more smoother numerical solution.



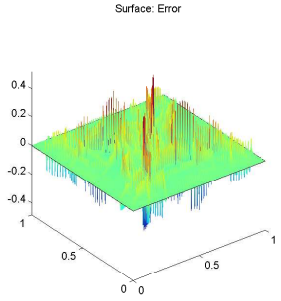
(a) Error view with no refinement



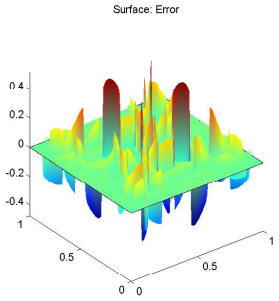
(b) error view with 1 refinement



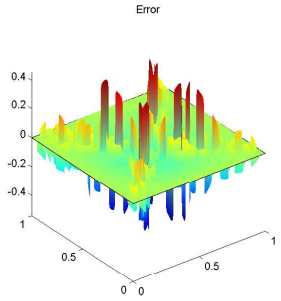
(a) Error view with 2 refinements



(b) error view with 3 refinements



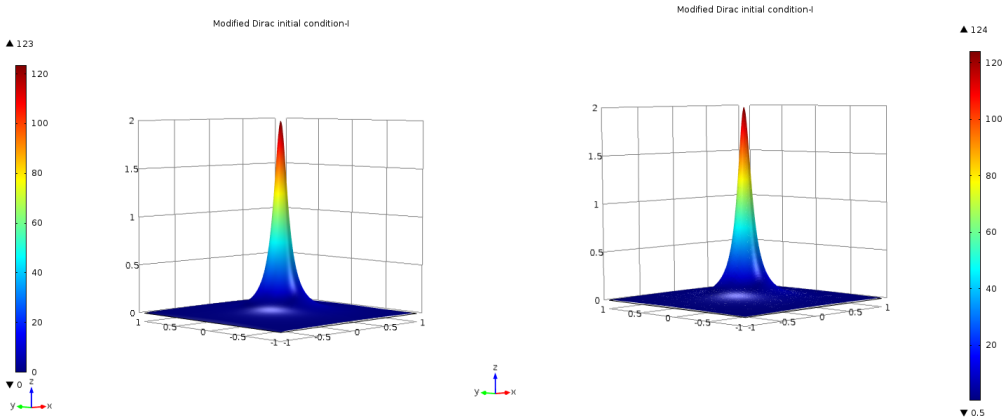
(a) Error view with 4 refinements



(b) error view with 5 refinements

The figures above show how our error decreases graphically after each refinement. The results above are with the quadratic order that mean p = 2.

4.4.4 Dirac initial conditions



(a) Lagrange and element order linear.

(b) Discontinuous Lagrange and element order linear.

(a) Computed solution at x = 1.5 of the model problem (6) with the Dirac initial conditions I, function $f = 1/(y^2 + z^2 + \alpha)$ and (α= 0.1) used in function is to avoid singularities at $(y, z) = (0, 0)$.

(b) Computed solution with discontinuous Lagrange elements at x = 1.5 of the model problem (6).

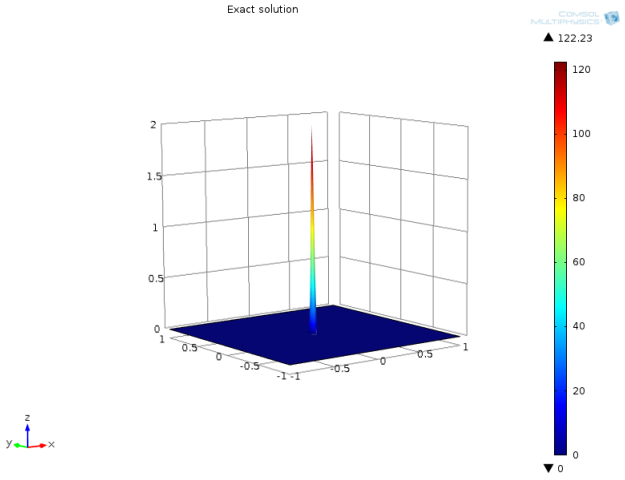
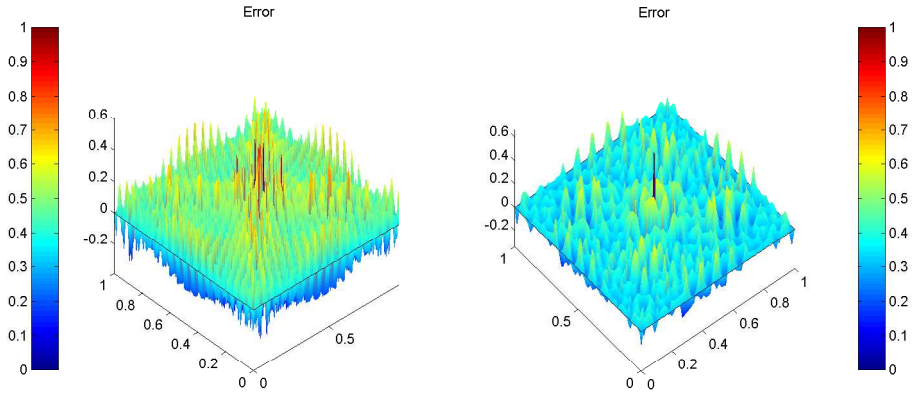


Figure 25: Exact solution .

Finite element convergence study for $p = 3$

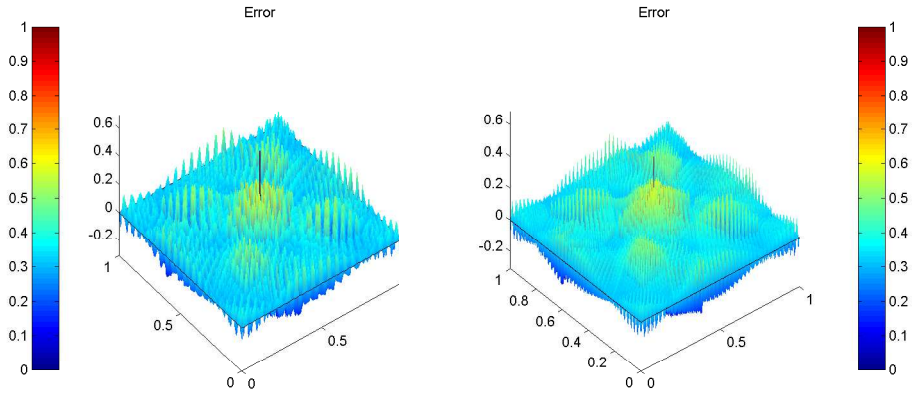
Number of refinement	Element	Vertices	DOF	$e_n = \|w - w_h\|_{L_2}$	e_n^2	e_n/e_{n+1}
0	272	157	1285	1.484e-05	2.204e-10	0.00
1	1088	585	5017	9.289e-07	8.629e-13	15.98
2	4352	2257	19825	5.799e-08	3.363e-15	16.02
3	17408	8865	78817	3.620e-09	1.311e-17	16.02

Table 11: Table for Lagrange elements for p=3



(a) Error view with no refinement

(b) error view with 1 refinement



(a) Error view with 2 refinements

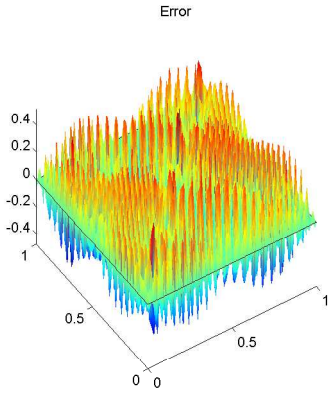
(b) error view with 3 refinements

The figures above show how our error decreases graphically after each refinement. The results above are for p=3.

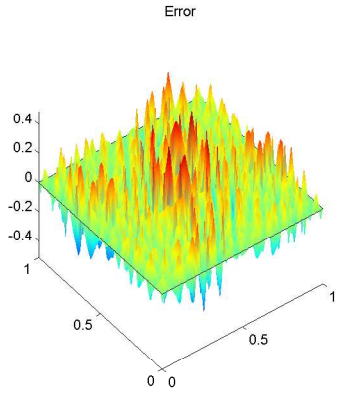
Finite element convergence study for $p = 4$

Number of refinement	Element	Vertices	DOF	$e_n = \|w - w_h\|_{L_2}$	e_n^2	e_n/e_{n+1}
0	272	157	1285	3.361e-07	1.130e-13	0.00
1	1088	585	5017	1.063e-08	1.129e-16	31.63
2	4352	2257	19825	3.341e-10	1.116e-19	31.81
3	17408	8865	78817	1.047e-11	1.097e-22	31.90

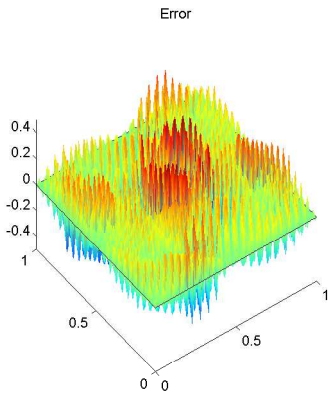
Table 12: Table for Lagrange elements when p=4



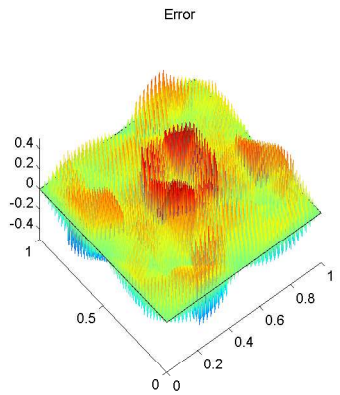
(a) Error view with no refinement



(b) error view with 1 refinement



(a) Error view with 2 refinements



(b) error view with 3 refinements

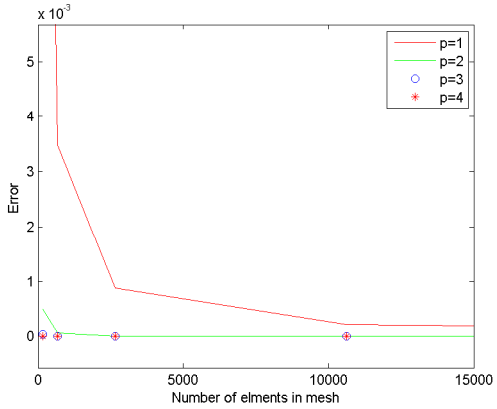


Figure 30: convergence study with respect to the element order

In the figure above we have number of nodes (elements) on the X-axis and the error on Y-axis. From the figure we can see that increasing the number of elements yields faster convergence. This can also be seen from the results for $p = 1$ and $p = 2$ in above figure.

Dirac initial condition solution view in three dimension.

For the three dimensional Dirac initial condition the function used is defined as $f = 1/(y^2 + z^2 + \alpha)$ where α is introduce in order to avoid the effect of singular point.

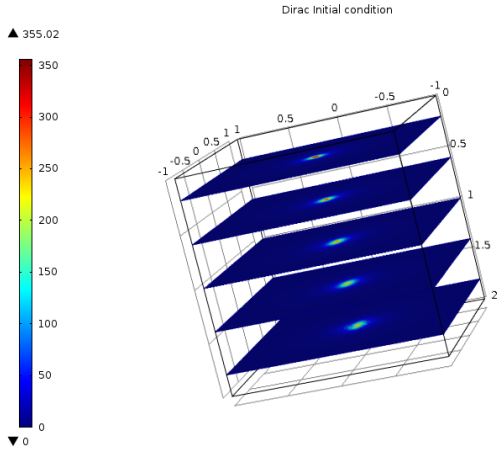


Figure 31: The solution for model problem (6) with Dirac initial condition 1. Here we have shape function as being polynomial of quadratic order.

We choose α=0.002. Number of degrees of freedom is 541357 (plus 24470 internal DOFs).
The diffusion coefficient is $\{\{0,0,0\},\{0,0,0\},\{0,0,0.002\}\}$ and the convection coefficient is taken as $\{1, 0.02 \times z, 0\}$, damping or mass coefficient is zero. Time generating a solutions is 100 *seconds* or (1 minute and 40 seconds) physical memory is 11.17 GB and virtual memory is 12.43 GB. The number of tetrahedral elements are 396552, triangular elements are 11884, and edge elements are 348. For the size of mesh here we used maximum element size 0.07 and minimum element size 0.003.

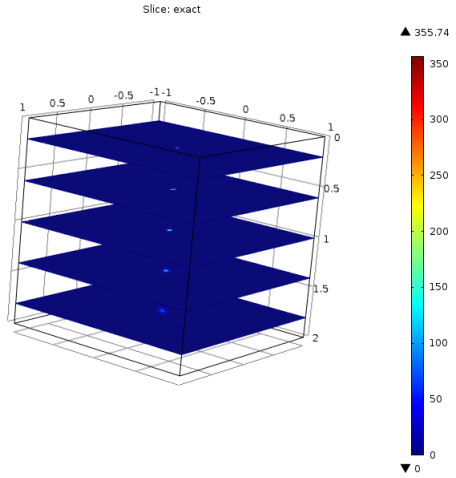


Figure 32: Exact solution of our model problem (23) and the exact solution in 3d.

Three dimensional solution of Dirac initial condition with height.

Next we test the environment as in above case of Dirac initial condition assuming $f = 1/(y^2 + z^2 + \alpha)$ with the same assumptions, however we change the convection coefficient from $\{1, 0.02 \times z, 0\}$ to $\{1, 0.1 \times z, 0\}$.

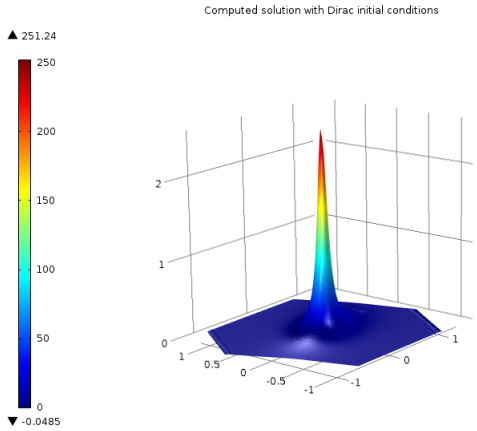


Figure 33: The solution for model problem (6) with Dirac-Initial conditions. Here we have shape function as being polynomial of Quadratic order.

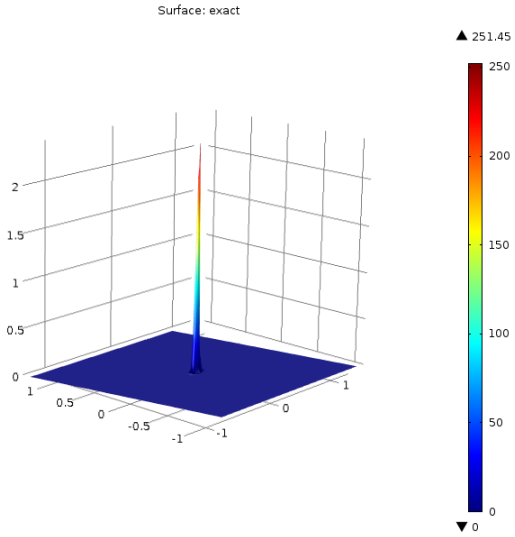


Figure 34: Exact solution of our model problem (23) and the exact solution in 2d, obtained from 3d model.

4.4.5 Modified Dirac initial conditions

For modified Dirac i.c. in two dimensional model we defined the function as $f = (1/\sqrt{x^2 + y^2 + \alpha}) \times 10$. The diffusion coefficient for two dimension as $\{\{0, 0\}, \{0, 0.002\}\}$ and the convection coefficient is taken as $\{0.1, 0\}$.

Damping or mass coefficient is zero.

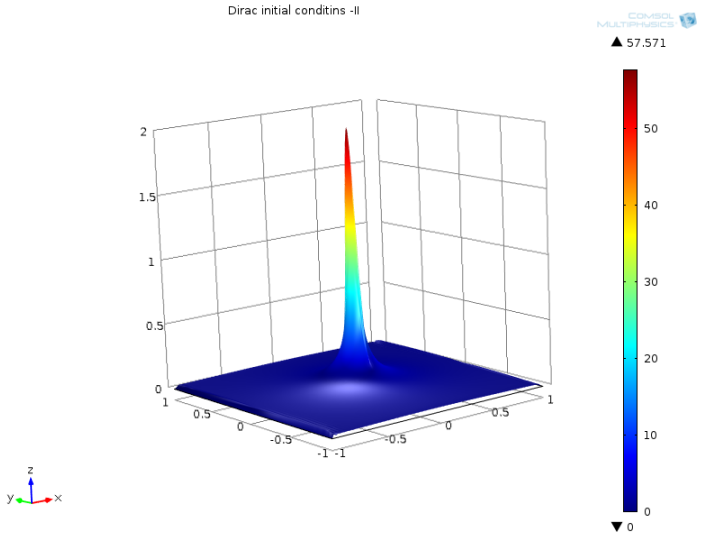


Figure 35: Above is the solution of model problem (6) with modified Dirac initial conditions. And below is the exact solution with the same mesh size. Here we have Lagrange shape function nd linear elements.

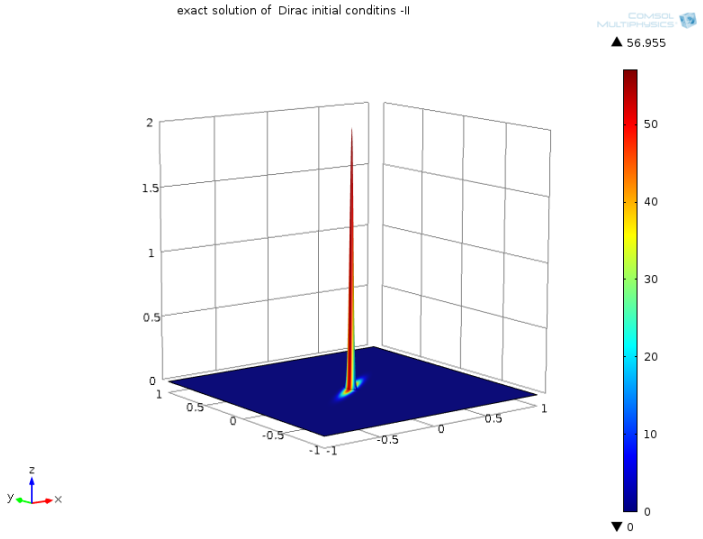


Figure 36: Above is the solution of model problem (6) with the Dirac initial conditions II. And below is the exact solution with same mesh size. Here we have shape function Lagrange and linear elements.

Modified Dirac initial condition results in three Dimensions

To compute the solution with modified Dirac i.c. in three dimensional model we defined the function $f = (1/\sqrt{y^2 + z^2 + \alpha}) \times 10$. The diffusion coefficient in triangulation is $\{\{0, 0, 0\}, \{0, 0, 0\}, \{0, 0, 0.002\}\}$ and the convection coefficient is taken $\{1, 0.02 \times z, 0\}$, damping or mass coefficient is zero.

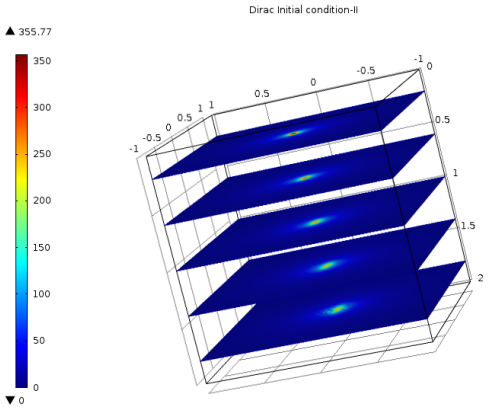


Figure 37: Solution for model problem (6) with Dirac initial condition. Here we have shape function as being polynomial of quadratic order.

The mesh used to obtain the above results has a step size $h = 0.025$ in y and z variable. In x direction the step size was chosen as $k = 0.0005$ so the total nodes we have solved is 6,561,000.

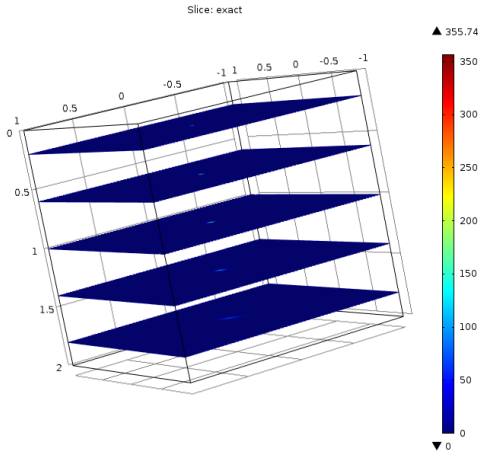


Figure 38: Exact solution of our model problem (23) and the exact solution in 3d.

The time to generated solution is 103 s or (1 minute and 43 seconds) physical memory: 11.25 GB and virtual memory is 12.51 GB.

Modified Dirac initial conditions solution with height.

Numerical results obtained for Dirac initial condition in three dimensional model, with height expression are shown bellow. The result is for $\alpha = 0.002$ with the same number of degrees of freedom as above.

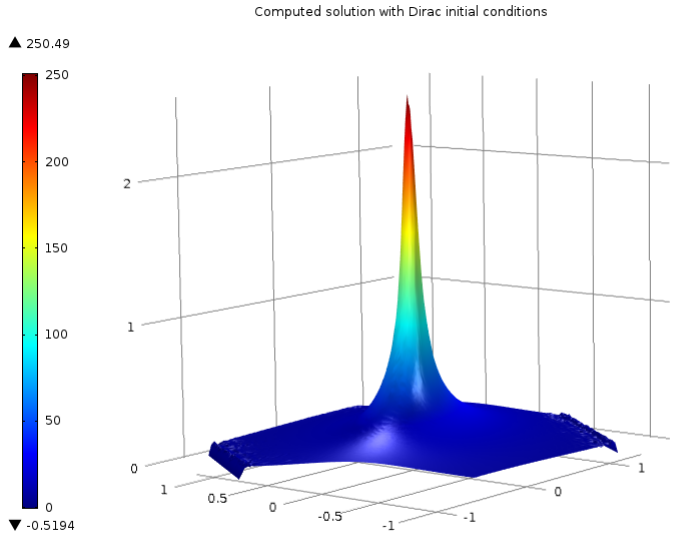


Figure 39: Solution for model problem (6) with Dirac-I initial conditions. Here we choose shape function as polynomial of quadratic order.

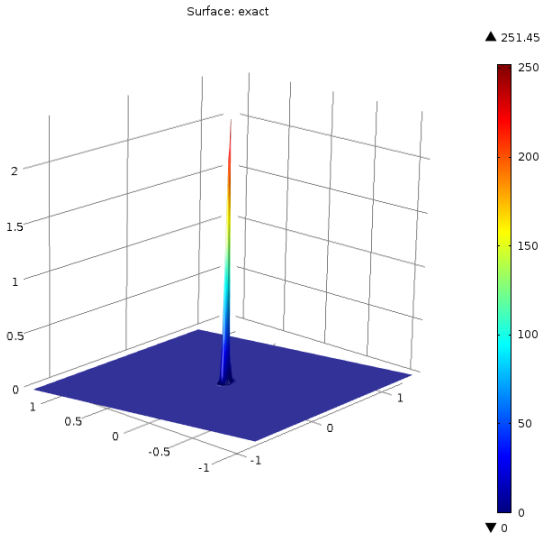


Figure 40: Exact solution of our model problem (23) and the exact solution in 2d, obtained from 3d model.

In Table below we have computed the L_∞ norms for error between exact and computed solution in table (13) we used Discontinuous Galerkin

L_1	Maxwellian	Hyperbolic	Dirac-I	DiracII
SSD	0.23	0.46	0.21	13
CSD	0.19	0.31	0.31	12.3
CG	0.28	0.28	0.33	13.21

Table 13: Error for different initial conditions used for FEM, in the L_1 setting.

5 Conclusion

Finite element method is commonly used as numerical method for solving the partial differential equations especially for elliptic PDEs and in our case for convection dominated convection-diffusion problem. We considered different algorithms for pencil beam model based on Fermi and Fokker Planck equations. These numerical schemes are derived for Galerkin methods such as Standard Galerkin, Semi-Streamline Diffusion, Characteristic Galerkin and for Characteristic Streamline Diffusion.

We choose certain initial data for implementations and illustrate the results of different algorithms. We demonstrated our results for beam problems in two and three dimensional models. As our aim is to remove the oscillatory behavior with hyperbolic initial conditions and the formation of layers in a solution with

maxwellian initial condition, we performed computations decreasing the coefficient of the convection term. But keeping in mind that still it will be convection dominated energy transfer problem.

We begin our computations with semi streamline diffusion and characteristic streamline diffusion methods. Our examples of section (4.4) justify that these are more stable and accurate compared to the standard Galerkin and characteristic Galerkin methods, for all types of initial data. As for convergence our tables of section (4.4) show that the solutions with Dirac initial condition are more suited in case of characteristic streamline diffusion method. We can also see using figures that Maxwellian initial condition gives accurate results in characteristics streamline diffusion scheme, while semi streamline diffusion method gives an accurate solution with hyperbolic initial conditions.

We can conclude that the oscillatory behavior obtained in [10] are appearing for problems with non-smooth initial data. In this work we removed these oscillations by modifying L_2 projections, and decreasing the dominance of the coefficient of convection term. The oscillations are also controlled by taking the small steps in the penetration variable.

References

[1] Mohammad Asadzadeh. On the stability of characteristic schemes for the fermi equation. Appl Comput Math. 2002;1(1):158–174.

[2] E W Larsen, Borgers Christoph. Asymptotic derivation of the fermi pencil-beam approximation. NuclSciEngrg. 1996;123:343–357.

[3] P Andreo. Monte Carlo Techniques in medical radiation physics. Phy Med Biol. 1991;36(3):861–920.

[4] B Rossi KGreisen. Cosmic-ray theory. Rev Mod Phys. 1941;13:309–340.

[5] L L Eyges . Multiple scattering with energy loss. Phys Rev. 1948;74:1534–35.

[6] K R Hogstrom P R Almond M D Mills. Electron beam dose calculation. Phys Med Biol. 1981;2:445–459.

[7] W Ulmer D Harder. A triple Gaussian pencil beam model for photon beam treatment planning Z. Med Phys. 1995;5:25–30.

[8] Adams AR. Sobolve spaces. New York: Academic press; 1975.

[9] M Asadzadeh, E W Larsen. Linear transport equations in flatland with small angular diffusion and their finite element approximation. Science Direct;p. 495 – 514.

[10] S Naqos. Numerical algorithms for electron beams, master thesis. Chalmers University of Technology, Department of Mathematics. 2005;.

[11] M Asadzadeh. Streamline diffusion methods for Fermi and Fokker–Planck equations,. Transport Theory Statistica Physics. 1997;26(3):319–340.

[12] M Asadzadeh. A posteriori error estimates for the Fokker–Planck and Fermi pencil beam equations. Math Models Meth Appl Sci. (2000);48(10 (5)):737–769.

[13] The MathWorks Inc. Matlab: the language of technical computing; 2014. Available from: `http://www.mathworks.se/products/matlab/`.

[14] G C Pomraning. The Fokker-Planck operator as an asymptotic limit. Math Models Methods Appl Sci. 1992;2:21–36.

[15] AhnesjÖ MMAspradakis. Dose calculation for external photon beams in radiotherapy. Phy Med Biol. 1999;44:R99–R155.

[16] C Johanson. A new approach to algorithms for convection problems which are based on exact transport + prjection. Comput Methods Appl Mech Engrg. 1992;(100):45–62.

Appendices

A Code

The models developed in Comsol Multiphysics and the Live Link Matlab code are available, and one can get help for them by email and my email address is, mianaseer@gmail.com

Matlab code obtained from COMSOL model with some changes.

Main Matlab file for running the file generated from COMSOL model.

A.1 Main file for running code

```
% set the maximum number of refinements
nrefmaximum = 3;

% set the order of the Lagrange elements used
p = 3;

% preallocate vectors:
nElem   = zeros(nrefmaximum+1,1);
Npts    = zeros(nrefmaximum+1,1);
nDoF    = zeros(nrefmaximum+1,1);
normsq  = zeros(nrefmaximum+1,1);
err     = zeros(nrefmaximum+1,1);
Rr      = zeros(nrefmaximum+1,1);
Qr      = zeros(nrefmaximum+1,1);


% obtain square of the norm of the FEM error
% on the refinement level r:
for r=0:nrefmaximum
    [e,nElem, nVertex, nnDofs]=fermiformatlab(r,p);
    %[e,nElem, nVertex, nnDofs]=maxwillian_3d_surface(r,p);
    normsq(r+1)=e;
    nElem(r+1) = nnElem;
    Npts(r+1) = nVertex;
    nDoF(r+1)  = nnDofs;
end

for r=0:nrefmaximum
    err(r+1) = sqrt(normsq(r+1));
    if r>=1
        Rr(r+1)=err(r)/err(r+1);
        Qr(r+1)=log(Rr(r+1))/log(2);
    end
```

```
end
fprintf('Lagrange Elements with order p = %2d and nrefmaximum ...\\
...= %3d \n',p, nrefmaximum)
fprintf('  r    N_e     N_v  DoF   enorminfsq     enorminf     Rr
Qr\n')
for r = 0:nrefmaximum
fprintf('%5d  %5d  %5d  %5d %11.3e %15.3e %9.2f %9.2f\n',r,nElem(r+1),..\\
   ...Npts(r+1),nDoF(r+1),normsq(r+1),err(r+1),Rr(r+1),Qr(r+1))
end
```

Matlab program from COMSOL model build in two dimensions.

A.2 Matlab Model Code

```
function [e, nElem, nVertex, nDofs] = Dirac_linear1(nref, p)
% Dirac_linear1.m
%
% Model exported on Nov 10 2014, 22:01 by COMSOL 4.4.0.248.

import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.modelPath('C:\Users\Naseer\Desktop');

model.modelNode.create('comp1');

model.geom.create('geom1', 2);

model.mesh.create('mesh1', 'geom1');

model.physics.create('c', 'CoefficientFormPDE', 'geom1', {'u'});

model.study.create('std1');
model.study('std1').feature.create('stat', 'Stationary');
model.study('std1').feature('stat').activate('c', true);

model.geom('geom1').feature.create('sq1', 'Square');
model.geom('geom1').feature('sq1').set('size', '2');
model.geom('geom1').feature('sq1').set('base', 'center');
model.geom('geom1').runPre('fin');
model.geom('geom1').run;

model.physics('c').feature('cfeq1').set('da', 1, '0');
model.physics('c').feature('cfeq1').set('be', 1, {'y' '0'});
```

```
model.physics('c').feature('cfeq1').set('f', 1, '0');
model.physics('c').feature('cfeq1').set('c', 1, 1, '0');
model.physics('c').feature('cfeq1').set('c', 1, 4, '0.002');
model.physics('c').prop('ShapeProperty').set('order', 1, p);
model.physics('c').feature.create('dir1', 'DirichletBoundary', 1);
model.physics('c').feature.create('zflx2', 'ZeroFluxBoundary', 1);
model.physics('c').feature.create('init2', 'init', 2);
model.physics('c').feature('dir1').selection.set([1 4]);
model.physics('c').feature('zflx2').selection.set([2 3]);
model.physics('c').feature('init2').selection.set([1]);
model.physics('c').feature('init2').selection.all;
model.physics('c').feature('init2').set('u', 1, 'a1');

model.variable.create('var1');
model.variable('var1').set('a1', '1/(x^2+y^2+.0081)');

model.mesh('mesh1').feature.create('ftri1', 'FreeTri');
model.mesh('mesh1').feature('size').set('hauto', '9');

model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
model.sol('sol1').feature('st1').set('studystep', 'stat');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature('v1').set('control', 'stat');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('fc1', 'FullyCoupled');
model.sol('sol1').feature('s1').feature('fc1').set('linsolver', 'dDef');
model.sol('sol1').feature('s1').feature.remove('fcDef');
model.sol('sol1').attach('std1');

model.result.create('pg1', 2);
model.result('pg1').set('data', 'dset1');
model.result('pg1').feature.create('surf1', 'Surface');
model.result('pg1').feature('surf1').set('expr', 'u');

model.sol('sol1').runAll;

model.result('pg1').run;
model.result('pg1').run;
model.result('pg1').feature('surf1').feature.create('hght1', 'Height');
model.result('pg1').run;
model.result.create('pg2', 'PlotGroup2D');
model.result('pg2').run;
model.result('pg2').feature.create('surf1', 'Surface');
```

```
model.result('pg2').feature('surf1').set('expr','((sqrt(3)/(pi*.002*1.5^2))..\\
..*exp((-2*(3*((x/1)^2)-3*(x/2)*y+y^2))/(0.002*1.5)))-u');
model.result('pg2').feature('surf1').set('descractive', 'on');
model.result('pg2').feature('surf1').set('descr', 'View of error');
model.result('pg2').feature('surf1').set('titletype', 'manual');
model.result('pg2').feature('surf1').set('title', 'view of error');
model.result('pg2').feature('surf1').feature.create('hght1', 'Height');
model.result('pg2').run;
model.result.numerical.create('int1', 'IntSurface');
model.result.numerical('int1').set('expr', '(((sqrt(3)/(pi*.002*1.5^2))...\\
...*exp((-2*(3*((x/1)^2)-3*(x/2)*y+y^2))/(0.002*1.5)))-u)^2');
model.result.table.create('tbl1', 'Table');
model.result.table('tbl1').comments('Surface Integration 1 ....\\
...((((sqrt(3)/(pi*.002*1.5^2))*...\\
...exp((-2*(3*((x/1)^2)-3*(x/2)*y+y^2))/(0.002*1.5)))-u)^2)');
model.result.numerical('int1').set('table', 'tbl1');
model.result.numerical('int1').setResult;
model.result.numerical('int1').selection.all;
model.result.numerical('int1').set('descractive', 'on');
model.result.numerical('int1').set('descr', 'Error');
model.result.table.create('tbl2', 'Table');
model.result.table('tbl2').comments('Surface Integration 1 ...\\
...((((sqrt(3)/(pi*.002*1.5^2))*...\\
...exp((-2*(3*((x/1)^2)-3*(x/2)*y+y^2))/(0.002*1.5)))-u)^2)');
model.result.numerical('int1').set('table', 'tbl2');
model.result.numerical('int1').setResult;
e=model.result.numerical('int1').getReal();
if (nref > 0)
model.mesh('mesh1').feature.create('ref1', 'Refine');
model.mesh('mesh1').feature('ref1').set('numrefine', nref);
model.mesh('mesh1').run;
model.mesh('mesh1').feature.create('ref2', 'Refine');
model.mesh('mesh1').feature('ref2').set('boxcoord', 'on');
model.mesh('mesh1').feature('ref2').set('numrefine', nref);
model.mesh('mesh1').feature('ref2').set('xmin', '-0.5');
model.mesh('mesh1').feature('ref2').set('xmax', '0.5');
model.mesh('mesh1').feature('ref2').set('ymin', '-0.5');
model.mesh('mesh1').feature('ref2').set('ymax', '0.5');
model.mesh('mesh1').run;

model.sol('sol1').study('std1');
model.sol('sol1').feature.remove('s1');
model.sol('sol1').feature.remove('v1');
model.sol('sol1').feature.remove('st1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
```

```
model.sol('sol1').feature('st1').set('studystep', 'stat');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature('v1').set('control', 'stat');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('fc1', 'FullyCoupled');
model.sol('sol1').feature('s1').feature('fc1').set('linsolver', 'dDef');
model.sol('sol1').feature('s1').feature.remove('fcDef');
model.sol('sol1').attach('std1');
model.sol('sol1').runAll;

model.result('pg1').run;
model.result.numerical('int1').set('table', 'tbl2');
model.result.numerical('int1').appendResult;
model.result.numerical('int1').set('table', 'tbl2');
model.result.numerical('int1').appendResult;
e=model.result.numerical('int1').getReal();
end
xmi = model.sol('sol1').xmeshInfo;
nDofs = xmi.nDofs;
nElem = model.mesh('mesh1').getNumElem;
nVertex = model.mesh('mesh1').getNumVertex;
figure;
mphplot(model, 'pg1')
filename = ['model_p', int2str(p), '_r', int2str(nref), '_sol', '.jpg'];
print('-djpeg100',filename);
figure;
mphplot(model, 'pg2')
filename = ['model_p', int2str(p), '_r', int2str(nref), '_err', '.jpg'];
print('-djpeg100',filename);
```