Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2024, 32 (1) 48–60
http://journals.rudn.ru/miph

Research article

UDC 519.65:519.217
PACS 07.05.Tp, 02.70.–c
DOI: 10.22363/2658-4670-2024-32-1-48-60 EDN: HEYUGO

Solving the eikonal equation by the FSM method in Julia language

Christina A. Stepa[1], Arseny V. Fedorov[1], Migran N. Gevorkyan[1], Anna V. Korolkova[1], Dmitry S. Kulyabov[1,2]

[1] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation*
[2] *Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation*

(received: December 15, 2023; revised: January 25, 2024; accepted: February 14, 2024)

Abstract. There are two main approaches to the numerical solution of the eikonal equation: reducing it to a system of ODES (method of characteristics) and constructing specialized methods for the numerical solution of this equation in the form of a partial differential equation. The latter approach includes the FSM (Fast sweeping method) method. It is reasonable to assume that a specialized method should have greater versatility. The purpose of this work is to evaluate the applicability of the FSM method for constructing beams and fronts. The implementation of the FSM method in the Eikonal library of the Julia programming language was used. The method was used for numerical simulation of spherical lenses by Maxwell, Luneburg and Eaton. These lenses were chosen because their optical properties have been well studied. A special case of flat lenses was chosen as the easiest to visualize and interpret the results. The results of the calculations are presented in the form of images of fronts and rays for each of the lenses. From the analysis of the obtained images, it is concluded that the FSM method is well suited for constructing electromagnetic wave fronts. An attempt to visualize ray trajectories based on the results of his work encounters a number of difficulties and in some cases gives an incorrect visual picture.

Key words and phrases: eikonal equation, geometric optics, wave optics, Julia language, Fast Sweeping Method

1. Introduction

In this article, we use the FSM (Fast Sweeping Method) method to solve the Eikonal equation using the example of three classical lenses: Luneburg, Maxwell and Eaton. These examples illustrate the limitations of the FSM method—it does a good job of calculating wave fronts, but is poorly applicable for calculating the trajectory of rays.

To model lenses, we use the Julia language and the Eikonal library, which implements the FSM method. The simulation results are visualized using the Mackie.jl library. Schematic illustrations are created using a separate vector graphics language, Asymptote.

1.1. Structure of the article

The paper consists of an introduction, a theoretical part, a description of the FSM (Fast sweeping method) [1–5], a description of implementation this method, a visualisation and discussion of the results.

In the theoretical part, the eikonal equation in Cartesian coordinate system is given, and the spherical lenses used for numerical experiments are schematically described.

In the next part, the numerical scheme of the FSM method with detailed formulas for the two-dimensional case is described, and its advantages and disadvantages compared to the feature method are briefly analyzed. Below is a brief description of the Eikonal library [6] for the Julia language [7] and a description of the program we have written that implements a numerical experiment.

In the final part, images of fronts and rays are presented, the results are analysed and conclusions are made about the advantages and disadvantages of the FSM method concerning the visualisation of the calculations.

1.2. Designations and agreements

For the purposes of this paper, we have followed standard notation, centred on the classic monograph [8]. All vector quantities are in bold, e.g., the position of a point X is denoted as $\mathbf{x} = (x^1, x^2, x^3)^T = (x, y, z)^T$. All vectors are considered columns and their components are numbered with upper indices. The eikonal function is denoted as $u(\mathbf{x})$, the refractive index is denoted as $n(\mathbf{x})$.

2. Eikonal equation

Previously, the authors in the paper [9] gave a detailed derivation of the eikonal equation for curvilinear coordinates, and in the paper [5] the characteristics method was considered. In this paper, however, we use the FSM method, which uses a rectangular grid, so the eikonal equation must be written in Cartesian coordinates.

We will consider the two-dimensional case of the eikonal equation in Cartesian coordinates with a boundary condition:

$$\begin{cases} \left(\dfrac{\partial u}{\partial x}\right)^2 + \left(\dfrac{\partial u}{\partial y}\right)^2 = n^2(x, y), \\ u(x, y) = 0, \quad (x, y) \in \Gamma \subset \mathbb{R}^2. \end{cases} \tag{1}$$

By specifying the points of the set Γ, the position of the radiation source and the boundary of the medium are specified. Written in this form, the eikonal equation is a nonlinear boundary value problem for the hyperbolic partial derivative equation.

Two approaches to the numerical solution of the eikonal equation can be distinguished:

- transformation to a system of ordinary differential equations (a system of Hamilton equations) by the method of characteristics [10, 11], and then applying one of numerous methods for the numerical solution of such equations;
- approach to the problem as a stationary boundary value problem: development of an efficient numerical algorithm for solving the system of nonlinear equations obtained by discretization (this type of methods includes, for example, the fast marching method).

In this paper, we focus on a method called FSM (*Fast Sweeping Method*). The authors do not know the standard translation of the name of this method, so the abbreviation FSM is used throughout the text.

The method was proposed in 2000 [1]. The basic idea of the method is to use Godunov's counter-flow difference scheme and Gauss–Zeidel iterative scheme with variable order of passing the mesh nodes. A detailed description of the numerical scheme is given in the Section 4.

FSM is simple to implement and requires a finite number of iterations. The complexity of the algorithm is $O(N)$ for N grid points. The number of iterations is independent of the number of grid nodes (of the grid size). The FSM method can be extended to the general case of the Hamilton–Jacobi equation.

3. Luneburg, Maxwell and Eaton lenses

3.1. General description of the lenses

Consider a lens, which is a sphere with centre at point X_0 and radius vector $\mathbf{X}_0$. The source of electromagnetic waves is placed at a point with radius vector $\mathbf{x}_0$.

It should be emphasised immediately that the method used for the numerical solution of the eikonal equation entails a different mathematical description of the electromagnetic wave.

- When using the method of characteristics, it is natural to interpret the radiation in the form of rays. Each ray in this case is a solution of the ODE system for given initial values of the generalised coordinates $\mathbf{x}$ and impulses $\mathbf{p}$. The initial values of the coordinates $\mathbf{x}_0$ specify the position of the source, i.e., the beginning of the ray, and the initial values of the impulses $\mathbf{p}_0$ specify the direction of the ray.
- The FSM method uses the wave interpretation of optics and assumes that the eikonal function $u(\mathbf{x})$ is initially defined at each point in space, or more precisely at each grid point (see section 4).

The location of the source is given by the boundary condition $u(\mathbf{x}_0) = 0$ of the system (1), where $\mathbf{x}_0$ is the radius vector of the points belonging to the source.

Figures 1 and 2 show lenses that receive electromagnetic radiation from a point source (figure 1) and a flat extended source (figure 2). The radiation in the figures is represented as rays, but in the FSM method there is no way to explicitly specify the direction and source of the rays, as it is assumed that the radiation is already present at every point in the region under consideration. This causes certain difficulties when it is necessary to visualise exactly the ray optical pattern.

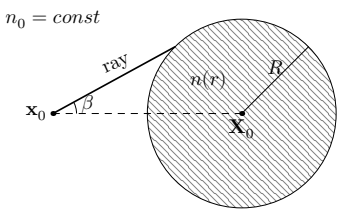

Figure 1. Lens with a point source

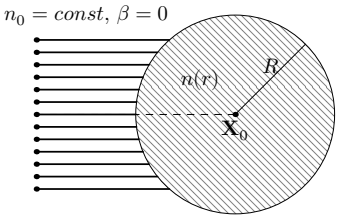

Figure 2. Lens with a flat light source

The refractive index outside the lens is constant and equal to n_0, while inside the lens it is a function of the distance from the centre of the lens to the current point $n(r)$, where $r = \|\mathbf{x} - \mathbf{X}_0\|$. To simplify the calculations, the centre of the lens should be placed at the origin. This especially simplifies the solution of the problem in cylindrical and spherical coordinates.

It may be convenient to define the location of a point source relative to the lens. Then its coordinates are determined by the lens radius R, the distance from the lens to the source d, and the angle θ, which is set off in a counterclockwise direction in the right-hand coordinate system, as shown in the figure 3. Then the radius of the source vector is given by the parametric equation of a circle with radius $R + d$

$$\mathbf{x}_0 = \mathbf{X}_0 + (d + R)(\cos\theta, \sin\theta)^T.$$

In particular, if the source lies on the lens as shown in the figure 4, its coordinates are given by the radius vector

$$\mathbf{x}_0 = \mathbf{X}_0 + R(\cos\theta, \sin\theta)^T.$$

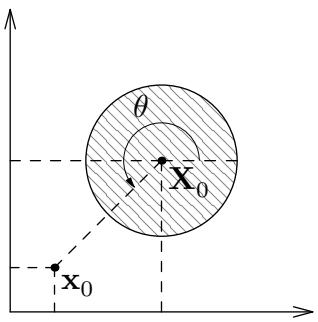

Figure 3. The location of the point source is given relative to the lens

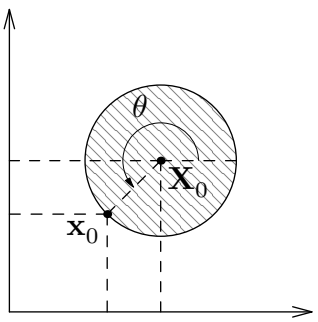

Figure 4. The point source is placed on the lens

The numerical scheme of the FSM method does not require to know the derivatives of the refractive index function $n(\mathbf{x})$, which can be considered as an advantage of this method over the characteristic method.

3.2. Luneburg lens

A Luneburg lens [8, 12] is a spherical lens of radius R centred at the point (X_0, Y_0, Z_0) with a refractive index of the following form

$$n(r) = \begin{cases} n_0\sqrt{2 - \left(\frac{r}{R}\right)^2}, & r \leqslant R, \\ n_0, & r > R, \end{cases} \tag{2}$$

where in Cartesian coordinates $r(x, y, z) = \sqrt{(x - X_0)^2 + (y - Y_0)^2 + (z - Z_0)^2}$ is the distance from the centre of the lens to an arbitrary point (x, y, z). It follows from the formula that the coefficient n varies continuously from $n_0\sqrt{2}$ to n_0 starting from the centre of the lens and ending at its boundary. For calculations it is more convenient to rewrite the expression for r in index form:

$$r(x^1, x^2, x^3) = \sqrt{(x^1 - X_0^1)^2 + (x^2 - X_0^2)^2 + (x^3 - X_0^3)^2} = \sqrt{\sum_{i=1}^{3}(x^i - X_0^i)^2}.$$

3.3. Maxwell lens

Maxwell's lens [8, 13] is also a spherical lens of radius R centred at the point $\mathbf{X}_0$ with a refractive index of the following kind:

$$n(r) = \begin{cases} \dfrac{n_0}{1 + \left(\frac{r}{R}\right)^2}, & r \leqslant R, \\ n_0, & r > R. \end{cases} \tag{3}$$

Figure 5 plots the change in refractive index for Maxwell and Luneburg lenses as a function of the radius of the point vector.

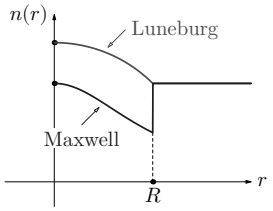

Figure 5. Refractive index of the Luneburg lens and the Maxwell lens

3.4. Eaton Lens

In the flat case, Eaton's lens [14] is a disc formed by two circles with radii R and $2R$, which is shown in the diagram of the lens in the figure 6

$$n(r) = \begin{cases} n_0\sqrt{\dfrac{2R}{r} - 1}, & r \in [R, 2R], \\ n_0, & r \notin [R, 2R]. \end{cases} \tag{4}$$

3.5. Source location

The optical properties of lenses are clearly shown by placing the radiation source at a specific point.

– For a Luneburg lens, the point source is usually placed on the surface of the lens so that the outgoing rays passing through the lens are parallel to each other, as shown in the figure 7.

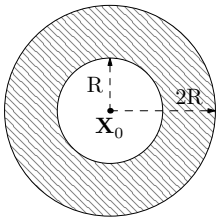

Figure 6. The scheme of the Eaton lens

– For Maxwell lenses, a point source is also placed on the surface of the lens. In this case, the outgoing rays are focused and converge at a point diametrical to the source, as shown in the figure 8.
– For the Eaton lens, the source is placed inside a sphere (circle) of small diameter R. In this case, all radiation does not extend beyond the lens and the rays are focused at a point symmetrical about the centre of the lens, as shown in the figure 9.

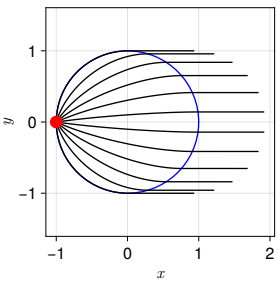

Figure 7. Luneburg lens

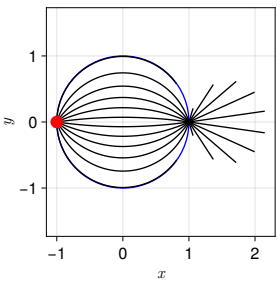

Figure 8. Maxwell lens

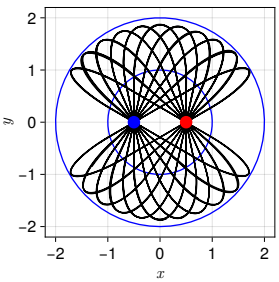

Figure 9. Eaton lens

The ray paths in figures 7, 8, and 9 are calculated using the characteristic [5] method.

If we consider the above source configuration, some regions of space will be completely free of rays. For an Eaton lens, all rays will be enclosed inside the large circle of the lens. For the Luneburg and Maxwell lenses, the areas where rays can pass through are shown by the hatching in the figures 10 and 11. No ray can penetrate the white colour regions.

The figures 10 and 11 are only schemes that conventionally show the areas where rays are present and absent. However, the ray paths are not shown inside the lens.

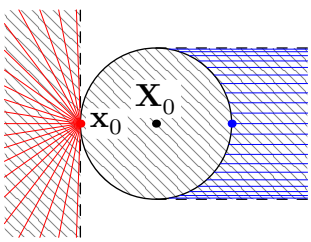

Figure 10. Luneburg lens

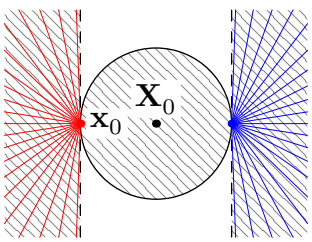

Figure 11. Maxwell lens

4. Fast sweeping method

4.1. FSM for two-dimensional eikonal equation

Let us proceed to the description of the numerical scheme. Let us divide the whole integration region into discrete nodes using the rectangular grid shown in the figure 12, where

- along the axis Ox we have I partition points $x_1 < x_2 < x_3 < \ldots < x_{I-1} < x_I$,
- along the Oy-axis, we have J partition points $y_1 < y_2 < y_3 < \ldots < y_{J-1} < y_J$.

The grid will consist of $I \times J$ nodes with coordinates (x_i, y_j), where $i = 1, \ldots, I$, and $j = 1, \ldots, J$.

Assume that the partitioning is chosen such that the grid spacing h is the same for both axes. The grid function u_{ij} approximates the function $u(x, y)$ at the grid nodes, i.e., $u_{ij} \approx u(x_i, y_j)$ and only at the point (x_0, y_0) does exact equality hold (figure 13).

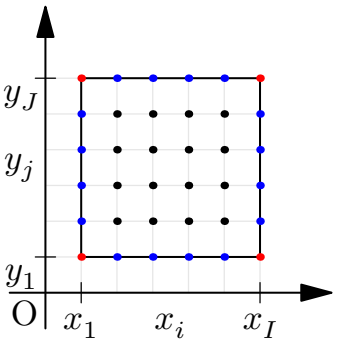

Figure 12. Different grid points of partitioning of the integration domain

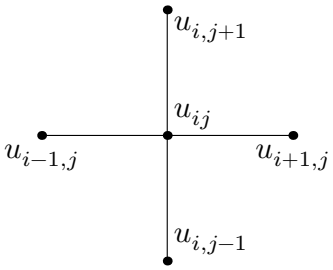

Figure 13. Numerical scheme template. u_{ij} are the grid functions

All grid points can be divided into three groups, which are highlighted in the figure 12 with different colours:

1. The internal grid points with indices $i = 2, \ldots, I-1$ and $j = 2, \ldots, J-1$ are shown in black in the diagram 12.
2. Points of the four grid boundaries with the indices
   - left boundary: $i = 1, \quad j = 2, \ldots, J-1$,
   - right boundary: $i = I, \quad j = 2, \ldots, J-1$,
   - lower boundary $i = 2, \ldots, I-1, \quad j = 1$,
   - upper boundary $i = 2, \ldots, I-1, \quad j = J$,

   are shown in blue in the diagram 12.
3. Corner points with the following fixed indices:
   - lower left corner point $i = 1, \quad j = 1$,
   - upper left corner point $i = 1, \quad j = J$,
   - lower right corner point $i = I, \quad j = 1$,
   - upper right corner point $i = I, \quad j = J$,

   are marked in red in the diagram 12.

Let's move on to presenting the algorithm. The Godunov' scheme (counter-current difference scheme) is used as the sampling scheme for the interior points of the domain. Let us introduce the following notations:

$$u_{xmin} = \min(u_{i-1,j}, u_{i+1,j}), \; u_{ymin} = \min(u_{i,j-1}, u_{i,j+1}), \; n_{ij} = n(x_i, y_j),$$

and also an indicator function:

$$(x)^+ = \begin{cases} x, & x > 0, \\ 0, & x \leqslant 0. \end{cases}$$

To initialise the calculations, first of all, it is necessary to set the values of the grid function $u_{ij} = 0$ on the boundary Γ. These values will remain unchanged in the subsequent calculations. For all

other points of the grid function u_{ij}, it is necessary to assign sufficiently large positive values, which it obviously cannot reach. During the operation of the numerical scheme these values will be recalculated.

The computational process consists of four sweeps of the entire rectangular region. Each such sweep is two nested loops, where the indices are run in the following order:

- $i = 1, \dots, I$ and $j = 1, \dots, J$ — direct order,
- $i = I, \dots, 1$ and $j = 1, \dots, J$ — mixed order,
- $i = I, \dots, 1$ and $j = J, \dots, 1$ — reverse order,
- $i = 1, \dots, I$ and $j = J, \dots, 1$ — mixed order.

At each step, we should solve a nonlinear equation whose coefficients for each group of points will change slightly.

Group I Points $i = 2, \dots, I-1$ and $j = 2, \dots, J-1$

$$[(u_{ij} - u_{xmin})^+]^2 + [(u_{ij} - u_{ymin})^+]^2 = n_{ij}^2 h^2$$

Group II The following points belong to this group:

- left boundary: $i = 1, \quad j = 2, \dots, J-1$:

$$[(u_{1j} - u_{2j})^+]^2 + [(u_{1j} - u_{ymin})^+]^2 = n_{1j}^2 h^2,$$

- right boundary: $i = I, \quad j = 2, \dots, J-1$:

$$[(u_{Ij} - u_{I-1,j})^+]^2 + [(u_{Ij} - u_{ymin})^+]^2 = n_{Ij}^2 h^2,$$

- lower boundary $i = 2, \dots, I-1, \quad j = 1$:

$$[(u_{i1} - u_{xmin})^+]^2 + [(u_{i1} - u_{i2})^+]^2 = n_{i1}^2 h^2,$$

- upper boundary $i = 2, \dots, I-1, \quad j = J$:

$$[(u_{iJ} - u_{xmin})^+]^2 + [(u_{iJ} - u_{i,J-1})^+]^2 = n_{iJ}^2 h^2.$$

Group III The following points belong to this group:

- left lower corner point $i = 1, \quad j = 1$:

$$[(u_{11} - u_{21})^+]^2 + [(u_{11} - u_{12})^+]^2 = n_{11}^2 h^2,$$

- upper left corner point $i = 1, \quad j = J$:

$$[(u_{1J} - u_{2J})^+]^2 + [(u_{1J} - u_{1,J-1})^+]^2 = n_{1J}^2 h^2,$$

- lower right corner point $i = I, \quad j = 1$:

$$[(u_{I1} - u_{I-1,1})^+]^2 + [(u_{I1} - u_{I2})^+]^2 = n_{I1}^2 h^2,$$

- upper right corner point $i = I, \quad j = J$:

$$[(u_{IJ} - u_{I-1,J})^+]^2 + [(u_{IJ} - u_{I,J-1})^+]^2 = n_{IJ}^2 h^2.$$

Each of these equations differs only in the numerical coefficients and has the following form:

$$\left[(x-a)^+\right]^2 + \left[(x-b)^+\right]^2 = n_{ij}^2 h^2.$$

It can be reduced to a quadratic equation and the solution can be written as

$$x = \begin{cases} \min(a,b) + n_{ij}h, & |a-b| \geqslant n_{ij}h, \\ \dfrac{a+b+\sqrt{2n_{ij}^2h^2 - (a-b)^2}}{2}, & |a-b| < n_{ij}h. \end{cases}$$

4.2. FSM calculation in Julia language

For numerical modelling of lenses using the FSM method, we used the Eikonal [6] package for the Julia [15, 16] programming language. The Eikonal package is a small library that implements the Fast Sweeping and Fast Marching methods. The package is registered in the official Julia package repository and can be installed using standard methods.

Both methods are implemented for arbitrary dimensionality and their source code can fit in a single source file. For such a small package, the documentation is quite detailed and allows you to understand the functionality of the package relatively quickly. There is also a set of tests, which can also serve as illustrative examples.

Let's consider the use of the Eikonal library to compute fronts in the case of Maxwell and Luneburg planar lenses. In addition to this library, we will use our Lenses module described above, and also SVector package.

First of all, let's import all the necessary modules. We read the source code of the Lenses module using `include` and then add it to the common namespace using `using`. Import modules from the official repository using `using`.

```
include("../src/lense.jl")

# We use the FSM method implemented in the Eikonal library
using Eikonal
using StaticArrays: SVector
using .Lenses
```

We need a parametric equation of the circle that returns the coordinates as integers, because the Eikonal module uses an integer grid. We define it as a one-line function. Using the syntax of a dot with an operator or function (e.g. `.+`) allows you to apply the function and operator to all elements of an array or tuple at once.

```
circle_xy(center, R, ϕ) = round.(Int, center .+ (R*cos(ϕ), R*sin(ϕ)))
```

We define the required parameters in the form of constants. Each constant is provided with a documentation line, so no additional comments are needed. The lens type is selected using the `select_lense` function from the Lenses module, which allows you to choose which lens to calculate for right at the beginning of the program.

```
const centre = (500, 500)
"Lens radius"
const R = 300
"Refractive index"
const n₀ = 1.0
"Position of the source relative to the lens"
const source = circle_xy(center, R, π)
"Lens type"
const LENSE = select_lense(length(ARGS)>=1 ? ARGS[1] : "")(R, n₀,
 ↪  SVector(center..., 0.0))
"To draw rays or not"
const RAYS = true
```

Next, we set the grid size (I, J) for the approximating function u_{ij}, initialise the method using the function `FastSweeping` and initialise the array `v`, which in this library denotes the values of the refractive index n_{ij} in the grid nodes. To calculate the values of n_{ij} we use the function `n` from Lenses.

```
const tsize = (1000, 1000)

fsm = FastSweeping(Float64, tsize)

for x=1:tsize[1], y=1:tsize[2]
    fsm.v[x, y] = Lenses.n(SVector(x, y, 0.0), LENSE)
end
```

At the next step we start the calculations. The function `init!` allows you to set the boundary values, which in our case consist of a single point - the ray source. The result of calculations will be written in the attribute `fsm.t`, in our notations it is the approximating grid function u_{ij}.

```
println("Sweeping")
sweep!(fsm, verbose=false)
```

After obtaining the computational results, it remains to visualise the fronts and rays. We use the Makie library, while the Eikonal examples use the Plots library. We create images, axes on it, lens contour as a circle and display it on the coordinate plane. The source is visualised as a point. The fronts are visualised trivially, using the function `contour!`, which displays level lines for a function from two variables u_{ij}. The coordinate arrays x and y may not be passed to it, as they are the same as the indices i and j since we have defined an integer grid.

```
fig01 = Figure(size=(500, 500))
ax01 = Axis(fig01[1, 1])

# Contour of the lens as a circle
const lense_contour = Circle(Point2(center .|> Float64), R)

# Draw the contour of the lens
lines!(ax01, lense_contour, color=:blue)

# Source as a point
scatter!(ax01, source..., color=:red)

# Wavefront as contours of a function from two variables u(x, y)
contour!(ax01, fsm.t, levels=100)
```

Visualising rays is not such a trivial task anymore. The author of the Eikonal library has provided the `ray` function, which calculates ray points using the fastest gradient descent method. To use it, you need to specify the end point of the ray. The ray source will be selected as the ray origin.

```
  for θ = pi/6:0.025:pi/3
    pos = circle_xy(source, min(tsize...) - min(source...), θ)
    r1 = ray(fsm.t, pos)
    lines!(ax01, r1, color=:green)
  end
end
```

It should be noted that the `ray` function is unstable, because often its execution is accompanied by an exit beyond the `fsm.t` array boundaries, even if the ray end point is specified inside the rectangular area.

The result of the visualisation is shown in the figures 14 and 15. It should be noted here that in the case of the Maxwell lens, the application of the `ray` function gives rise to the following problem.

Rays emanating from a source on the surface of the lens must focus to a point (focus) located on the diametrically opposite side of the lens from the source and can only exit the lens after passing that point, as shown in the figure 16. So, when modelling electromagnetic radiation in the form of rays, only rays that have passed the lens and left the point of focus can exist outside the lens. It should be specified that we consider a beam of rays coming out of a point source at an angle in the interval $(-\pi/2, \pi/2)$.

Since the FSM method approximates the eikonal function $u(\mathbf{x})$ at each grid point, electromagnetic radiation will be present everywhere outside the lens in this modelling, as can be seen from the image of the fronts. Therefore, if you set the `ray` parameter `pos` to a point outside the lens surface when calling the `ray` function, the rays will be drawn incorrectly, as shown in the figure 18.

It is possible to obtain a conventionally correct image by specifying as the final position only those grid points that lie exclusively on the surface of the lens.

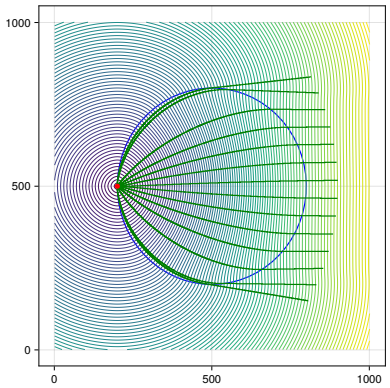

Figure 14. Fronts for the Luneburg lens

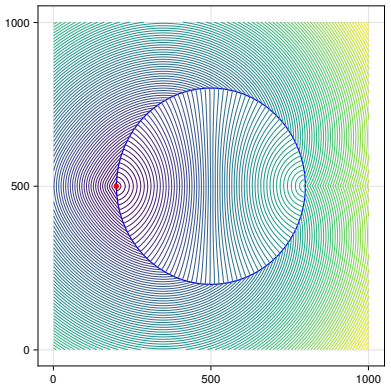

Figure 15. Fronts for the Maxwell lens

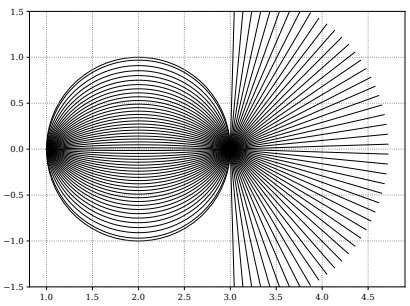

Figure 16. Correct image of the rays of Maxwell lens

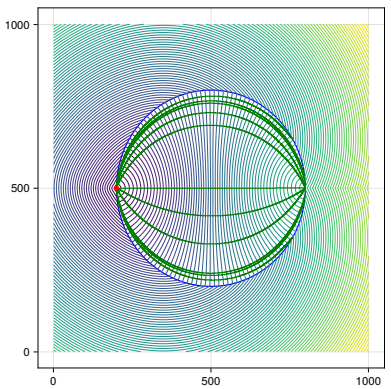

Figure 17. Conditionally correct image of rays of Maxwell lens

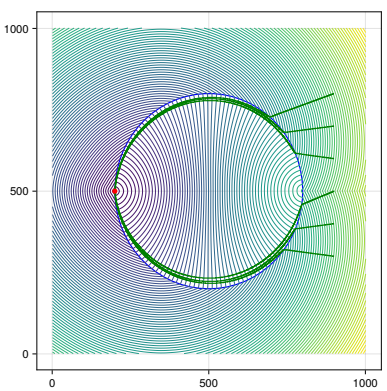

Figure 18. Error image of rays of Maxwell lens modeled by FSM method

5. Discussion

When solving the eikonal equation by the method of characteristics, it is reduced to a system of ODEs. Each solution of the system gives the trajectory of one ray. To obtain a beam of rays, it is necessary to repeatedly solve the obtained ODE system, changing each time the initial values. It can be said that this approach corresponds more to geometrical optics than to wave optics. It is not possible to calculate the points of wave fronts directly by the method of characteristics and requires additional manipulation with the coordinates of the points of the ray trajectory.

In contrast, the FSM method works directly with the Eikonal equation and does not require its transformation into any other form. It also does not require calculations of the derivatives of the refractive index function $n(\mathbf{x})$. The result of the method are approximated values of the function $u(\mathbf{x})$ for all grid points. Having these values, it is rather easy to visualize fronts by depicting level lines, but the visualization of rays has the problems described above.

Another feature of the FSM method is the fact that the electromagnetic field is initially assumed to be present at every point in the modeled region. As shown in figures 10 and 11 when interpreting electromagnetic radiation as rays, there are regions where the radiation does not penetrate.

6. Conclusion

The application of the FSM method for the numerical solution of the eikonal equation has been considered on the example of three well-studied lenses. The following points can be emphasized as advantages of the FSM method.

- It is not necessary to convert the eikonal equation to any other form.
- It is not necessary to find the derivatives of the refractive index function $n(\mathbf{x})$.
- Electromagnetic wave fronts can be constructed directly from the numerical solution.

Difficulties in ray construction and inconsistency with geometrical optics may be pointed out as a disadvantage, which was mentioned in the Discussion section 5.

Funding: This publication has been supported by the RUDN University Scientific Projects Grant System, project No 021934-0-000 (recipients Anna V. Korolkova; Migran N. Gevorkyan) and has been supported by the RUDN University Strategic Academic Leadership Program (recipient Dmitry S. Kulyabov; Arseny V. Fedorov; Christina A. Stepa).

References

1. Zhao, H. A fast sweeping method for Eikonal equations. *Mathematics of Computation* **74,** 603–627. doi:10.1090/s0025-5718-04-01678-3 (May 2004).
2. Gremaud, P. A. & Kuster, C. M. Computational study of fast methods for the eikonal equation. *SIAM Journal on Scientific Computing* **27,** 1803–1816. doi:10.1137/040605655 (Jan. 2006).
3. Jeong, W. & Whitaker, R. A fast eikonal equation solver for parallel systems. *SIAM conference* **84112,** 1–4 (2007).
4. Kulyabov, D. S., Gevorkyan, M. N. & Korolkova, A. V. *Software Implementation of the Eikonal Equation* in *Proceedings of the Selected Papers of the 8th International Conference "Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems" (ITTMM-2018), Moscow, Russia, April 16, 2018* (eds Kulyabov, D. S., Samouylov, K. E. & Sevastianov, L. A.) **2177** (Moscow, Apr. 2018), 25–32.
5. Kulyabov, D. S., Korolkova, A. V., Velieva, T. R. & Gevorkyan, M. N. *Numerical analysis of eikonal equation* in *Saratov Fall Meeting 2018: Laser Physics, Photonic Technologies, and Molecular Modeling* (ed Derbov, V. L.) **11066** (SPIE, Saratov, June 2019), 56. doi:10.1117/12.2525142. arXiv: 1906.09467.
6. Févotte, F. *Fast Sweeping and Fast Marching methods for the solution of eikonal equations* version 0.2.0. https://github.com/triscale-innov/Eikonal.jl (2023).
7. Lauwens, B. & Downey, A. *Think Julia* How to Think Like a Computer Scientist. 229 pp. (O'Reilly Media, Inc., 2019).
8. Born, M. & Wolf, E. *Principles of optics* 7th. 952 pp. (Cambridge University Press, 1999).
9. Fedorov, A. V., Stepa, C. A., Korolkova, A. V., Gevorkyan, M. N. & Kulyabov, D. S. Methodological derivation of the eikonal equation. *Discrete and Continuous Models and Applied Computational Science* **31,** 399–418. doi:10.22363/2658-4670-2023-31-4-399-418 (Dec. 2023).

10. Ivanov, D. I., Ivanov, I. E. & Kryukov, I. A. Hamilton–Jacobi equation-based algorithms for approximate solutions to certain problems in applied geometry. *Computational Mathematics and Mathematical Physics* **45,** 1297–1310 (8 2005).
11. Kabanikhin, S. I. & Krivorotko, O. I. Numerical solution eikonal equation. *Siberian Electronic Mathematical Reports* **10,** 28–34 (2013).
12. Fonseca, N. J. G., Tyc, T. & Quevedo–Teruel, O. A solution to the complement of the generalized Luneburg lens problem. *Communications Physics* **4.** doi:10.1038/s42005-021-00774-2 (2021).
13. Abbasi, M. A. B. & Fusco, V. F. Maxwell fisheye lens based retrodirective array. *Scientific Reports* **9.** doi:10.1038/s41598-019-52779-1 (Nov. 2019).
14. Zeng, Y. & Werner, D. H. Two-dimensional inside-out Eaton Lens. Design technique and TM-polarized wave properties. *Optical Express* **20,** 2335–2345. doi:10.1364/OE.20.002335 (Jan. 2012).
15. Gevorkyan, M. N., Kulyabov, D. S. & Sevastyanov, L. A. *Review of Julia programming language for scientific computing* in *The 6th International Conference "Distributed Computing and Grid-technologies in Science and Education"* (2014), 27.
16. Phillips, L. *Practical Julia. A hands-on introduction for scientific minds* 528 pp. (No Starch Press, Oct. 31, 2023).

To cite: Stepa C. A., Fedorov A. V., Gevorkyan M. N., Korolkova A. V., Kulyabov D. S., Solving the eikonal equation by the FSM method in Julia language, Discrete and Continuous Models and Applied Computational Science 32 (1)(2024)48–60.DOI: 10.22363/2658-4670-2024-32-1-48-60.

Information about the authors

Stepa, Christina A. (Russian Federation)—PhD student of Probability Theory and Cyber Security of Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University) (e-mail: 1042210111@pfur.ru, phone: +7 (495) 952-02-50, ORCID: https://orcid.org/0000-0002-4092-4326, ResearcherID: GLS-1445-2022)

Fedorov, Arseny V. (Russian Federation)—PhD student of Probability Theory and Cyber Security of Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University) (e-mail: 1042210107@rudn.ru, phone: +7 (495) 955-09-27, ORCID: https://orcid.org/0000-0002-3036-0117, ResearcherID: AGY-9849-2022, Scopus Author ID: 57219092618)

Gevorkyan, Migran N. (Russian Federation)—Candidate of Sciences in Physics and Mathematics, Associate Professor of Department of Probability Theory and Cyber Security of Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University) (e-mail: gevorkyan-mn@rudn.ru, phone: +7 (495) 955-09-27, ORCID: https://orcid.org/0000-0002-4834-4895, ResearcherID: E-9214-2016, Scopus Author ID: 57190004380)

Korolkova, Anna V. (Russian Federation)—Docent, Candidate of Sciences in Physics and Mathematics, Associate Professor of Department of Probability Theory and Cyber Security of Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University) (e-mail: korolkova-av@rudn.ru, phone: +7(495) 952-02-50, ORCID: https://orcid.org/0000-0001-7141-7610, ResearcherID: I-3191-2013, Scopus Author ID: 36968057600)

Kulyabov, Dmitry S. (Russian Federation)—Professor, Doctor of Sciences in Physics and Mathematics, Professor of the Department of Probability Theory and Cyber Security of Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University); Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: kulyabov-ds@rudn.ru, phone: +7 (495) 952-02-50, ORCID: https://orcid.org/0000-0002-0877-7063, ResearcherID: I-3183-2013, Scopus Author ID: 35194130800)

УДК 519.65:519.217
PACS 07.05.Tp, 02.70.−c
DOI: 10.22363/2658-4670-2024-32-1-48-60 EDN: HEYUGO

Решение уравнения эйконала методом FSM на языке Julia

К. А. Штепа[1], А. В. Фёдоров[1], М. Н. Геворкян[1], А. В. Королькова[1], Д. С. Кулябов[1,2]

[1] *Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация*
[2] *Объединённый институт ядерных исследований, ул. Жолио-Кюри, д. 6, Дубна, 141980, Российская Федерация*

Аннотация. Существует два основных подхода к численному решению уравнения эйконала: сведение его к системе ОДУ (метод характеристик) и конструирование специализированных методов для численного решения данного уравнения в виде дифференциального уравнения в частных производных. К последнему подходу относится метод FSM (Fast sweeping method). Резонно предположить, что специализированный метод должен обладать большей универсальностью. Цель данной работы — оценка применимости метода FSM для построения лучей и фронтов. Использовалась реализация метода FSM в библиотеке Eikonal языка программирования Julia. Метод применялся для численного моделирования сферических линз Максвелла, Люнеберга и Итона. Данные линзы были выбраны так как их оптические свойства хорошо изучены. Был выбран частный случай плоских линз, как наиболее простых для визуализации и интерпретации результатов. Результаты вычислений представлены в виде изображений фронтов и лучей для каждой из линз. Из анализа полученных изображений сделан вывод, что метод FSM хорошо подходит для построения фронтов электромагнитных волн. Попытка же по результатам его работы визуализировать траектории лучей наталкивается на ряд трудностей и в некоторых случаях дает неправильную визуальную картину.

Ключевые слова: уравнение эйконала, геометрическая оптика, волновая оптика, Julia language, FSM