Numerical Approximation of the Quantum Scattering Problem with the Finite Element Method

Máni Maríus Viðarsson

Faculty of Physical Sciences
University of Iceland
2014

NUMERICAL APPROXIMATION OF THE QUANTUM SCATTERING PROBLEM WITH THE FINITE ELEMENT METHOD

Máni Marius Viðarsson

60 ECTS thesis submitted in partial fulfillment of a
Magister Scientiarum degree in Mathematics

Advisors
Ragnar Sigurðsson
Sigurður Ingi Erlingsson

External Examiner
Sven Þ. Sigurðsson

Faculty of Physical Sciences
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, May 2014

Numerical Approximation of the Quantum Scattering Problem with the Finite Element Method
Approximation of Quantum Scattering with the FEM
60 ECTS thesis submitted in partial fulfillment of a M.Sc. degree in Mathematics

Faculty of Physical Sciences
School of Engineering and Natural Sciences
University of Iceland
VRII, Hjardarhagi 2-6
107, Reykjavik, Reykjavik
Iceland

Telephone: 525 4000

Bibliographic information:
Máni Marius Viðarsson, 2014, Numerical Approximation of the Quantum Scattering Problem with the Finite Element Method, M.Sc. thesis, Faculty of Physical Sciences, University of Iceland.

Printing: Háskólaprent, Fálkagata 2, 107 Reykjavík
Reykjavik, Iceland, May 2014

Abstract

In this thesis we deal with computation of the conductance in an electron transport through a semiconductor nanostructures system. In physics a method often used is the recursive Green function method (RGFM) which is applied to this problem as an approximation of the Green function for the Schrödinger equation in two spatial dimensions. The simplest approach is based on the finite difference method and it gives a simple block structure for the straight infinite-wire system to be solved. We implemented a finite element method (FEM), using both sequential and parallel computation, as well as a new grid refinement algorithm. The FEM is commonly used in applied mathematics for the solution of various partial differential equations. FEM is known for flexibility and adaptability and thus a helpful tool in minimizing computation time for complex problems.

Útdráttur

Þessi ritgerð fjallar um stærðfræðilegt líkan af rafeindaflutningi í örsmáum hálfleiðarakerfum. Algengasta aðferð sem notuð er í eðlisfræði við slíka útreikninga nefnist endurkvæm aðferð fyrir föll Greens og byggir sú aðferð á nálgun með aðferð endanlegra mismuna. Samhliða þeirri aðferð er fjallað um bútaaðferð til að reikna út leiðni fyrir sömu kerfi. Bútaaðferðin hefur ýmsa kosti umfram aðferð endanlegra mismuna, svo sem að einfaldara er að beita henni á kerfi með flókna lögun. Aðferðirnar nýta samhliða reikninga þegar kostur er. Með því móti eru nákvæmar lausnir reiknaðar á skemmri tíma. Sett er fram algrím til þess að fínskipta svæðum í þríhyrninganet og stjórna þar með nákvæmni útreikninganna.

Contents

List of Figures

Acknowledgments

First of all I would like to thank my advisor Prof. Ragnar Sigurðsson for his guidance, support and patience. It has been invaluable. I also would like to express my gratitude towards Prof. Sigurður Ingi Erlingsson for all his help, assistance and cooperation along with Gunnar Þorgilsson. Finally I would like to thank my parents, girlfriend and friends for technical and moral backing.

1. Introduction

Modern technology relies more and more on the use of semiconductor nanostructures and according to experts this trend will continue in the near-future. Gordon E. Moore in 1965 famously predicted [16] that the number of semiconductor components in computer hardware would double on integrated circuits every two years and this prediction has held to the present day. This resulted in an ever evolving arms race between technology companies to create smaller and smaller circuits. Therefore it is no surprise that the modeling of such systems is of great importance.
The theory behind such a system has been extensively studied by numerous researchers and relies on one of the cornerstones of modern science, quantum mechanics, with entire textbooks dedicated to this subject [5, 12]. However, commonly the results obtained from the theoretical models stem from a finite difference method used to approximate the solution. In this thesis we will detail the commonly used recursive Green function method (RGFM) while developing a finite element method (FEM). Generally, the finite element method is known for flexibility and adaptability and is thus a helpful tool in minimizing computation time for complex problems.
We want a model that can describe a particle traveling in a wire, then entering a domain of interest, the scattering domain, and either being reflected by it and leaving the same way it came or successfully navigating the domain and leaving in a different direction. Even though this type of model seems relatively simplistic it can help in describing numerous physical systems [13].
Quantum mechanics describes the time evolution of a system with the solution of the Schrödinger equation. In this thesis we study the problem of approximating the Green function for the time-independent non-relativistic Schrödinger equation for a single particle in an electrostatic potential,

$$\left(E + \frac{\hbar^2}{2m}\Delta - V(r)\right) G(r, r'; E) = \delta(r - r'), \tag{1.1}$$

where m is the mass of the particle, E its energy, $\hbar$ is the reduced Planck constant and $V(r)$ is the energy of some potential in the domain Ω in $\mathbb{R}^2$. This form of the problem will be used throughout the thesis.
Mathematically, the solution of this type of partial differential equation is called the Green function for the Helmholtz operator and has been studied in many fields, e.g. in electromagnetic radiation, seismology, and acoustics.

In Chapter 2 we begin by describing the general properties of the problem and model the domain as an infinite two-dimensional nanodevice. The domain Ω is of a special character. It is a bounded domain, possibly with holes, linked with two semi-infinite strips as shown in Figure 1.1.

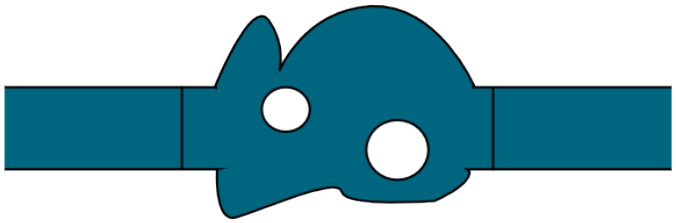

Figure 1.1: An example of domains used in this thesis. The strips extend infinitely to the left and right.

While the strips have uniform width, the shape of the central region can take any form.
The solution to (1.1) in the isolated strips with $V = 0$ is known analytically. After the strips have been added to the central domain, Ω_S, the solution in the strips asymptotically approaches the known analytical solution far away from the central domain. In physics this approach is known as a scattering boundary condition describing an open system.
We will derive some analytical solutions before modeling two distinct numerical methods in Chapter 3, a finite element method, and a finite differences method for comparison. Next we discuss a special grid refinement algorithm for the finite element method developed for this system.

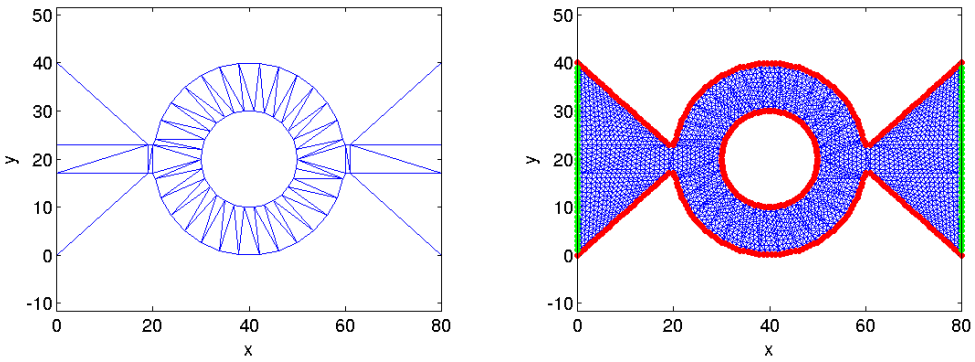

Figure 1.2: Coarse (left) and fine (right) triangular meshes of an annulus-shaped scattering domain. The right image uses the net refinement algorithm from section 3.2. The red dots indicate where the boundary values are 0 and the green indicate where the leads connect.

Thereafter we will use the methods developed to approximate solutions of the Green function $G(r, r'; E)$ for domains of different shapes and different potential energies $V(r)$.

Finally, in Chapter 4 we use these approximations to calculate the so-called transmission function which can give the probability of a particle entering the system on one side going all the way through to the other side.

The novelty of our approach lies in comparing the finite element method to the well established and popular recursive Green function method by highlighting the strengths and weaknesses of each method. Also, by using the FEM on an annular domain with complex geometry we were able to focus on some of the more intricate behavior of the solution. Finally, using our new grid refinement algorithm allowed us to precisely control the hardware resources needed for computation.

2. Theoretical description of the problem

2.1. General properties

Quantum mechanics describes the time evolution of the wave function $\Psi(r,t)$ of a single particle with mass m, in an electrostatic potential $V(r,t)$ with the non-relativistic Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\Psi(r,t) = \left(\frac{-\hbar^2}{2m}\Delta + V(r,t)\right)\Psi(r,t), \tag{2.1}$$

on some domain $\Omega \subset \mathbb{R}^2$ with specified boundary conditions. The wave function can then be used to calculate the probability of finding the particle at a certain position at a given time together with other measurable quantities. In this thesis we shall focus on electric fields which do not vary in time, i.e. $V(r,t) = V(r)$.
Equivalently we can solve

$$i\hbar\frac{\partial U}{\partial t}(r,r',t,t') + \frac{\hbar^2}{2m}\Delta U(r,r',t,t') - V(r)U(r,r',t,t') = \delta(r-r')\delta(t-t') \tag{2.2}$$

for a Green function $U(r,r',t,t')$, from which we can extract these quantities of interest directly. Since the coefficients and the potential V only depend on r, we have

$$U(r,r',t,t') = U(r,r',t-t',0), \tag{2.3}$$

thus there is no need to introduce t' as a separate variable. By using the following definition for a Fourier transform

$$\mathcal{F}(f)(\omega) = \int_{-\infty}^{\infty} e^{i\omega t} f(t)dt, \tag{2.4}$$

we have $\mathcal{F}\left(\frac{\partial}{\partial t}U\right)(\omega) = -i\omega\mathcal{F}(U)(\omega)$. We can apply the Fourier transform to (2.2) and define $G(r,r';E) \equiv \mathcal{F}(U)(r,r',\frac{E}{\hbar},0)$, where $E = \hbar\omega$. Thus (2.2) is equivalent to

$$\left(E + \frac{\hbar^2}{2m}\Delta - V(r)\right)G(r,r';E) = \left(E - \hat{H}\right)G(r,r';E) = \delta(r-r'), \qquad r,r' \in \Omega, \tag{2.5}$$

where $\hat{H}$ is the so-called Hamiltonian operator, with $G(r, r'; E) = 0$ if r or r' are on the boundary $\partial\Omega$.
Given a solution $G(r, r'; E)$ we can thus calculate variables of interest such as the total conductance of the system and other observables.

2.2. Description of the domain

We want our model of a 2D-domain $\Omega \subset \mathbb{R}^2$ to extend infinitely along one axis and to be bounded along the other axis. In order to describe scattering from a localized potential in the wire, we divide our domain Ω into three sub-domains: $\Omega_L = [-\infty, x_L[\times[0, L]$, Ω_S, and $\Omega_R =]x_R, \infty] \times [0, L]$ for some $L > 0$ and distinct points x_L and x_R where $x_L < x_R$.
The middle domain containing the potential is the central system where the scat-

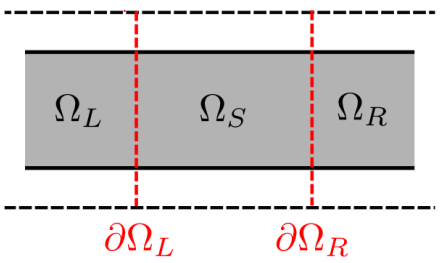

Figure 2.1: The domain and sub-domains.

tering takes place, and the other two domains represent the leads to the central system, where the asymptotic properties of the solution are known. Generally we allow Ω_S to be of any shape as long as it is connected and bounded along the y-axis.

With the domain well defined we can now restate (1.1) with boundary conditions as

$$\begin{aligned} \left(\frac{\hbar^2}{2m}\Delta + E - V(r)\right) G(r, r'; E) &= \delta(r - r'), \\ G(r, r'; E) &= 0 \quad \text{if} \quad r \in \partial\Omega, \\ G(r, r'; E) \text{ bounded as } & \quad r \to \pm\infty, \end{aligned} \tag{2.6}$$

for some $r' \in \Omega$. The last boundary condition is supplied since if we do not set any reasonable growth constraints when x tends to $\pm\infty$, we could get solutions that don't resemble any physical phenomena. We want to describe a wave coming along one lead towards the central system and being scattered by it.

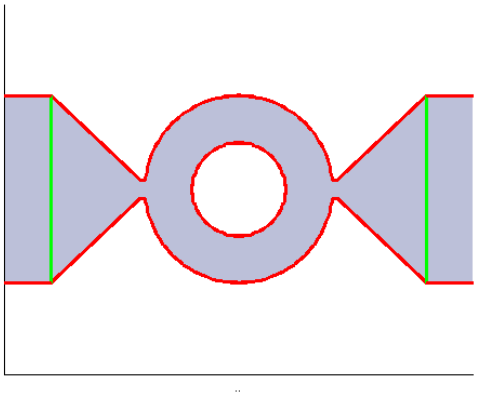

Figure 2.2: An example of a scattering region Ω_S, where the boundary $\partial\Omega_L$ and $\partial\Omega_R$ is green and the boundary where $G(r, r') = 0$ is red.

2.3. Analytical solutions

In general, it is not possible to give an analytical solution of (2.6). We will now study the infinite strip and the semi-infinite strip.

2.3.1. Green function in an unbounded domain

First, we start by solving

$$\left(\frac{\hbar^2}{2m}\Delta + E\right) G(r, r'; E) = \delta(r - r') \tag{2.7}$$

on the whole domain $\mathbb{R}^2$ with only the reasonable growth constraint at infinity as a boundary condition. It can be shown (see Appendix A) that the solution is

$$G(r, r'; E) = \frac{2m}{\hbar^2}\frac{1}{4} Y_0\left(\frac{\|r - r'\|}{\sqrt{\frac{2mE}{\hbar^2}}}\right), \tag{2.8}$$

where Y_0 is the Bessel function of the second kind. Note that in physics, if we are only interested in the outgoing Green function emanating from the point r', we additionally use the Sommerfeld radiation condition [15, p. 12] and thus the solution becomes [7]

$$G(r, r'; E) = -\frac{2m}{\hbar^2}\frac{i}{4} H_0^{(1)}\left(\frac{\|r - r'\|}{\sqrt{\frac{2mE}{\hbar^2}}}\right), \tag{2.9}$$

where $H_0^{(1)}$ is the first Hankel function.
Knowing this fundamental solution we can now construct the solution to (2.6) on Ω with Dirichlet boundary conditions by adding a function $u(r, r')$ to $G(r, r')$ which satisfies the homogeneous problem with boundary conditions $u(r, r') = -G(r, r')$ on $\partial\Omega$. However, rather than solving the Dirichlet problem we derive the Green function in the form of a Fourier series for the infinite and semi-infinite strip.

2.3.2. Green function of an infinite wire

Let us now turn our focus to a system where Ω_S is just a straight segment between Ω_L and Ω_R, in other words $\Omega_S = [x_L, x_R] \times [0, L]$ with $V(r) = 0\ \forall r \in \Omega$. To simplify notation we fix $g(x, y)$ as $G(r, r'; E)$ for some r' and E.
We start by solving

$$\begin{aligned} \frac{\partial^2 g}{\partial x^2}(x,y) + \frac{\partial^2 g}{\partial y^2}(x,y) + \frac{2m}{\hbar^2} E g(x,y) &= \frac{2m}{\hbar^2}\delta(x-x')\delta(y-y') \\ g(x,0) = g(x,L) = 0 \qquad &\text{and} \qquad \|g(x,y)\| < \infty \end{aligned} \tag{2.10}$$

for some function $g(x, y)$. Since $g(x, 0) = g(x, L) = 0$ we can write g as a Fourier sine-series

$$g(x,y) = \sum_{n=1}^{\infty} g_n(x) \sin\left(\frac{n\pi}{L} y\right) \tag{2.11}$$

with some undetermined coefficients $g_n(x)$. Inserting this form back into (2.10) results in

$$\sum_{n=1}^{\infty} \left(g_n''(x) - \left(\frac{n\pi}{L}\right)^2 g_n(x) + \frac{2m}{\hbar^2} E g_n(x) \right) \sin\left(\frac{n\pi}{L} y\right) = \frac{2m}{\hbar^2}\delta(x-x')\delta(y-y'). \tag{2.12}$$

Here each coefficient in the above series should satisfy the following equation

$$\begin{aligned} \left(g_n''(x) - \left(\frac{n\pi}{L}\right)^2 g_n(x) + \frac{2m}{\hbar^2} E g_n(x) \right) &= \frac{2m}{\hbar^2}\delta(x-x')\frac{2}{L}\int_0^L \sin\left(\frac{n\pi}{L} y\right)\delta(y-y')dy \\ &= \frac{2}{L}\frac{2m}{\hbar^2}\delta(x-x')\sin\left(\frac{n\pi}{L} y'\right). \end{aligned} \tag{2.13}$$

We define $k_n \equiv \sqrt{\frac{2mE}{\hbar^2} - \frac{n^2\pi^2}{L^2}}$. Note that if $\frac{2mE}{\hbar^2} < \frac{n^2\pi^2}{L^2}$ depending on which root of k_n we choose to use, we get two different solutions; the solution associated with the principal root is named the retarded Green function, while the other root yields the advanced Green function. It is however simple to see that these solutions are complex conjugates of each other, thus we choose to exclusively focus on the retarded Green function in this thesis. Also note that if $k_n = 0$, the solution is

$$g_n(x) = \frac{2}{L}\frac{2m}{\hbar^2}\sin\left(\frac{n\pi}{L} y'\right)(x-x')H(x-x') + Ax + B \qquad A, B \in \mathbb{C}, \tag{2.14}$$

where H is the Heaviside step function, but since this solution blows up as $x \to \pm\infty$ we shall disregard it. The solutions of (2.13) are thus of the form

$$g_n(x) = \begin{cases} Ae^{ik_n x} + Be^{-ik_n x} & x \le x' \\ Ce^{ik_n x} + De^{-ik_n x} & x \ge x', \end{cases} \tag{2.15}$$

where $A, B, C, D \in \mathbb{C}$. Now for any x', if $x \to \infty$, $g_n(x)$ blows up unless $D = 0$, and equivalently when $x \to -\infty$ forces $A = 0$. Since $g_n(x)$ is continuous at $x = x'$ and its derivative should have a jump there equal to $\frac{2}{L}\frac{2m}{\hbar^2}\sin\left(\frac{n\pi}{L}y'\right)$, we get the following system of equations

$$\begin{aligned} Be^{-ik_n x'} - Ce^{ik_n x'} &= 0 \\ ik_n Be^{-ik_n x'} + ik_n Ce^{ik_n x'} &= \frac{2}{L}\frac{2m}{\hbar^2}\sin\left(\frac{n\pi}{L}y'\right). \end{aligned} \tag{2.16}$$

Solving for B and C reveals that

$$g_n(x) = \begin{cases} \frac{1}{ik_n}\frac{2m}{\hbar^2}\frac{1}{L}e^{ik_n x'}\sin\left(\frac{n\pi}{L}y'\right)e^{-ik_n x} & x \le x' \\ \frac{1}{ik_n}\frac{2m}{\hbar^2}\frac{1}{L}e^{-ik_n x'}\sin\left(\frac{n\pi}{L}y'\right)e^{ik_n x} & x \ge x' \end{cases} \tag{2.17}$$

and therefore we have

$$G(r, r'; E) = g(x, y) = \frac{2m}{\hbar^2}\sum_{n=1}^{\infty}\frac{e^{ik_n|x-x'|}}{ik_n}\frac{1}{L}\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right). \tag{2.18}$$

Note that if $k_n = 0$ for some n, we skip the corresponding term in the sum.
Since we supplied the additional boundary condition in (2.6) that $G(r, r'; E)$ should be bounded as $r \to \pm\infty$, our solution is uniquely determined. Let us now prove that (2.18) is indeed a solution of (2.6). Take $\phi \in C_0^\infty(\Omega)$. We can again write ϕ as a Fourier sine-series

$$\phi(x, y) = \sum_{n=1}^{\infty}\phi_n(x)\sin\left(\frac{n\pi}{L}y\right). \tag{2.19}$$

Now suppose that $r' = (x', y') = (0, y')$ then

$$\begin{aligned} \phi(0, y') &= \langle\delta(x)\delta(y - y'), \phi(x, y)\rangle = \left\langle\left(\frac{\hbar^2}{2m}\Delta + E\right)G, \phi(x, y)\right\rangle \\ &= \left\langle G, \left(\frac{\hbar^2}{2m}\Delta + E\right)\phi(x, y)\right\rangle \\ &= \int_{-\infty}^{\infty}\int_0^L\left(\frac{2m}{\hbar^2}\sum_{j=1}^{\infty}\frac{e^{ik_j|x|}}{ik_j}\frac{1}{L}\sin\left(\frac{j\pi y}{L}\right)\sin\left(\frac{j\pi y'}{L}\right)\right) \\ &\times\left(\sum_{n=1}^{\infty}\left(\frac{\hbar^2}{2m}\phi_n''(x) + (E - \frac{\hbar^2}{2m}\frac{n^2\pi^2}{L^2})\phi_n(x)\right)\sin\left(\frac{n\pi y}{L}\right)\right)dxdy. \end{aligned} \tag{2.20}$$

Since $\int_0^L \sin(\frac{j\pi}{L}y)\ \sin(\frac{n\pi}{L}y)dy = \frac{L}{2}\delta_{jn}$, we can simplify the previous equation to

$$= \frac{2m}{\hbar^2}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi y'}{L}\right)\frac{1}{2}\int_{-\infty}^{\infty}\frac{e^{ik_n|x|}}{ik_n}\left(\frac{\hbar^2}{2m}\phi_n''(x) + (E - \frac{\hbar^2}{2m}\frac{n^2\pi^2}{L^2})\phi_n(x)\right)dx. \quad (2.21)$$

By using integration by parts twice on the first part of the integral, it becomes

$$\begin{aligned}
&\int_{-\infty}^{\infty}\frac{e^{ik_n|x|}}{ik_n}\frac{\hbar^2}{2m}\phi_n''(x)dx \\
&\quad = -\int_{-\infty}^{\infty}\mathrm{sign}(x)e^{ik_n|x|}\frac{\hbar^2}{2m}\phi_n'(x)dx \\
&\quad = \left[-e^{ik_n|x|}\frac{\hbar^2}{2m}\phi_n(x)\right]_{x\to 0}^{x\to\infty} + \left[e^{ik_n|x|}\frac{\hbar^2}{2m}\phi_n(x)\right]_{x\to-\infty}^{x\to 0} + \int_{-\infty}^{\infty}ik_n e^{ik_n|x|}\frac{\hbar^2}{2m}\phi_n(x)dx \\
&\quad = \frac{\hbar^2}{2m}2\phi_n(0) + \int_{-\infty}^{\infty}\frac{e^{ik_n|x|}}{ik_n}(ik_n)^2\frac{\hbar^2}{2m}\phi_n(x)dx
\end{aligned} \quad (2.22)$$

Now substituting this back into (2.21), we get

$$\begin{aligned}
&\frac{2m}{\hbar^2}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi y'}{L}\right) \\
&\quad\times\frac{1}{2}\left(\frac{\hbar^2}{2m}2\phi_n(0) + \int_{-\infty}^{\infty}\frac{e^{ik_n|x|}}{ik_n}\underbrace{\left(-\frac{\hbar^2}{2m}k_n^2 + (E - \frac{\hbar^2}{2m}\frac{n^2\pi^2}{L^2})\right)}_{=0}\phi_n(x)dx\right) \quad (2.23) \\
&\quad = \sum_{n=1}^{\infty}\phi_n(0)\sin\left(\frac{n\pi y'}{L}\right) = \phi(0,y'),
\end{aligned}$$

Thus we have that (2.18) is a solution of (2.6).

2.3.3. Green function of a semi-infinite wire

Now we turn our focus to the left semi-infinite wire, that is, the system

$$\begin{aligned}
\left(\frac{\hbar^2}{2m}\Delta + E - V(r)\right)G(r,r';E) &= \delta(r-r'), \qquad r,r'\in\Omega_L, \\
G(x,0,x',y') = G(x,L,x',y') = G(x,y,x',0) = G(x,y,x',L) &= 0, \\
G(x_L,y,x',y') = G(x,y,x_L,y') &= 0, \\
r\mapsto G(r,r';E)\ \text{bounded as}\ r\to-\infty.&
\end{aligned} \quad (2.24)$$

Since we have already deduced the solution to the infinite wire in the previous section, we determine the solution to this system by applying the method of images

[18, p. 191] as

$$g_L(x,y,x',y') = \frac{2m}{\hbar^2}\sum_{n=1}^{\infty}\frac{\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right)}{ik_nL}\left(e^{ik_n|x-x'|}-e^{ik_n(2x_L-x-x')}\right). \quad (2.25)$$

Equivalently we can solve the semi-infinite right system as

$$g_R(x,y,x',y') = \frac{2m}{\hbar^2}\sum_{n=1}^{\infty}\frac{\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right)}{ik_nL}\left(e^{ik_n|x-x'|}-e^{ik_n(x+x'-2x_R)}\right). \quad (2.26)$$

2.4. Transmission function

Given that we have a solution $G(r,r';E)$ to (2.6), we can calculate the probability that an electron with energy E in either lead, Ω_L or Ω_R, can travel through the central scattering region Ω_S and into the other lead, by using the transmission function $T(E)$, that is given by [10]

$$\begin{aligned} T(E) = \int_{\partial\Omega_L}\int_{\partial\Omega_L}\int_{\partial\Omega_R}\int_{\partial\Omega_R} &\Gamma_L(r_L,r'_L;E)G(r'_L,r_R;E) \\ &\times\Gamma_R(r_R,r'_R;E)(G(r'_R,r_L;E))^* dr_L dr'_L dr_R dr'_R, \end{aligned} \quad (2.27)$$

where

$$\Gamma_J(r,r';E) = \frac{\hbar^2}{2m}\mathrm{Im}\left(\frac{\partial^2 G(r,r';E)}{\partial n_J \partial n'_J}\right). \quad (2.28)$$

Here n_J denotes the normal vector at $\partial\Omega_J$ pointing out of the area Ω_J for $J=\{L,R\}$. We can define a normalized transmission probability

$$P_T(E) = \begin{cases} 0 & \text{if } E < \frac{\hbar^2}{2m}\frac{n^2\pi^2}{L^2} \\ \frac{T(E)}{n_0(E)} & \text{else,} \end{cases} \quad (2.29)$$

where

$$n_0(E) = \max\left(\left\{n\in\mathbb{N}\middle|\frac{n^2\pi^2}{L^2}\leq\frac{2mE}{\hbar^2}\right\}\right). \quad (2.30)$$

This choice of normalization might seem strange at first, but the physical interpretation is that when n_0 increases by 1, an electron can enter the scattering region in yet another higher energy wave-mode (perpendicular to the transport direction).
The transmission function can also be used to calculate the conductance of the nanodevice by using the Fisher-Lee relation [8].

3. Numerical methods

In this Chapter we will discuss two distinct numerical methods used to solve our system; the finite element method and the recursive Green function method which uses finite differences.

3.1. Weak formulation of the boundary value problem

To formulate our finite element method we use the Galerkin method [1, p. 367]. The following derivation is mostly based on Havu et al. [10].
We want to approximate the solution using a linear combination of piecewise linear functions with local support. In doing so we lose some of the smoothness but gain a numerical approximation which we can calculate using computers. Thus we need to introduce the weak formulation of the boundary value problem. We start by multiplying (2.6) with a sufficiently smooth test function $\psi(r) \in C_0^\infty(\Omega)$ and integrating over the domain Ω_S

$$\int_{\Omega_S} \psi(r) \left(\frac{\hbar^2}{2m}\Delta + E - V(r) \right) G(r,r')dr = \int_{\Omega_S} \psi(r)\delta(r-r')dr = \psi(r'). \tag{3.1}$$

Now we apply Green's first identity [18, p. 178]

$$\begin{aligned} -\frac{\hbar^2}{2m} \int_{\Omega_S} \nabla\psi(r) \cdot \nabla G(r,r')dr + \int_{\Omega_S} (E - V(r))\psi(r)G(r,r')dr \\ + \frac{\hbar^2}{2m} \int_{\partial\Omega_L} \psi(r) \frac{\partial G(r,r')}{\partial n_L} dr + \frac{\hbar^2}{2m} \int_{\partial\Omega_R} \psi(r) \frac{\partial G(r,r')}{\partial n_R} dr \\ = \psi(r'). \end{aligned} \tag{3.2}$$

Here n_L and n_R are normal vectors pointing out of Ω_S. Thus we have circumvented having to evaluate the Laplacian of the Green function. Calculating the directional derivative on the boundary in the latter two integrals can, however, still be problematic. We shall circumnavigate this by constructing a so-called Dirichlet-to-Neumann

map [14, p. 127]. Thus we turn our focus to the left semi-infinite wire on the domain Ω_L with Dirichlet boundary conditions at $\partial\Omega_L$,

$$\begin{aligned}\left(\frac{\hbar^2}{2m}\Delta + E - V(r)\right) g_L(r,s) &= \delta(r-s), && r,s \in \Omega_L \\ g_L(r,s) &= 0 && \text{if } r \in \partial\Omega, \\ g_L(r,s;E) & && \text{bounded as } r \to -\infty.\end{aligned} \tag{3.3}$$

Again we multiply the equation with a test function $u(r)$, integrate over Ω_L and use Green's first identity twice

$$\begin{aligned}u(s) &= \int_{\Omega_L} u(r)\delta(r-s)dr = \int_{\Omega_L} u(r)\left(\frac{\hbar^2}{2m}\Delta + E - V(r)\right) g_L(r,s)dr \\ &= \int_{\Omega_L} u(r)[E-V(r)]g_L(r,s;E)dr - \frac{\hbar^2}{2m}\int_{\Omega_L} \nabla u(r)\cdot\nabla g_L(r,s;E)dr \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_L} u(r_L)\frac{\partial g_L(r_L,s)}{\partial n_L}dr_L \\ &= \int_{\Omega_L} u(r)[E-V(r)]g_L(r,s;E)dr + \frac{\hbar^2}{2m}\int_{\Omega_L}\Delta u(r) g_L(r,s)dr \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_L} u(r_L)\frac{\partial g_L(r_L,s)}{\partial n_L}dr_L - \frac{\hbar^2}{2m}\int_{\partial\Omega_L}\frac{\partial u(r_L)}{\partial n_L} g_L(r_L,s)dr_L.\end{aligned} \tag{3.4}$$

Since $g_L(r_L,s) = 0$ on $\partial\Omega_L$ the last integral is 0. If we set $u(s) = G(s,r')$, where r' is some point in Ω_S, we have

$$\begin{aligned}G(s,r') = &\int_{\Omega_L} g_L(r,s)\left[\frac{\hbar^2}{2m}\Delta + E - V(r)\right] G(r,r')dr \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_L} G(r_L,r')\frac{\partial g_L(r_L,s)}{\partial n_L}dr_L.\end{aligned} \tag{3.5}$$

Here again the first integral is zero since r and r' lie in different domains, Ω_L and Ω_S respectively. Now by differentiating with respect to s, letting $s \to r'_L \in \partial\Omega_L$, multiplying with a test-function $\psi(r'_L)$ and integrating over $\partial\Omega_L$, we obtain

$$\int_{\partial\Omega_L}\frac{\partial G(r'_L,r')}{\partial n'_L}\psi(r'_L)dr'_L = \frac{\hbar^2}{2m}\int_{\partial\Omega_L}\int_{\partial\Omega_L} G(r_L,r')\frac{\partial^2 g_L(r_L,r'_L)}{\partial n_L \partial n'_L}\psi(r'_L)dr_L dr'_L. \tag{3.6}$$

We can repeat this process for the right semi-infinite lead. Then finally substituting these two boundary integrals back into (3.2) we get

$$\begin{aligned}-\frac{\hbar^2}{2m}\int_\Omega \nabla\psi(r)\cdot\nabla G(r,r')dr + &\int_\Omega (E-V(r))\psi(r)G(r,r')dr \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_L}\int_{\partial\Omega_L}\psi(r'_L)\frac{\partial^2 g_L(r_L,r'_L)}{\partial n_L\partial n'_L}G(r_L,r')dr_L dr'_L \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_R}\int_{\partial\Omega_R}\psi(r'_R)\frac{\partial^2 g_R(r_R,r'_R)}{\partial n_R\partial n'_R}G(r_R,r')dr_R dr'_R = \psi(r').\end{aligned} \tag{3.7}$$

Since in this thesis we focus only on systems where $V(r) = 0$ for $r \in \Omega_L$ or $r \in \Omega_R$, we know the analytical solutions g_L and g_R from Chapter 2.3.3. Thus we have successfully transformed our problem (2.6) into a more manageable form, requiring us only to evaluate the gradient of the Green function instead of the Laplacian.
Finally with these prerequisites out of the way we can turn our attention to the numerical implementation.

3.1.1. Finite element method

We choose approximate $G(r, r')$ as a linear combination of basis functions, each with local support,

$$G(r, r') \approx \sum_{j,l=1}^{N} g_{jl}\phi_j(r)\phi_l(r') \qquad \text{for some } g_{jl} \in \mathbb{C}. \tag{3.8}$$

We now discretize Ω_S into a triangular mesh $\mathcal{S}$, e.g. with the algorithm described in section 3.2 such that each boundary $\partial\Omega_L$ and $\partial\Omega_R$ contains M nodes spaced equally with distance a apart. We choose our basis functions to be the standard simple piecewise linear functions [3, p. 56], such that $\phi_j = 0$ everywhere except in the elements that contains node j where $\phi_j(x_j, y_j) = 1$ and goes to zero at every adjacent node. Choosing our basis functions in such a manner ensures that the resulting matrix equation stays relatively sparse.
Now by choosing $\psi(r) = \phi_k(r)$, (3.7) is thus equivalent to

$$\begin{aligned}\sum_{i,j=1}^{N} g_{jl}\phi_l(r')\Bigg[&- \frac{\hbar^2}{2m}\int_\Omega \nabla\phi_k(r)\cdot\nabla\phi_j(r)dr + \int_\Omega (E - V(r))\phi_k(r)\phi_j(r)dr \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_L}\int_{\partial\Omega_L} \phi_k(r'_L)\frac{\partial^2 g_L(r_L, r'_L)}{\partial n_L \partial n'_L}\phi_j(r_L)dr_L dr'_L \\ &+ \frac{\hbar^2}{2m}\int_{\partial\Omega_R}\int_{\partial\Omega_R} \phi_k(r'_R)\frac{\partial^2 g_R(r_R, r'_R)}{\partial n_R \partial n'_R}\phi_j(r_R)dr_R dr'_R\Bigg] = \phi_k(r').\end{aligned} \tag{3.9}$$

Note that since $G(r,r') = G(r',r)$, we also have that $g_{jl} = g_{lj}$ and therefore we can determine the coefficients g_{jl} by inverting the matrix A given by

$$\begin{aligned} A_{jk} = &\underbrace{-\frac{\hbar^2}{2m}\int_\Omega \nabla\phi_k(r)\cdot\nabla\phi_j(r)dr}_{1} + \underbrace{\int_\Omega (E-V(r))\phi_k(r)\phi_j(r)dr}_{2} \\ &+ \underbrace{\frac{\hbar^2}{2m}\int_{\partial\Omega_L}\int_{\partial\Omega_L}\phi_k(r'_L)\frac{\partial^2 g_L(r_L,r'_L)}{\partial n_L \partial n'_L}\phi_j(r_L)dr_L dr'_L}_{3} \\ &+ \underbrace{\frac{\hbar^2}{2m}\int_{\partial\Omega_R}\int_{\partial\Omega_R}\phi_k(r'_R)\frac{\partial^2 g_R(r_R,r'_R)}{\partial n_R \partial n'_R}\phi_j(r_R)dr_R dr'_R}_{4}. \end{aligned} \tag{3.10}$$

Here integral 1 is named the stiffness matrix, integral 2 is the mass matrix and integral 3 and 4 are the boundary matrices.
Let us now calculate the contribution in (3.10) from the nonzero-basis functions on a triangle $\mathcal{T}$ in $\mathcal{S}$, but first we need to clarify our notation. In our setup we follow unpublished lecture notes by Sven Þ. Sigurðsson. A similar derivation can be found in [17, p. 337]. We denote the corners of the triangle $\mathcal{T}$ by A, B and C. Let h_A be the height of the triangle measured from point A to the opposite edge, T be the area of the triangle,ϕ_A, ϕ_B, ϕ_C be the basis functions that take the value 1 at A, B, C respectively, and the vectors $\overrightarrow{l_A} = \overrightarrow{BC}, \overrightarrow{l_B} = \overrightarrow{CA}, \overrightarrow{l_C} = \overrightarrow{AB}$.

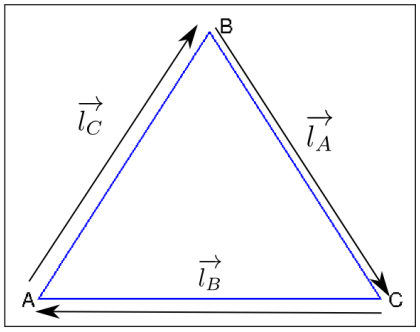

Figure 3.1: A typical triangular $\mathcal{T}$ element in $\mathcal{S}$.

Stiffness matrix

We start by evaluating the stiffness matrix

$$-\frac{\hbar^2}{2m}\int_{\mathcal{T}} \nabla\phi_k(r)\cdot\nabla\phi_j(r)dr. \tag{3.11}$$

The gradient of the basis function can be denoted as

$$\nabla\phi_j = -\frac{1}{h_j}\frac{1}{\|\overrightarrow{l_j^R}\|}\overrightarrow{l_j^R} = -\frac{1}{2T}\overrightarrow{l_j^R} \qquad j = A, B, C. \tag{3.12}$$

Here $\overrightarrow{l_j^R}$ is the vector $\overrightarrow{l_j}$ turned 90° so that it points outside of the triangle. Thus the contribution from two basis functions on element $\mathcal{T}$ of the first integral of (3.10) simply becomes

$$\begin{aligned} -\frac{\hbar^2}{2m}\int_{\mathcal{T}} \nabla\phi_k(r)\cdot\nabla\phi_j(r)dr &= -\frac{\hbar^2}{2m}\int_{\mathcal{T}} \nabla\phi_k(r)\cdot\nabla\phi_j(r)dr \\ &= -\frac{\hbar^2}{2m}\left(T\frac{1}{4T^2}\overrightarrow{l_k^R}\cdot\overrightarrow{l_j^R}\right) = -\frac{\hbar^2}{2m}\frac{1}{4T}\overrightarrow{l_k}\cdot\overrightarrow{l_j}\,. \end{aligned} \tag{3.13}$$

Therefore the contribution from each element in $\mathcal{S}$ is

$$-\frac{\hbar^2}{2m}\frac{1}{4T}\begin{bmatrix} \overrightarrow{l_A}\cdot\overrightarrow{l_A} & \overrightarrow{l_B}\cdot\overrightarrow{l_A} & \overrightarrow{l_C}\cdot\overrightarrow{l_A} \\ \overrightarrow{l_A}\cdot\overrightarrow{l_B} & \overrightarrow{l_B}\cdot\overrightarrow{l_B} & \overrightarrow{l_C}\cdot\overrightarrow{l_B} \\ \overrightarrow{l_A}\cdot\overrightarrow{l_C} & \overrightarrow{l_B}\cdot\overrightarrow{l_C} & \overrightarrow{l_C}\cdot\overrightarrow{l_C} \end{bmatrix}. \tag{3.14}$$

Mass matrix

Next we focus on the mass matrix

$$\int_{\mathcal{T}}[E - V(r)]\phi_k(r)\phi_j(r)dr. \tag{3.15}$$

A well known approximation for an integral of some function $u(x, y)$ over a triangle $\mathcal{T}$ is given by

$$\iint_{\mathcal{T}} u(x,y)dxdy \approx \frac{T}{3}\left(u_{AB} + u_{BC} + u_{CA}\right), \tag{3.16}$$

where u_{AB} is the value that u takes on the middle of the edge between A and B. The contribution from two basis functions over element $\mathcal{T}$ from the second integral of (3.10) becomes

$$\begin{aligned} &\int_{\mathcal{T}}[E - V(r)]\phi_A(r)\phi_B(r)dr \\ &\approx \frac{T}{3}\left(\frac{1}{2}\cdot\frac{1}{2}\cdot(E - V_{AB}) + \frac{1}{2}\cdot 0\cdot(E - V_{BC}) + 0\cdot\frac{1}{2}\cdot(E - V_{CA})\right), \end{aligned} \tag{3.17}$$

and

$$\begin{aligned} &\int_{\mathcal{T}}[E - V(r)]\phi_A(r)\phi_A(r)dr \\ &\approx \frac{T}{3}\left(\frac{1}{2}\cdot\frac{1}{2}\cdot(E - V_{AB}) + 0\cdot 0\cdot(E - V_{BC}) + \frac{1}{2}\cdot\frac{1}{2}\cdot(E - V_{CA})\right). \end{aligned} \tag{3.18}$$

So the element-wise contribution becomes

$$\frac{T}{12}\left(E\begin{bmatrix}2&1&1\\1&2&1\\1&1&2\end{bmatrix}-\begin{bmatrix}V_{AB}+V_{CA}&V_{AB}&V_{CA}\\V_{AB}&V_{AB}+V_{BC}&V_{BC}\\V_{CA}&V_{BC}&V_{CA}+V_{BC}\end{bmatrix}\right). \tag{3.19}$$

Boundary matrix

Now all that is left is the boundary matrix

$$\frac{\hbar^2}{2m}\int_{\partial\Omega_L}\int_{\partial\Omega_L}\phi_k(r'_L)\frac{\partial^2 g_L(r_L,r'_L)}{\partial n_L\partial n'_L}\phi_j(r_L)dr_L dr'_L. \tag{3.20}$$

We will only focus on the integral over the left boundary. The case of the right boundary is analogous. Note that we get a nonzero contribution from this boundary integral as long as both functions ϕ_k and ϕ_j take nonzero values on $\partial\Omega_L$, thus it is impractical to look at the contribution from each triangle individually. Suppose that $\phi_k(r'_L)\neq 0$ and $\phi_j(r_L)\neq 0$ for some $r_L, r'_L\in\partial\Omega_L$. Now the derivative of (2.25) with respect to n_L and n'_L, where $x_L=0$ and $x'<x$, becomes

$$\frac{\partial g_L}{\partial n_L}=-\frac{2m}{\hbar^2}\sum_{m=1}^{\infty}\frac{\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right)}{ik_mL}\left((-ik_m)e^{ik_m(x-x')}-(-ik_m)e^{-ik_m(x+x')}\right), \tag{3.21}$$

and

$$\begin{aligned}\frac{\partial^2 g_L}{\partial n_L\partial n'_L}&=\frac{2m}{\hbar^2}\sum_{m=1}^{\infty}\frac{\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right)}{ik_mL}\left((-ik_m)(ik_m)e^{ik_m(x-x')}-(-ik_m)^2e^{-ik_m(x+x')}\right)\\&=\frac{2m}{\hbar^2}\sum_{m=1}^{\infty}-\frac{ik_m}{L}\sin\left(\frac{n\pi}{L}y\right)\sin\left(\frac{n\pi}{L}y'\right)\left(e^{ik_m(x-x')}+e^{-ik_m(x+x')}\right).\end{aligned} \tag{3.22}$$

Since we are currently only working on the boundary $\partial\Omega_L$, we can write the basis functions as

$$\phi_k(x_L,y)=\begin{cases}\frac{1}{a}(y-y_{k-1}) & y\in[y_{k-1},y_k]\\ \frac{1}{a}(y_{k+1}-y) & y\in[y_k,y_{k+1}]\\ 0 \quad \text{else}\end{cases} \tag{3.23}$$

where a is the distance between two nodes on the boundary, $\phi_k(x_L, y_k) = 1$ and $y_{k\pm1} = y_k \pm a$. We can now calculate integral 3 directly for ϕ_k and ϕ_j as

$$\begin{aligned}
&\frac{\hbar^2}{2m}\int_{\Omega_L}\int_{\Omega_L}\phi_k(r'_L)\frac{\partial^2 g_L(r_L, r'_L)}{\partial n_L \partial n'_L}\phi_j(r_L)dr_L dr'_L \\
&= -\sum_{n=1}^{\infty}\frac{2ik_n}{La^2}\left(\frac{L^2}{n^2\pi^2}\left(2\sin\left(\frac{n\pi}{L}y_k\right) - \sin\left(\frac{n\pi}{L}y_{k-1}\right) - \sin\left(\frac{n\pi}{L}y_{k+1}\right)\right)\right) \\
&\times\left(\frac{L^2}{n^2\pi^2}\left(2\sin\left(\frac{n\pi}{L}y_j\right) - \sin\left(\frac{n\pi}{L}y_{j-1}\right) - \sin\left(\frac{n\pi}{L}y_{j+1}\right)\right)\right).
\end{aligned} \tag{3.24}$$

3.2. Grid generation

To discretize the domain Ω_S for the FEM there are many methods available, e.g. making a regular grid with nodes spaced evenly throughout the domain or some Delauney triangulation method [6]. In this thesis we will mainly focus on our own method. First we will give a quick overview over how the method works, followed by a worked example and a detailed listing of the algorithm. Finally appendix B shows a MATLAB implementation.
Start with a coarse triangular mesh $\mathcal{S}_0$ and a function $f(p, \overrightarrow{v})$. This function tells us, for a given node p in the mesh, the distance at which a new node should be added on any edge connected to p. For each triangle in $\mathcal{S}_0$, add new nodes to its edges as in algorithm 2. Then in algorithm 3 we add an edge between the new nodes furthest apart in the edges with the most nodes. We are thus left with a smaller triangle, which we use algorithm 3 on recursively, and a trapezoid which we turn into triangles with algorithm 4.

3.2.1. Worked example

Start with a triangle mesh $\mathcal{S}$ and a positive real function $f((x, y), \overrightarrow{v})$. Select a triangle $\mathcal{T}$ from $\mathcal{S}$ with vertices A, B and C.

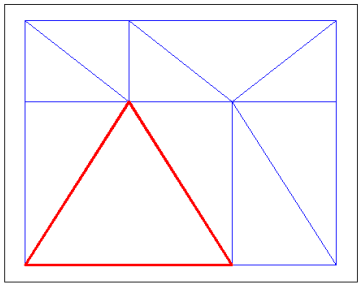

Figure 3.2: A triangular mesh $\mathcal{S}$

Step 1

Select an edge of the triangle $\mathcal{T}$, e.g. edge AB. Denote unit vector $v = \overrightarrow{AB}/\|\overrightarrow{AB}\|$. Evaluate

$$Q_{i+1} = Q_i + \overrightarrow{v} f(Q_i, \overrightarrow{v}) \qquad Q_0 = A.$$

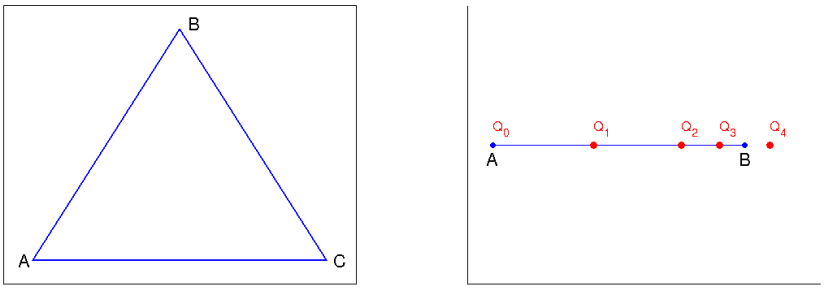

Figure 3.3: A typical triangular element $\mathcal{T}$ and the addition of new nodes to one of it's edges.

Find the Q_i with the lowest index that lands outside the edge AB, Q_n.
Now we add $n-1$ evenly spaced nodes to edge AB and repeat for the other edges of triangle $\mathcal{T}$. This step is implemented in algorithm 2.

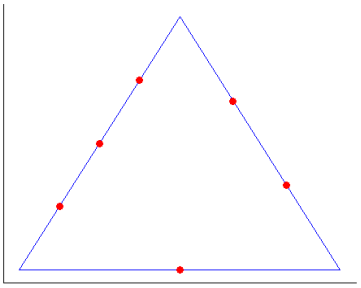

Figure 3.4: Triangle $\mathcal{T}$ with new nodes on it's edges

Step 2

If the triangle $\mathcal{T}$ received no new nodes in step 1, then we are done.
If only one edge received new nodes, connect those to the opposite corner to create smaller triangles.
If neither of the above cases is satisfied, sort the edges from fewest new nodes to the most. For the two edges with the most new nodes, connect the two nodes on opposite edges together that are the furthest away from each other. Create new

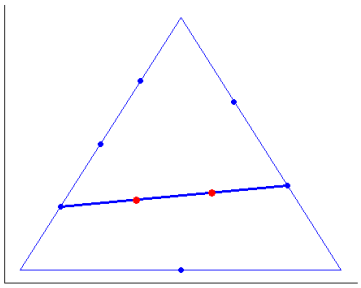

Figure 3.5: New edge added to $\mathcal{T}$.

nodes on this edge with the same method as described in step 1. Now we are left with a smaller triangle and a trapezoid. Repeat step 2 with the smaller triangle, continue onto step 3 with the trapezoid. This step is implemented in algorithm 3.

Step 3

Connect the nodes in the trapezoid such that the edges added are the shortest possible length. This step is finally implemented in algorithm 4

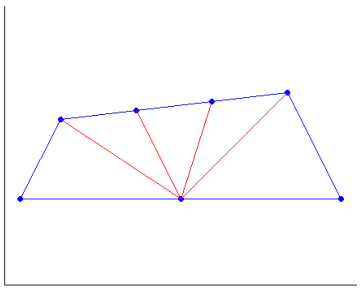

Figure 3.6: The remaining trapezoid triangularized.

Repeat the whole process for each triangle in the mesh $\mathcal{S}$

3.2.2. Algorithm listing

Algorithm 1: `netref`(TRI, f)

Data: TRI triangle list, function f
Result: $nTRI$ new refined triangle list
for $i \leftarrow 1$ **to** $size(TRI)$ **do**
  for $j \leftarrow 1$ **to** 3 **do**
    if *we have not added new nodes to edge* j *of triangle* i *in* TRI **then**
      add new nodes to edge j *in triangle* i *with function* `newnodes` ;
for $i \leftarrow 1$ **to** $size(TRI)$ **do**
  Let a, b, c *be the edges of triangle* i *in* TRI *such that* c *contains the fewest nodes while* $[a, b, c]$ *maintains the original ordering (clockwise or counter-clockwise) of* TRI ;
  $tTRI \leftarrow$`netrefrec(`a,b,c`)`;
  add $tTRI$ *to* $nTRI$;

Algorithm 2: `newnodes`(p_1, p_2)

Data: p_1 and p_2 endpoints of an edge, function $f(p, v)$
Result: nE edge with new nodes
$v \leftarrow \frac{p_2 - p_1}{\|p_2 - p_1\|}$;
$q \leftarrow p_1$;
$qi \leftarrow 0$;
while q *between* p_1 *and* p_2 **do**
  $q \leftarrow q + v \cdot f(q, v)$;
  $qi++$;
$q \leftarrow p_2$;
$rqi \leftarrow 0$;
while q *between* p_1 *and* p_2 **do**
  $q \leftarrow q - v \cdot f(q, -v)$;
  $rqi++$;
add $(\max(qi, rqi) - 1)$ *evenly spaced nodes between* p_1 *and* p_2 *to* nE;

Algorithm 3: `netrefrec`(a, b, c, f)

Data: a, b, c edges of a triangle, function f
Result: $tTRI$ triangle list
if *at least two edges contain only 2 nodes* **then**
  if $size(a) = size(b) = size(c) = 2$ **then**
    $tTRI \leftarrow [a, b, c]$;
  else
    if $size(a) > size(b)$ **then**
      for $j \leftarrow 1$ **to** $size(a) - 1$ **do**
        add $[a(j), a(j+1), b(2)]$ to $tTRI$;
    else
      for $j \leftarrow 1$ **to** $size(a) - 1$ **do**
        add $[a(j), a(j+1), b(2)]$ to $tTRI$;
else
  Add a new edge d from between the nodes in a and b farthest from each other (excluding endpoints), $a(1)$ and $b(end)$;
  Add new nodes on d with `newnodes`;
  $t1TRI \leftarrow$ `traptri(`d, c`)`;
  Rename the edges $[a(2 : end), b(1 : end - 1), d]$ as a_1, b_1, c_1 such that c_1 contains the fewest nodes while $[a_1, b_1, c_1]$ maintain the original ordering (clockwise or counterclockwise);
  $t2TRI \leftarrow$ `netrefrec(`a_1, b_1, c_1`)`;
  add $t1TRI$ and $t2TRI$ to $tTRI$;

Algorithm 4: `traptri`(a, b)

Data: a and b opposite edges of the trapezoid containing extra nodes
Result: $nTRI$ triangle list
Let $d(x, y) = \|x - y\|$, unless if either x or y does not exist then $d(x, y) = \infty$;
while $(size(a) > 1) \wedge (size(b) > 1)$ **do**
  if $d(a(2), b(1)) > d(a(1), b(2))$ **then**
    add triangle $[b(2), b(1), a(1)]$ to $nTRI$;
    remove $b(1)$ from b;
  else
    add triangle $[a(1), a(2), b(1)]$ to $nTRI$;
    remove $a(1)$ from a;

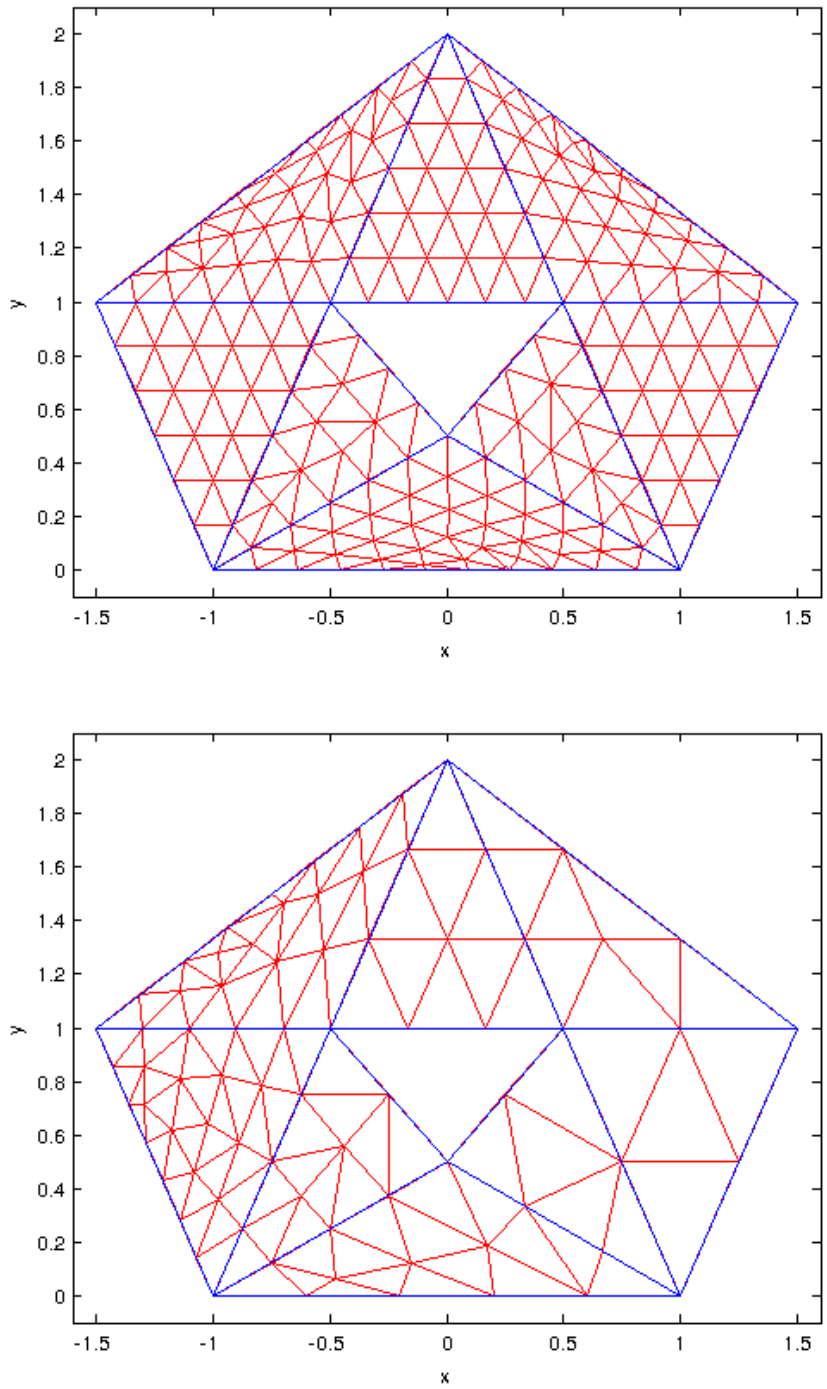

Figure 3.7: The net refinement algorithm with two different functions $f(p, \overrightarrow{v})$. On top $f(p, \overrightarrow{v}) = 0.2$ and on bottom $f(p, \overrightarrow{v}) = 0.25(p_x + 2)$. Here the blue lines show the original net, while the red and blue form the refined net. The central triangle is left blank intentionally.

3.3. Recursive Green function method

Now we will give an overview over the Recursive Green function method used to solve our system for comparison. It is a numerical method that uses finite differences to approximate the solution. For a more physical approach refer to [5, 19]. To simplify the notation we shall only consider systems where the scattering domain has the shape $\Omega_S = [x_L, x_R] \times [0, L]$.
We start by discretizing our domain Ω with a regular grid of lattice size a, where $a = L/(M-1)$ for some $M \in \mathbb{N}\backslash\{0, 1\}$.

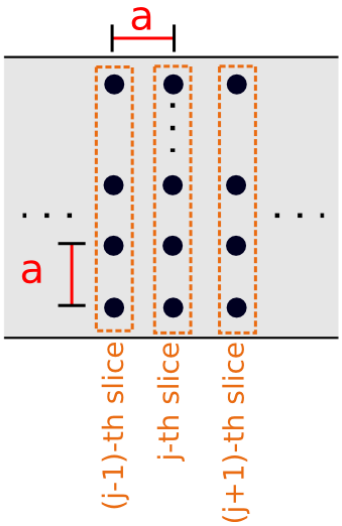

Figure 3.8: Regular grid discretization of the domain Ω for RGFM

By replacing the Laplace operator with the central finite difference five-point stencil

$$\Delta f(x,y) \approx \frac{f(x-a,y) + f(x+a,y) + f(x,y-a) + f(x,y+a) - 4f(x,y)}{a^2}, \quad (3.25)$$

we can approximate (2.6) with the matrix equation

$$[EI - H]\, G = I. \quad (3.26)$$

Here I is the identity matrix and

$$H = t \begin{bmatrix} \ddots & & & & \\ & H_{j-1} & -I & 0 & \\ & -I & H_j & -I & \\ & 0 & -I & H_{j+1} & \\ & & & & \ddots \end{bmatrix}, \quad (3.27)$$

where $t = \frac{\hbar^2}{2ma^2}$ and H_j is the $M \times M$ matrix representing the j-th vertical slice of the system, in other words

$$H_j = \begin{bmatrix} 4 + V(ia, ja)/t & -1 & & 0 \\ -1 & 4 + V((i+1)a, ja)/t & & \vdots \\ & & \ddots & -1 \\ 0 & \cdots & -1 & 4 + V((i+M)a, ja)/t \end{bmatrix}. \tag{3.28}$$

Note that since we have an infinite amount of vertical slices in our system, the matrices in (3.26) are $(\infty \times \infty)$, it would be impossible to solve this system directly numerically. To circumvent this we partition our domain Ω into the three subdomains Ω_L, Ω_R and Ω_S as was mentioned earlier. Now we can rewrite our matrix equation to the block tridiagonal equation

$$\begin{pmatrix} EI - H_L & -T_{LS} & 0 \\ -T_{LS}^* & EI - H_S & -T_{SR} \\ 0 & -T_{SR}^* & EI - H_R \end{pmatrix} \begin{pmatrix} G_{LL} & G_{LS} & G_{LR} \\ G_{SL} & G_{SS} & G_{SR} \\ G_{RL} & G_{RS} & G_{RR} \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix}. \tag{3.29}$$

Here G_{LS} denotes the Green function $G(r, r')$ where $r \in \Omega_L$ and $r' \in \Omega_S$, T_{LS} the interface block matrix between Ω_L and Ω_S and so on. The matrices T_{LS} and T_{SR} are extremely sparse, namely $T_{LS}(i, j) = T_{LS}(j, i) = t \neq 0$ only if $(x_i, y_i) \in \Omega_L$ and $(x_j, y_j) \in \Omega_S$ (or vice versa) and they are both neighbors in the grid. We note that the equation we get by multiplying the second row with the second column (3.29)

$$-T_{LS}^* G_{LS} + (EI - H_S)G_{SS} - T_{SR}G_{SR} = I \tag{3.30}$$

is an $(N \times N)$ system of equations, where N is the number of vertical slices in the scattering domain. In the equation we get by multiplying the first row with the second column, we can isolate the G_{LS} as

$$(EI - H_L)G_{LS} + T_{LS}G_S S = 0 \Rightarrow G_{LS} = (EI - H_L)^{-1} T_{LS} G_{SS}. \tag{3.31}$$

Similarly we can isolate G_{SR}. When we substitute these equations back into (3.30) we are left with

$$-\underbrace{T_{LS}^*(EI - H_L)^{-1}T_{LS}}_{\Sigma_L} G_{SS} + (E - H_S)G_{SS} - \underbrace{T_{SR}(EI - H_R)^{-1}T_{SR}^*}_{\Sigma_R} G_{SS} = I \tag{3.32}$$

or more compact

$$G_{SS}(E) = [EI - H_S - \Sigma_R - \Sigma_L]^{-1} := A^{-1}. \tag{3.33}$$

This equation can now be solved with standard linear algebra methods. Henceforth we shall denote G_{SS} only as G. Since Σ_R and Σ_L have only one nonzero block each at $\partial\Omega_R$ and $\partial\Omega_L$ respectively, the whole matrix A is $M \times M$ block tridiagonal.
G_{SS} or G is the Green function of the central system under the influence of the leads.

T_{LS} and T_{SR} are the coupling of the central system to the leads. In a closed system the energy spectrum of H_S is real valued since H_S is hermitian. The self-energies Σ_R and Σ_L describe the broadening of the energy levels of $\hat{H}$ due to the coupling to the leads. We have effectively *opened the system*, and will be tracting the dynamical processes in the central system as scattering processes in an *open system*.

3.3.1. Solving the system of equations

Once we have assembled our matrix A from (3.33), we have to decide which properties of the system we are interested in. If we are interested in studying the whole matrix G, algorithms similar to Golub's Band LU factorization [9, p. 152] handle this task nicely. If on the other hand we are only interested in the transmission function, we can take a more optimal approach.
According to [5, p. 148] the transmission function can be calculated as

$$T(E) = \mathrm{Tr}(\Gamma_L G \Gamma_R G^*), \qquad \Gamma_K = i[\Sigma_K - \Sigma_K^*], \text{ where } K = R \text{ or } L. \tag{3.34}$$

Exploiting the fact that Γ_L and Γ_R have only one non-zero block γ_L, γ_R with dimension $M \times M$ at $\partial\Omega_L$ and $\partial\Omega_R$ respectively allows us to restate the previous equations as

$$T = \mathrm{Tr}(\gamma_R G_{N,1} \gamma_L G_{N,1}^*), \tag{3.35}$$

where $G_{N,1}$ denotes the $M \times M$ lower left block of G. This block $G_{N,1}$ tells us, under a normal ordering, the value of the Green function $G(r, r')$ where r and r' lie on opposite leads. Therefore we have reduced our problem to only finding one block of G instead of inverting the whole matrix A.

Using a block tridiagonal LU factorization [9, p. 174], L and U have the following structure

$$L = \begin{bmatrix} I & & \cdots & 0 \\ L_1 & I & & \vdots \\ & L_2 & I & \\ \vdots & & \ddots & \\ 0 & \cdots & L_{N-1} & I \end{bmatrix} \text{ and } U = \begin{bmatrix} U_1 & I & \cdots & 0 \\ & U_2 & I & \vdots \\ & & \ddots & \\ \vdots & & U_{N-1} & I \\ 0 & \cdots & & U_N \end{bmatrix} \tag{3.36}$$

Using back- and forward substitution, instead of solving $AG = I$ for G we can

$$\begin{gathered} \text{first solve } LY = I \text{ for } Y \\ \text{and then solve } UG = Y \text{ for } G. \end{gathered} \tag{3.37}$$

Since we are only after the block $G_{N,1}$, we need to carefully consider which blocks need solving and which don't in each system. In the second equation of (3.37),

looking only at the product of the last rows of U with the first columns of G, we have

$$U_N G_{N,1} = Y_{N,1} \qquad \Leftrightarrow \qquad G_{N,1} = U_N^{-1} Y_{N,1}. \tag{3.38}$$

Now by looking at the product of the i-th row of L with the first column of Y we have

$$L_{i-1} Y_{i-1,1} + I Y_{i,1} = 0 \qquad \Leftrightarrow \qquad Y_{i,1} = -L_{i-1} Y_{i-1,1} \qquad \text{for } i \in \{2, ..., N\} \tag{3.39}$$

and $Y_{1,1} = I$. Therefore to calculate $G_{N,1}$ we only need to evaluate the following equation

$$G_{N,1} = (-1)^{N-1} U_n^{-1} \left(L_{N-1} L_{N-2} \cdots L_1 \right). \tag{3.40}$$

By using a block tridiagonal LU factorization we thus not only save time by not having to find the whole inverse of A, but we also can reduce our memory usage greatly by only having to store at most matrices the size of $M \times M$ at a time as long as the domain has this highly regular shape.

4. Results

4.1. Comparison with analytical solutions

To verify the solution from our finite element method program we need to compare it to known solutions. There is, however, one preliminary decision left; we need to determine how many terms in the series (3.24) from the boundary integral we want to use in our calculations. As we can see in Figure 4.1, we can truncate the sum after a few tens of terms, thus in all subsequent calculations we will use 50 terms of this series.

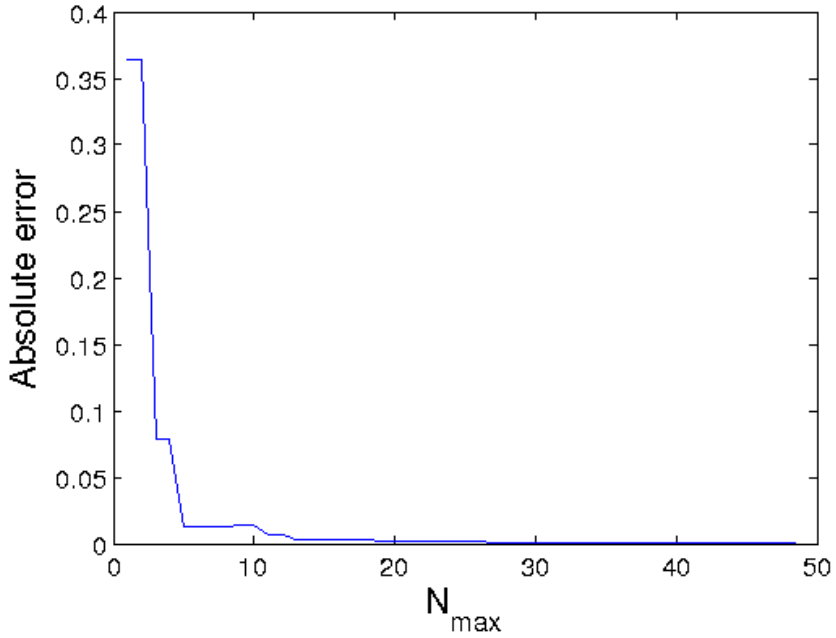

Figure 4.1: Absolute error of the boundary series (3.24) versus number of terms used. Here $E = \frac{\hbar^2}{2ma}$, $L = 4a$, $\phi_k(x_L, 2a) = \phi_j(x_L, 3a) = 1$.

Now we start with a simple case.

4.1.1. Straight scattering domain without an electrostatic potential

Suppose that we are looking at a straight wire, that is $\Omega_S = [x_L, x_R] \times [0, L]$ and $V(r) = 0$ for all $r \in \Omega$. We discretize the domain with a regular grid, where all nodes are spaced distance a apart. To find the solution we have to invert matrix A from (3.10). Now for a certain energy E and point r' we see the solution $G(r, r'; E)$ from the FEM program in Figure 4.3. The characteristics are as we expect, an extremum at $r = r'$ and an overall wave-like behavior while still satisfying the boundary conditions.

To assure ourselves that this solutions is correct we look at the transmission function, seen in Figure 4.2. Notice that for low energies the analytical solution and the solutions from the FEM coincide. As the energy increases the calculated solutions branch off one after another, thus requiring us to refine the mesh to obtain accurate results.

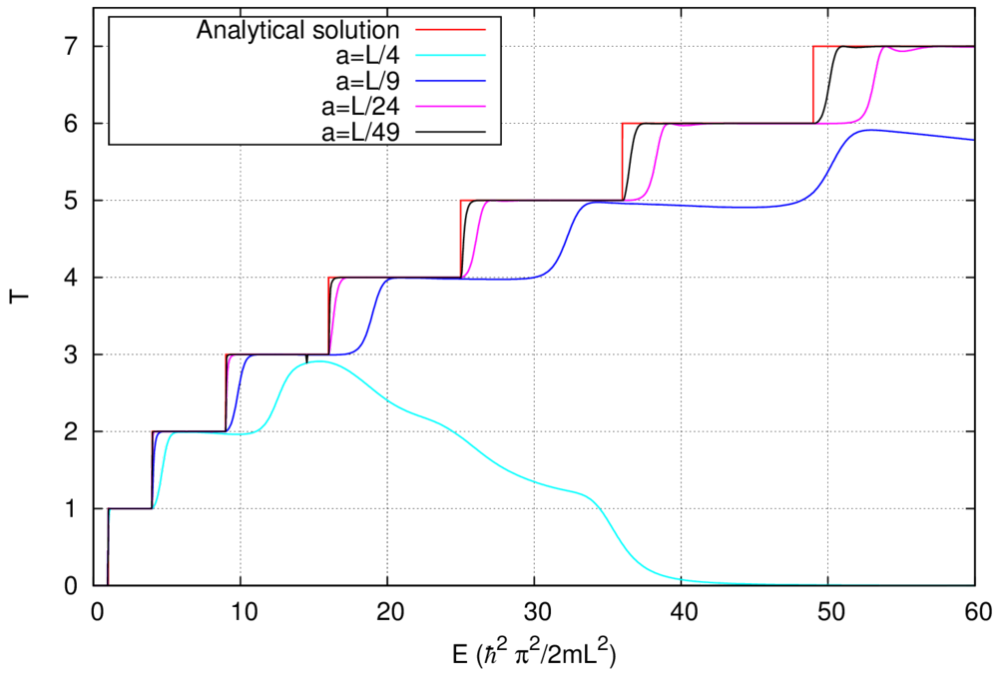

Figure 4.2: Transmission coefficient for the straight scattering domain with no electrostatic potential $V(r)$ calculated with FEM. Since a determines the number of nodes in the grid, $a = L/4$ is calculated on a 5×5 grid, $a = L/9$ uses a 10×10 grid and so on.

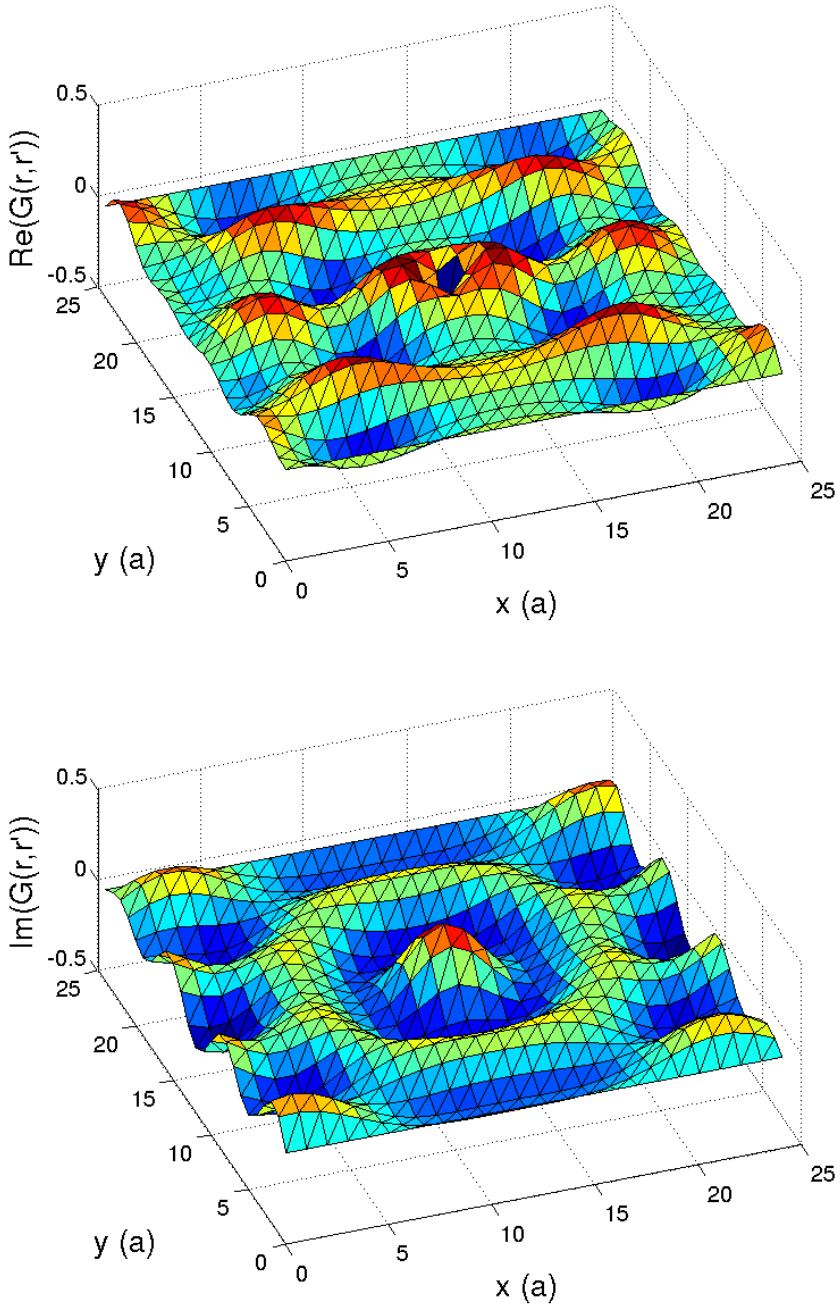

Figure 4.3: Green function calculated using FEM. Here r' is located at the center of Ω_S and $E = \frac{\hbar^2}{2mL^2}$.

4.1.2. Straight scattering domain with an electrostatic potential

Now, we add an electrostatic potential with the following shape:
for $r = (x, y) \in \Omega_S = [x_L, x_R] \times [0, L]$ we have

$$V_1(x, y) = \begin{cases} 2V_0 & \text{if } \frac{1}{8}(x_R - x_L) < x - x_L < \frac{7}{8}(x_R - x_L) \\ 0 & \text{else,} \end{cases} \tag{4.1}$$

where $V_0 = \frac{\hbar^2 \pi^2}{2mL^2}$. Note that this potential is invariant in the y-direction.
For a 1-D system the transmission coefficient of a particle traveling through a potential barrier with height V_0 is divided into two cases [4, p. 73]: $E > V_0$ and $E < V_0$. When $E > V_0$ we have

$$T_{1D}(E) = \frac{4E(E - V_0)}{4E(E - V_0) + V_0^2 \sin^2\left(\sqrt{2m(E - V_0)}l/\hbar\right)}, \tag{4.2}$$

where l is the length of the potential. While when $E < V_0$, we get a tunneling effect and the transmission coefficient becomes

$$T_{1D}(E) = \frac{4E(V_0 - E)}{4E(V_0 - E) + V_0^2 \sinh^2\left(\sqrt{2m(V_0 - E)}l/\hbar\right)}. \tag{4.3}$$

Now since our potential V_1 is invariant in the y-direction the analytical transmission coefficient simply becomes a sum over the different wave-modes where each wave-mode's contribution to the transmission coefficient is as in the 1-D case. In other words

$$T(E) = \sum_{n=1}^{\infty} H(E - E_n) T_{1D}(E - E_n), \tag{4.4}$$

where H is the Heaviside step function and E_n is the energy required for our particle to enter the n-th wave-mode. Figure (4.4) shows the transmission coefficient where we added V_1 and $V_2 \equiv -V_1$ to Ω_S. The derivation of the analytical transmission coefficient for V_2 is analogous to V_1. As we can see the numerical FEM results approximate the analytical solution fairly well.

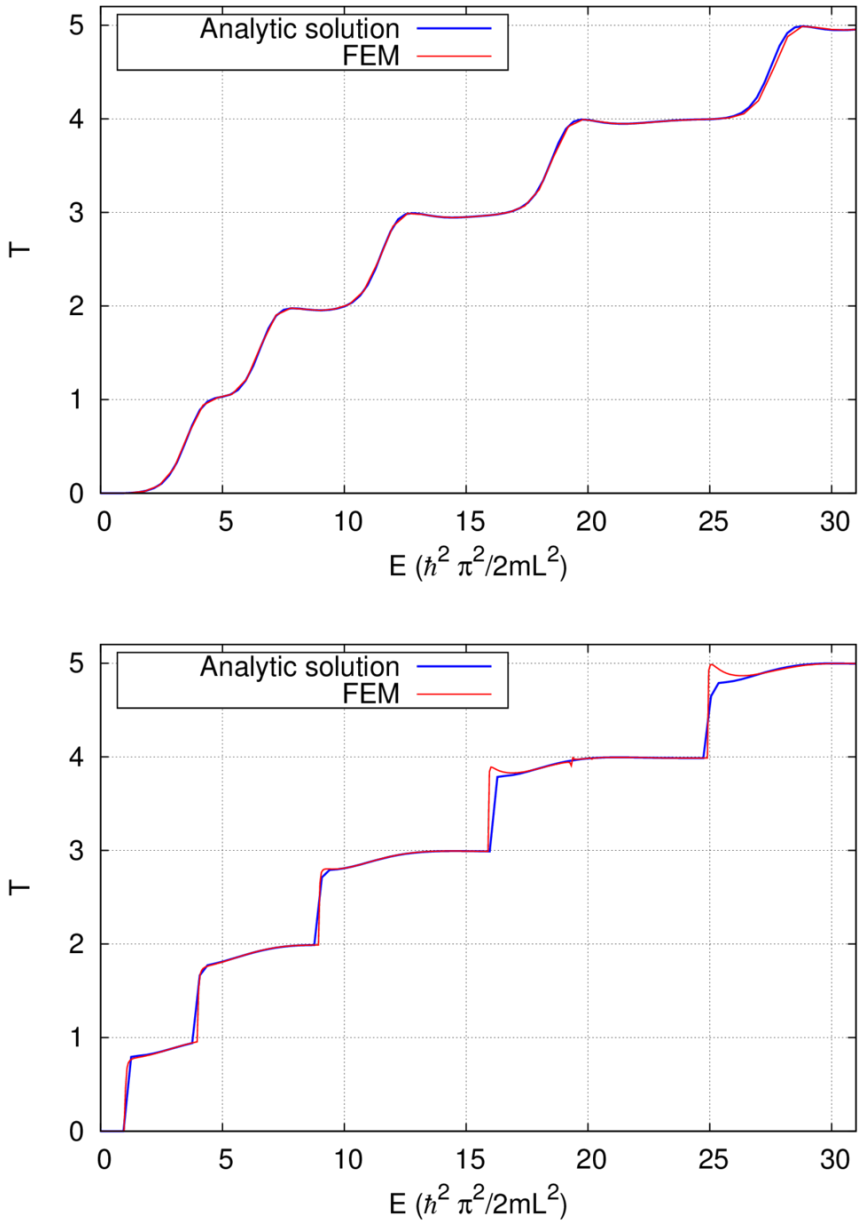

Figure 4.4: Transmission coefficient for the straight scattering domain with electrostatic potential V_1 (top) and $V_2 = -V_1$ (bottom) calculated with FEM on a regular 49×49 grid.

Finally let V_3 be the potential as in Figure 4.5. Note that this electrostatic potential is not invariant in the y-direction, therefore we are unable to construct the analytical transmission coefficient from the one dimensional case.

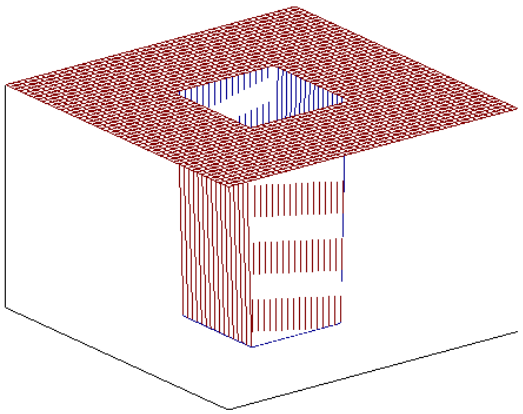

Figure 4.5: The shape of the electrostatic potential V_3 with depth $-0.02\frac{\hbar^2\pi^2}{2mL^2}$

Figure 4.6 shows that we get a dip in the transmission coefficient when $E \in [7, 8]\frac{\hbar^2\pi^2}{2mL^2}$. Physically we can interpret this as when a particle enters the system from one lead, it is more likely to be reflected in the potential well and return to the same lead for a certain energy. This behavior is well known and is caused by an evanescent state situated just below the third subband and it coincides with other known solutions [2].

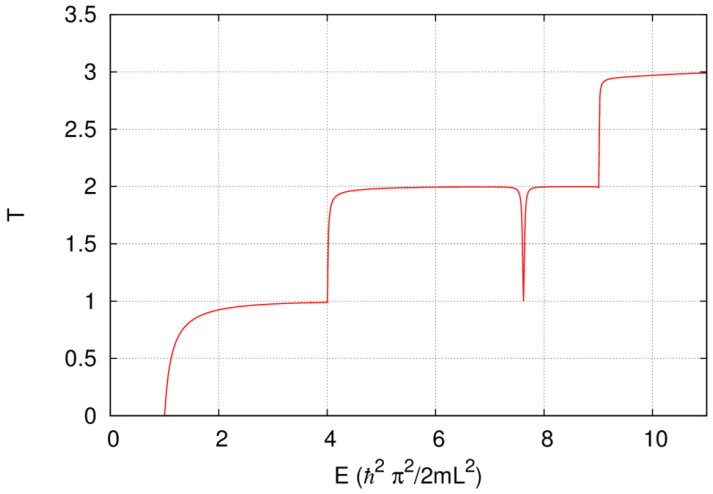

Figure 4.6: Transmission coefficient for a straight scattering domain with electrostatic potential V_3 calculated using the FEM on a 25×25 regular grid.

4.2. Comparison of the numerical methods

Now let us compare the solutions using the finite element method and the recursive Green function method.

4.2.1. Straight scattering domain with an electrostatic potential

Figure 4.7 shows the transmission coefficient from analytical solution compared to the FEM and RGFM for a straight scattering domain Ω_S with electrostatic potential V_1 as in the previous section. We see that the numerical methods approximate

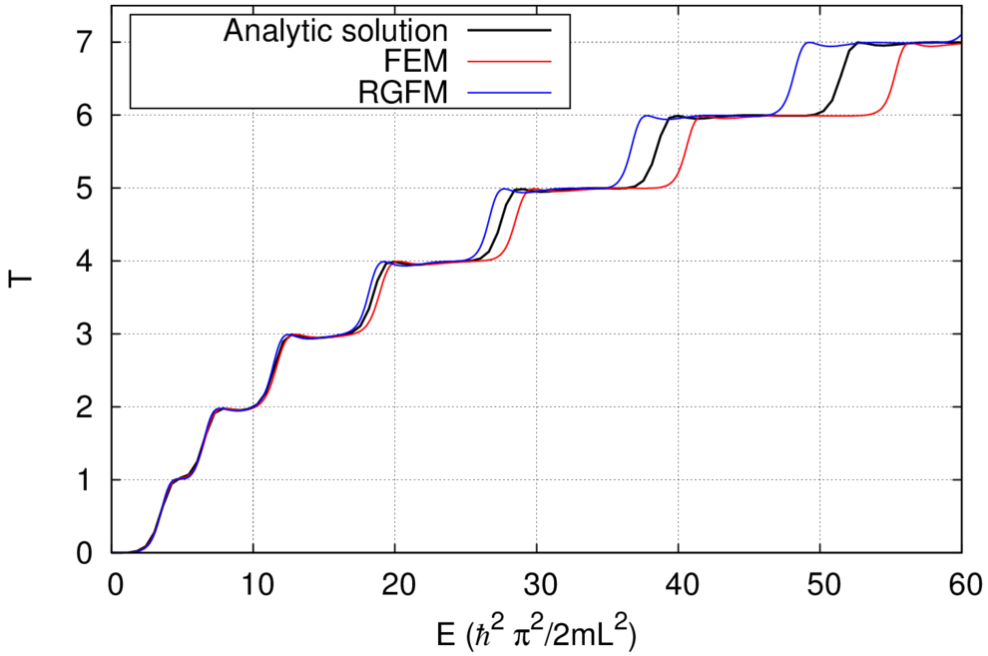

Figure 4.7: Comparison of transmission coefficients in a system with potential V_1. The grid size for the numerical solutions was equal.

the analytical solution about equally well. Note that the RGFM overestimates the transmission coefficient while the FEM underestimates it. This is most likely caused by the different ways these two methods approximate the boundary integrals (i.e. self energy). This effect could be reduced by using higher order approximations or finer grids.
For domains with such simple shape it is advantageous to use the RGFM method since it is less computationally intensive for this special case.

4.2.2. Bottle-neck shaped scattering domain

Next we shall study a system with a different shape; a bottle-neck scattering domain. Let Ω_S be as in Figure 4.8, where the leads connect to the left and right side and the width of the central region being 1/3 of width of the leads.

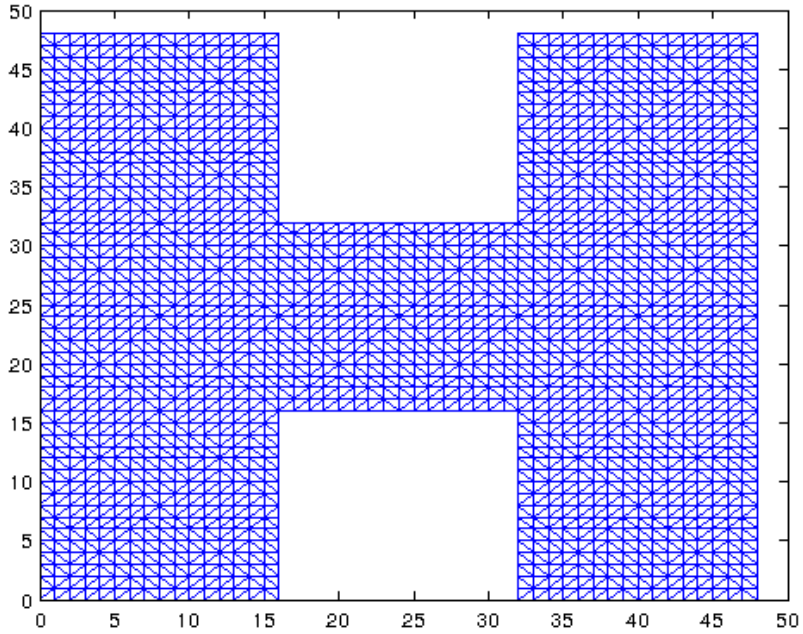

Figure 4.8: Discretized bottle-neck shaped scattering domain.

We compare the transmission coefficients, see Figure 4.9, for three different solutions: a RGFM solution and two FEM solutions with a regular and irregular grid. The grid used by the RGFM and regular grid FEM is as in Figure 4.8 while the irregular grid FEM uses the grid seen in Figure 4.10. These two grids have comparable amounts of nodes, 1759 and 1779 respectively. The irregular grid was generated using the algorithm discussed in section 3.2 by refining a more coarse grid. This grid was chosen to be finer at the leads to improve numerical accuracy, as discussed in [10], and at the corners since these areas are generally troublesome in numerical calculations.
We can see, however, that the methods using the regular grids appear to be more stable since the irregular FEM method does not reach the fifth wave mode. Also note the behavior of overestimation by the RGFM and underestimation by the FEM is prevalent as before. Again it seems that using the RGFM is more effective.

Thus to fully utilize the strengths of the FEM we shall look at scattering domains with even more complex geometry.

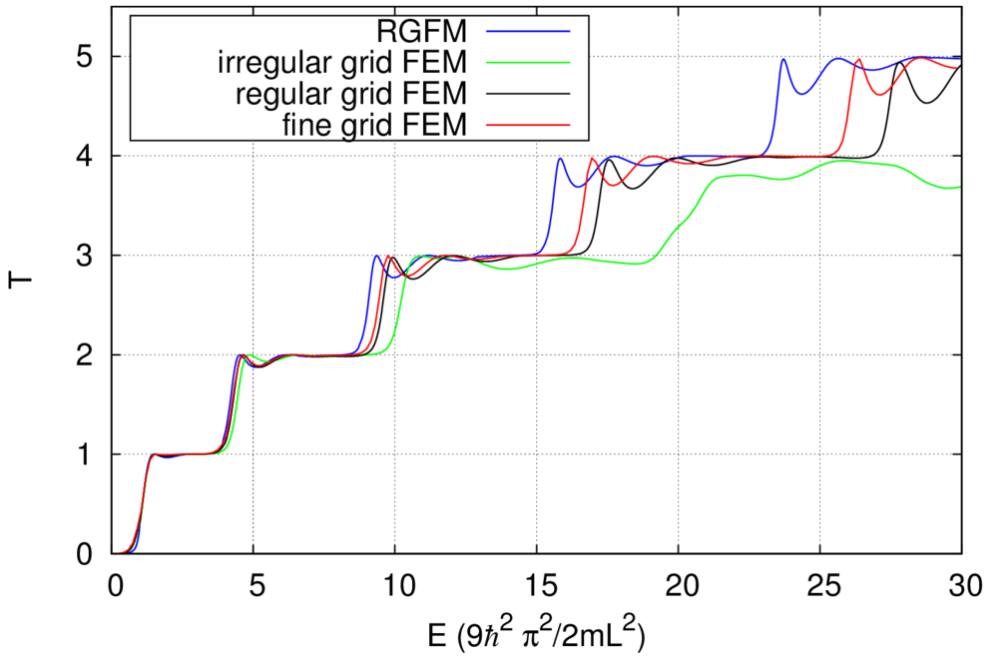

Figure 4.9: Transmission coefficient of a bottle-neck shaped scattering domain, calculated with RGFM and a regular and irregular grid FEM. The fine grid FEM is also on a regular grid and has about 4 times as many nodes as the other methods.

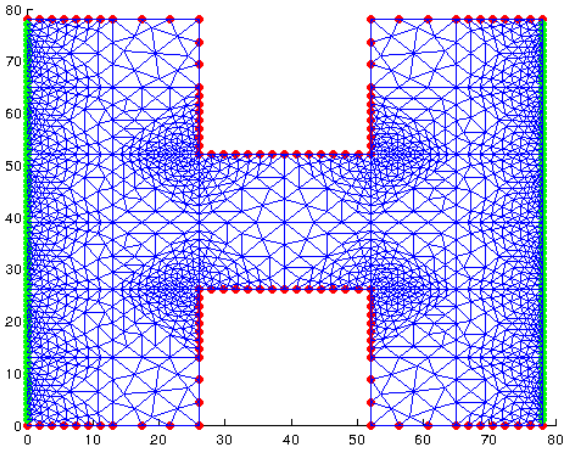

Figure 4.10: Irregular grid used by the FEM on the bottle-neck shaped scattering domain generated by the grid refinement algorithm. In the figure green dots represent the leads and red dots the Dirichlet boundary.

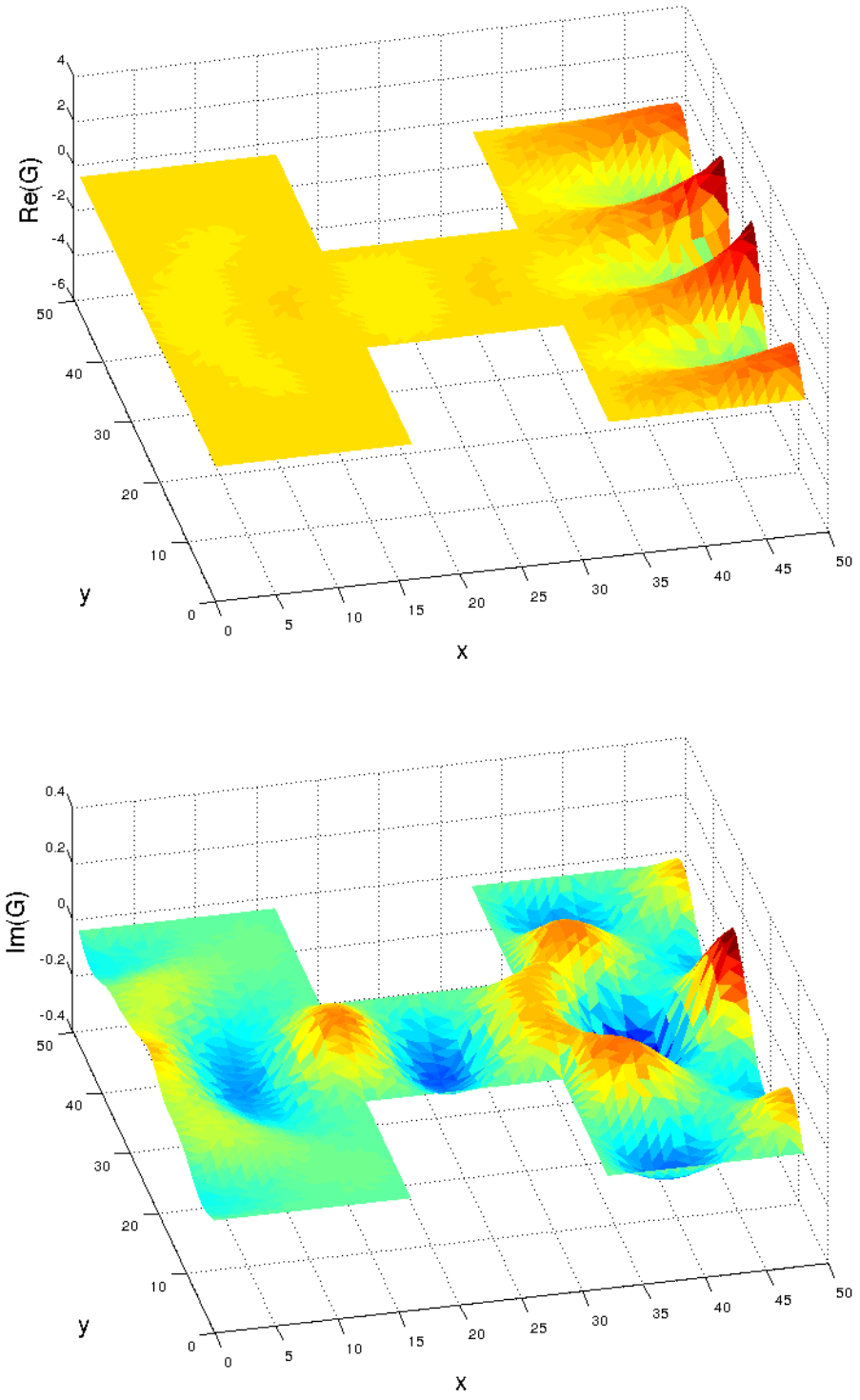

Figure 4.11: Green function of the bottle-neck shaped scattering domain with $Re(G(r,r'))$ (top) and $Im(G(r,r'))$ (bottom), where r' lies on the middle of the right lead and $E = 5\frac{9\hbar^2\pi^2}{2mL^2}$.

4.3. Numerical Modeling of quantum wire with annular shape

Finally we turn to a more complicated system; an annular scattering domain. Let the scattering domain be as in Figure 4.12.

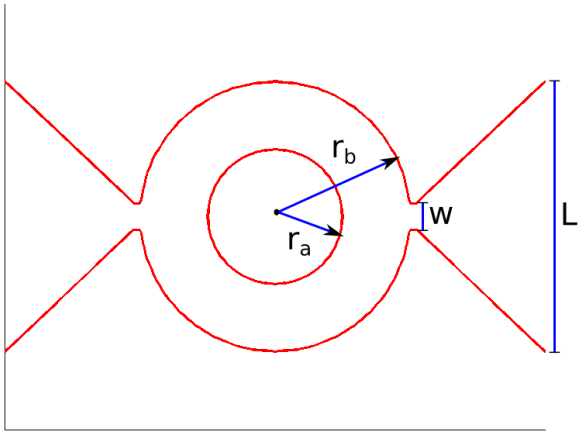

Figure 4.12: Annular scattering domain annotated with variable names.

In all subsequent calculations we will use radii r_a and r_b such that $r_b/r_a = 2$. Computations from our FEM program yields results as seen in Figure 4.14 and 4.13.

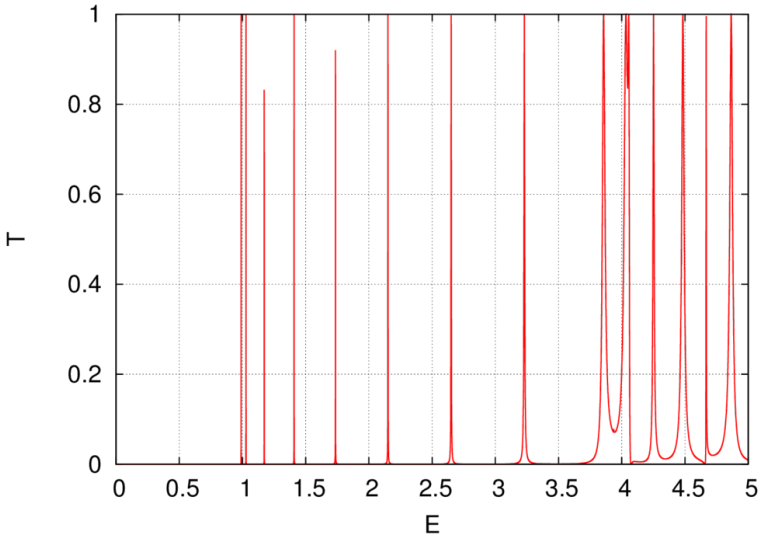

Figure 4.13: Transmission function for the annulus as a function of energy. The energy is scaled as $\frac{\hbar^2\pi^2}{2m(r_b-r_a)^2}$. Here $w/L = 1/10$.

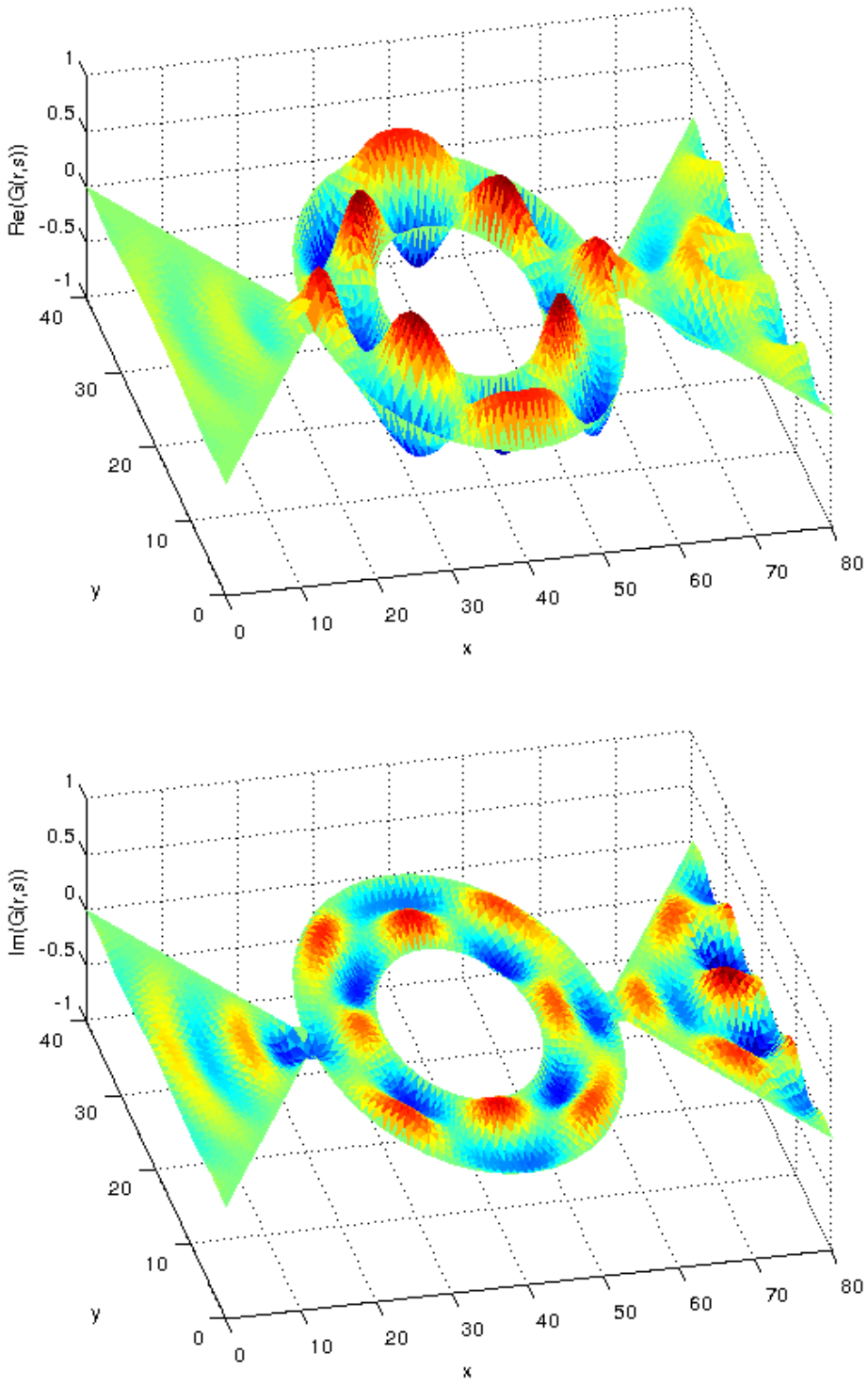

Figure 4.14: A plot of $Re(G(r,r'))$ (top) and $Im(G(r,r'))$ (bottom), where r' lies close to the right lead and $E = 4.8644\frac{\hbar^2\pi^2}{2m(r_b-r_a)^2}$. Here $w/L = 1/10$.

4.3.1. Asymptotic behavior

Now to verify the accuracy of our solutions take note of the following problem: Let $f(r,\theta) : \mathbb{R}^2 \to \mathbb{C}$ be a solution to the Helmholtz equation

$$\begin{aligned} &(\Delta + k^2) f(r,\theta) = 0, \qquad k \in \mathbb{R} \\ &f(r_a,\theta) = f(r_b,\theta) = 0, \end{aligned} \tag{4.5}$$

on an annulus with radii r_a and r_b. It is easy to show that

$$f(r,\theta) = \sum_{n=-\infty}^{\infty} \left(c_n J_n(kr) + d_n Y_n(kr)\right) e^{in\theta}, \tag{4.6}$$

satisfies (4.5) with some coefficients c_n and d_n and Bessel functions J_n and Y_n. We also know that for a given n, whether

$$\det\left(\begin{bmatrix} J_n(kr_a) & Y_n(kr_a) \\ J_n(kr_b) & Y_n(kr_b) \end{bmatrix}\right) = 0, \tag{4.7}$$

holds, is a necessary and sufficient condition for J_n and Y_n to be in the solution basis. Thus if this does not hold we have that $c_n = d_n = 0$. Equivalently we could look at this problem as finding eigenvalues and their associated eigenfunctions for the Laplace equation on an annulus.

Since the Hamiltonian

$$\hat{H} = -\frac{\hbar^2}{2m}\Delta + V(r), \tag{4.8}$$

is an operator that possesses a complete set of eigenfunctions [7], we can construct the Green function as a eigenfunction expansion

$$G(r,r';E) = \sum_{n=0}^{\infty} \frac{\psi_n(r)\psi_n(r')^*}{E - \lambda_n}. \tag{4.9}$$

Here ψ_n denotes the n-th eigenfunction and λ_n the corresponding eigenvalue. This equation only holds if $E \neq \lambda_n$ for all n. Therefore when w/L is small the eigenfunctions for the Hamiltonian and (4.5) are nearly identical, and thus if $E \approx \lambda_n$, we have $\mathrm{Re}(G(r,r')) \approx \mathrm{Im}(G(r,r')) \approx \psi_n(r)$ for r' outside the annular center. We should consequently expect the transmission coefficient to jump up as E gets closer to an eigenvalue λ_n.
Thus when we consider our original problem again we would expect that as $w/L \to 0$, our solution $G(r,r')$ would asymptotically approach $f(r,\theta)$ for r' on $\partial\Omega_R$ (or $\partial\Omega_L$) and $k = \sqrt{\frac{2mE}{\hbar^2}}$ in (4.5). Note that for low values of k we get only a few n for which (4.7) is satisfied, but they get more numerous as k grows. This behavior is reflected in the transmission coefficient in Figure 4.13; the first tops are few and thin, but

they get more frequent and wider as E grows.
We can calculate the lowest value of $k = k_0$ for which (4.7) holds for some n. As it turns out it holds only for $n = 0$. The second lowest value of $k = k_1$ for which the equation holds is for $n = \pm 1$.

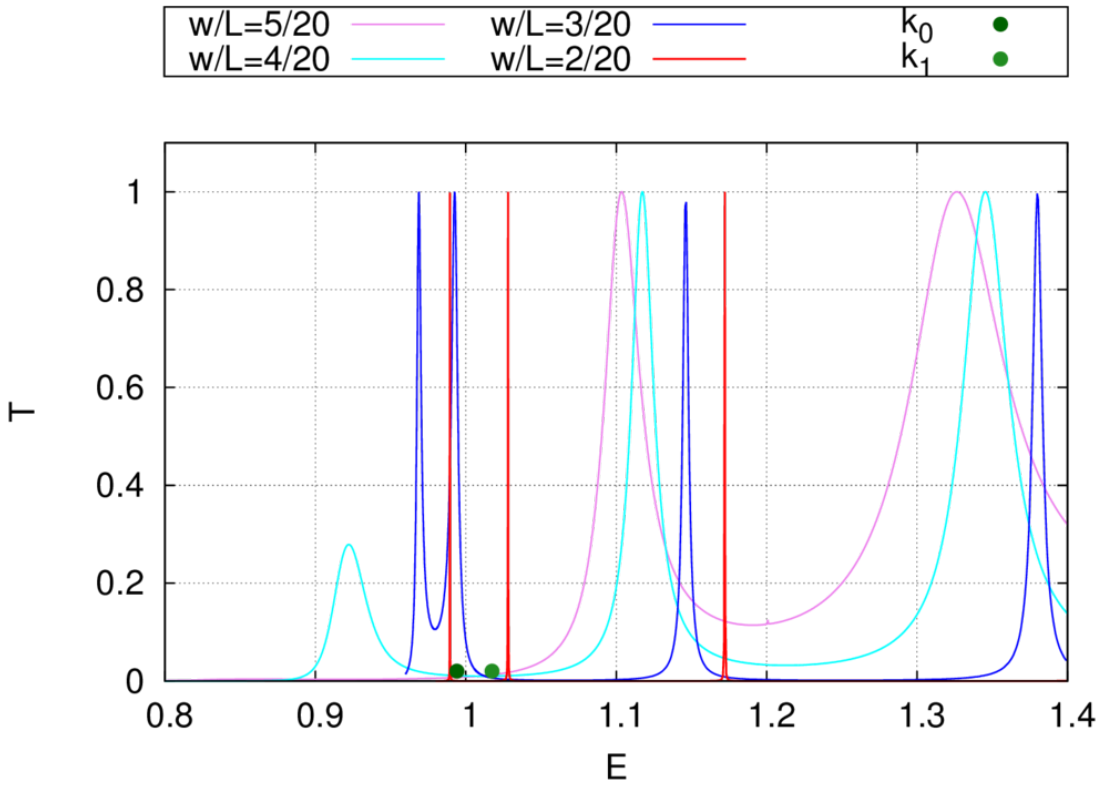

Figure 4.15: Transmission function for the annulus for different values of w/L. The energy is scaled as $\frac{\hbar^2\pi^2}{2m(r_b-r_a)^2}$

As we see from Figure 4.15, the smaller the gap w/L is, the closer the first two tops of the transmission coefficients are to k_0 and k_1. Thus if we graph the Green function for different values of w/L at energies equating to the first two tops in the transmission coefficients for each value of w/L and compare them to $f(r, \theta)$ at $k = k_0$ and $k = k_1$ as in Figure 4.16 and 4.17, we see that as w/L decreases we get solutions with more resemblance to $f(r, \theta)$. Note that in Figure 4.16 both the the real and imaginary part of the solution $G(r, r')$ with $w/L = 2/20$ resemble the real part of the eigenfunction but with a different sign. We can see that this stems from (4.9) when $E < \lambda_0$.
We conclude that our solutions follow the asymptotic behavior that we were seeking.

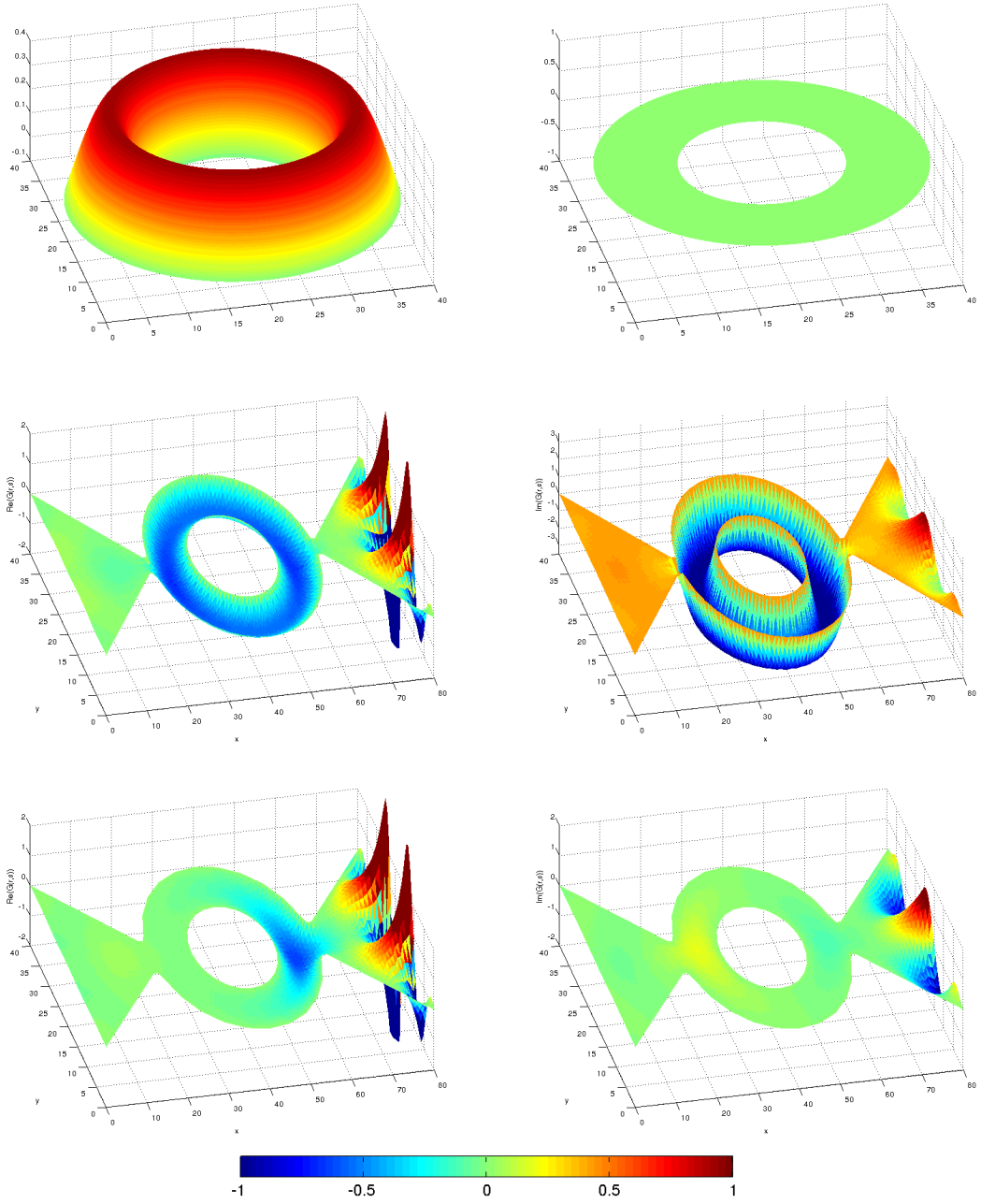

Figure 4.16: Comparison between $f(r,\theta)$ (top) at $k = k_0$ and G(r,r') with $w/L = 2/20$ (middle) and $w/L = 4/20$ (bottom) for energies equal to the first top in the transmission coefficient. Column 1 denotes the real value while column 2 is the imaginary value. Here r' is on the center of the right lead. Values outside the range of the color bar are mapped to the respective ends. Note that the color scale for the eigenfunctions (top) are exaggerated.

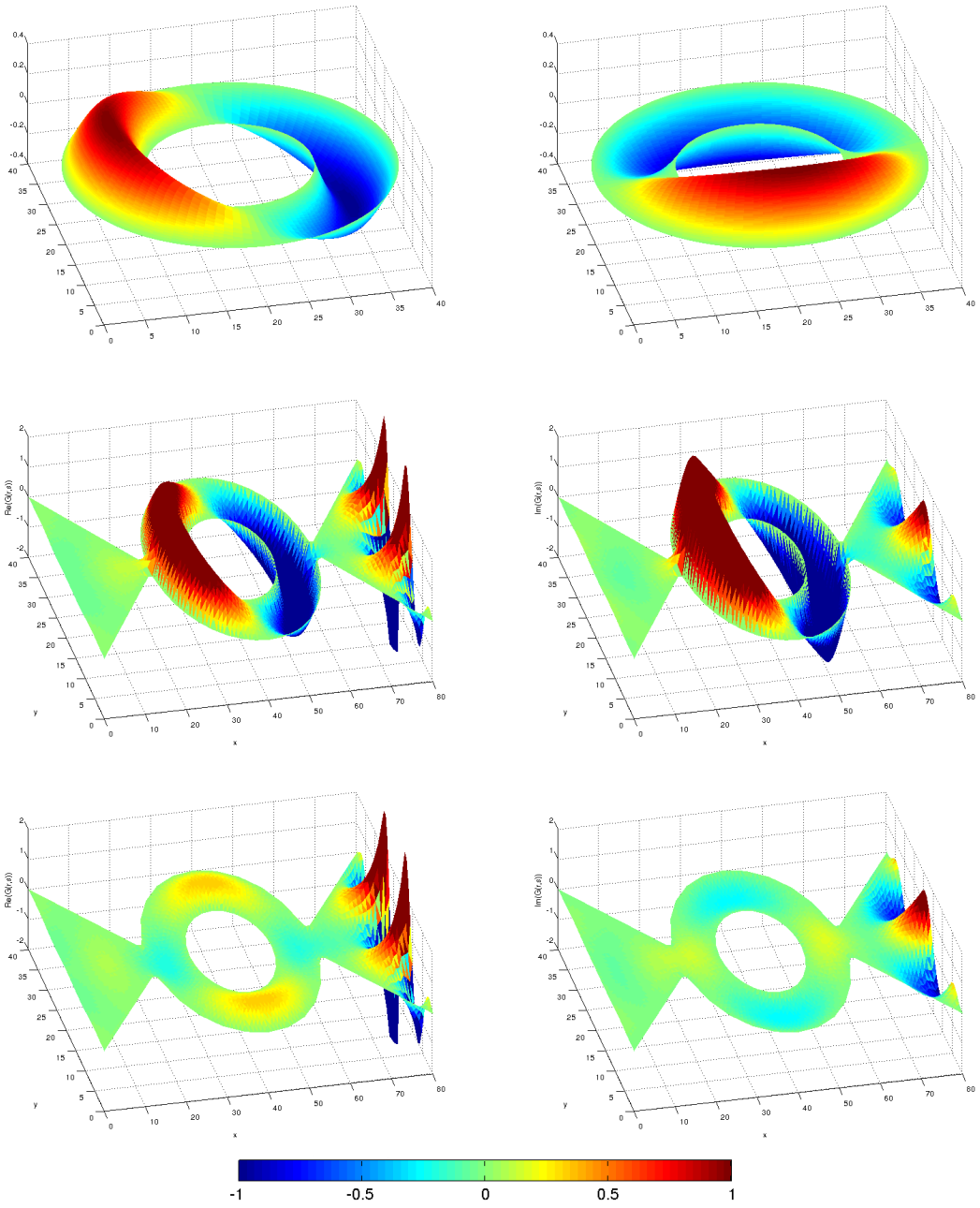

Figure 4.17: Comparison between $f(r,\theta)$ (top) at $k = k_1$ and $G(r,r')$ with $w/L = 2/20$ (middle) and $w/L = 4/20$ (bottom) for energies equal to the second top in the transmission coefficient. Column 1 denotes the real value while column 2 is the imaginary value. Here r' is on the center of the right lead. Values outside the range of the color bar are mapped to the respective ends. Note that the color scale for the eigenfunctions (top) are exaggerated.

Finally we can make some comparison between the quality of the solution from the finite element method and recursive Green function method on the annular domain. For a narrow gap, $w/L = 2/20$, we would expect the first two tops of the transmission coefficient to be close to k_0 and k_1. When we compare the distance between the first and second top to k_0 and k_1 respectively for the different methods we get Figure 4.18.

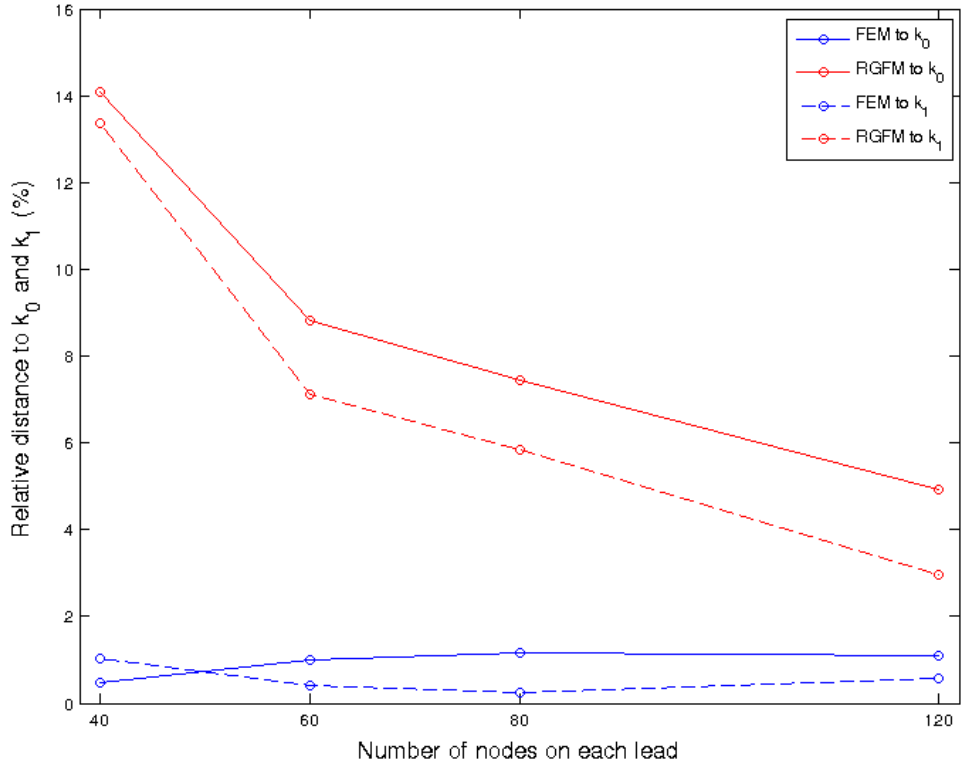

Figure 4.18: Relative distance of the first two tops of the transmission coefficient to the eigenvalues k_0 and k_1 for the FEM and RGFM. For top t, the relative distance is $t = |t - k_i|/k_i$ with $i = 0$ or 1. The number of nodes used by each method in the different grids where similar.

As we see, the position of the tops according to the finite element method are much closer to where we expect the correct values to be. While the positions of the RGFM are further away, they seem to be converging to the same point as the FEM. This gives us some empirical evidence of the better numerical accuracy of the finite element method.

4.4. Parallelization

We must solve the system multiple times for different energy values to view how the system behaves. To speed up calculations and better utilize available hardware, the FEM code was parallelized. Since we can solve our system for different energy values independent of each other, we can do these calculations in parallel. We use open-MPI[1] for this task; suppose we have a computer cluster, then each computer participating in the computation gets a subset of energy values to calculate and the results are gathered in the end on a single computer. If each computer has multiple cores we can parallelize the code even further; for a given computer and energy value, a computer has to calculate the inverse of matrix A (3.10) and we parallelize this by using routines from Intel's MKL-library[2]. Additionally if we also want to compute the transmission coefficient, we can do those calculations in parallel by using OpenMP[3].
Thus given the right hardware the speedup can be tremendous. Most of the code was written in Fortran. It would also have been possible to speed up each individual factorization of the matrix A from (3.10) using techniques such as a fan-in sparse Cholesky factorization algorithm [11]. In doing so a few computers would factor the same matrix, each starting at an independent block and then sharing these intermediate results to complete the factorization. But since the number of different energy values we wanted to calculate each solution at where usually much greater than the number of available computers participating in the computation, it was decided more fruitful to allocate each computer to different energy values.

[1] http://www.open-mpi.org/
[2] http://software.intel.com/en-us/intel-mkl
[3] http://openmp.org/

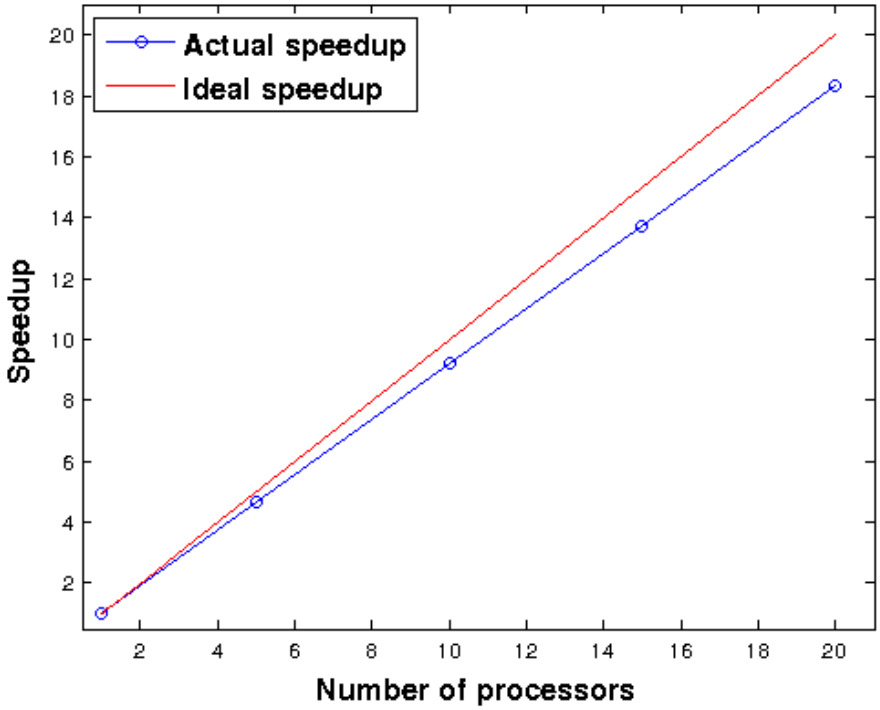

Figure 4.19: Typical speedup in the finite element code when using open-MPI. Speedup S_p is defined as the ratio between the time need for one processor the execute a program versus the time needed for p processors.

5. Conclusions

We started by detailing a rigorous analytical foundation for the quantum scattering problem and giving analytical solutions to simple cases. We then developed a finite element approach while exploring the traditional finite differences approach of the recursive Green function method for comparison. Next we detailed our finite element mesh refinement algorithm by giving pseudocode for reference. We calculated solutions for a straight scattering domain, with and without an electrostatic potential, and noted that empirically the two numerical methods approximated the analytical solutions equally well. When constructing the electrostatic potential in a specific way we also found more complex behaviors of evanescent states in our solutions from the finite element method lending additional credibility to the stability of the solution. Thereafter, we turned our attention to the bottle-neck scattering domain where again the numerical methods with regular grids produced comparable results, while a specialized irregular grid FEM produced less accurate results. However, when we studied the annular shaped scattering domain we found the same complex asymptotic behaviors in our FEM solution as predicted by analytical theory. Overall the finite element method thus showed again its qualities in flexibility and adaptability.

We successfully modeled a finite element approach of the quantum scattering problem which gives comparable results to traditional tools using (RGFM) for domains with simple geometry while excelling at scattering problems with more complex domains. Since most of the code was written to utilize parallel computation these results are scalable. Additionally, the new mesh refinement algorithm presented allows for precise control over hardware resources needed for computations. The way these systems work together allows for rapid prototyping of problems with varying geometry.
There are still many ways to extend the work done in this thesis such as implementing effects from magnetic fields or support for multiple terminal nanostructures. Also, the ordering of the FEM nodes for different geometries of the domain could be studied in order to reduce computation time. Finally, experimenting with different ways of implementing the boundary conditions on the left and right lead could give a better insight into the over- and underestimation trend of the two numerical methods seen in section 4.2.

Bibliography

[1] Kendall Atkinson and Weimin Han. *Theoretical numerical analysis.* Springer, third edition, 2009.

[2] Jens Hjorleifur Bardarson, Ingibjorg Magnusdottir, Gudny Gudmundsdottir, Chi-Shung Tang, Andrei Manulescu, and Vidar Gudmundsson. Coherent electronic transport in a multimode quantum channel with Gaussian-type scatterers. *Phys. Rev. B*, 70, December 2004.

[3] Dietrich Braess. *Finite elements.* Cambridge University press, third edition, 2007.

[4] Claude Cohen-Tannoudji, Bernard Diu, and Frank Laloe. *Quantum Mechanics, Vol. 1.* Wiley, first edition, 1991.

[5] Supriyo Datta. *Electronic transport in mesoscopic system.* Cambridge university press, 1995.

[6] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer, third edition, 2008.

[7] Eleftherios N. Economou. *Green's Functions in Quantum Physics.* Springer, third edition, 2006.

[8] Daniel S. Fisher and Patrick A. Lee. Relation between conductivity and transmission matrix. *Phys. Rev. B*, 23, June 1981.

[9] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, third edition, 1996.

[10] P. Havu, V. Havu, M. J. Puska, and R. M. Nieminen. Nonequilibrium electron transport in two-dimensional nanostructures modeled using Green's functions and the finite-element method. *Phys. Rev. B*, 69, March 2004.

[11] Michael T. Heath, Esmond NG, and Barry W. Peyton. Parallel algorithms for sparse linear systems. *SIAM review*, 33, September 1991.

[12] Thomas Ihn. *Electronic Quantum Transport in Mesoscopic Semiconductor Structures.* Springer, first edition, 2004.

[13] Yoseph Imry and Rolf Landauer. Conductance viewed as transmission. *Reviews of Modern Physics*, 71, 1999.

[14] Victor Isakov. *Inverse Problems for Partial Differential Equations.* Springer, second edition, 2006.

[15] P.A. Martin. *Multiple Scattering, Interaction of Time-Harmonic Waves with N Obstacles.* Cambridge, first edition, 2006.

[16] Gordon E. Moore. Cramming components onto integrated circuits. *Electronics Magazine*, page 4, 1969.

[17] Gilbert Strang and George Fix. *An analysis of the finite element method.* Wellesey-Cambridge, second edition, 2008.

[18] Walter A. Strauss. *Partial differential equations.* John Wiley & Sons, Inc., second edition, 2008.

[19] G. Thorgilsson, G. Viktorsson, and S. I. Erlingsson. Recursive Green's function method for multi-terminal nanostructures, 2013. http://arxiv.org/abs/1305.7363.

A. Analytical solution on the unbounded domain

We want to solve

$$\left(\frac{\hbar^2}{2m}\Delta + E\right) G(\mathbf{r}, \mathbf{r}') = \delta(\mathbf{r} - \mathbf{r}') \tag{A.1}$$

on the whole domain $\mathbb{R}^2$. Start by multiplying both sides with $2m/\hbar^2$

$$\left(\frac{\hbar^2}{2m}\Delta + \frac{2mE}{\hbar^2}\right) G(r, r') = \frac{2m}{\hbar^2}\delta(\mathbf{r} - \mathbf{r}'). \tag{A.2}$$

The solution should only depend on the distance between r and r', therefore we start by solving

$$(\Delta + k^2)g(r) = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}g(r)\right) + k^2 g(r) = 0, \qquad r > 0 \tag{A.3}$$

for some function $g(r)$ and $k^2 = \frac{2mE}{\hbar^2}$ in polar coordinates $\mathbf{r} = (r, \theta)$. Rewrite this equation as ordinary differential equation and multiply with r^2

$$r^2 g''(r) + r g'(r) + k^2 r^2 g(r) = 0. \tag{A.4}$$

Substitute $g(r)$ with $h(r)$ by and divide by k^2

$$g(r) = h(kr) \tag{A.5}$$

$$r^2 h''(r) + r h'(r) + r^2 h(r) = 0. \tag{A.6}$$

We look for solutions that are unbounded near $r = 0$ and they are known to be

$$h(r) = Y_0(r) = \frac{2}{\pi} J_0(r) \ln(r/2) + \sum_{n=0}^{\infty} b_n r^n, \qquad b_n \in \mathbb{R} \tag{A.7}$$

where Y_0 is the Bessel function of the second kind. Thus the Green function $G_0(r, r') = h(|r - r'|)$ is a fundamental solution to (A.1) and has the form

$$G(\mathbf{r}, \mathbf{r}') = cY_0\left(\frac{\|\mathbf{r} - \mathbf{r}'\|}{k}\right), \qquad c \in \mathbb{C}. \tag{A.8}$$

To determine the coefficient c, we note that for some test function $\phi(r,\theta) = \phi(r)$, $r' = 0$ and $G_0(r) = G(r,0)$, we have

$$\frac{2m}{\hbar^2}\phi(0) = \langle(\Delta+1)G,\phi\rangle = \langle G,(\Delta+1)\phi\rangle \tag{A.9}$$

$$= \lim_{\epsilon\to 0}\int_0^{2\pi}\int_\epsilon^\infty G_0(r)\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}(r,\theta)\right)+\phi(r,\theta)\right)rdrd\theta \tag{A.10}$$

$$= \lim_{\epsilon\to 0}\int_0^{2\pi}\left(\underbrace{\left[G_0(r)r\frac{\partial\phi}{\partial r}\right]_\epsilon^\infty}_{=0}+\int_\epsilon^\infty -G_0'(r)r\frac{\partial\phi}{\partial r}(r,\theta)+g(r)r\phi(r,\theta)dr\right)d\theta \tag{A.11}$$

$$= \lim_{\epsilon\to 0}\int_0^{2\pi}\left([-rG_0(r)\phi(r,\theta)]_\epsilon^\infty+\int_\epsilon^\infty\frac{d}{dr}\left(r\frac{d}{dr}G_0(r)\right)\phi(r,\theta)+G_0(r)r\phi(r,\theta)\right)drd\theta \tag{A.12}$$

$$= \lim_{\epsilon\to 0}\int_0^{2\pi}\left(\left(\underbrace{\epsilon G_0'(\epsilon)}_{2c/\pi}\phi(\epsilon,\theta)\right)+\int_\epsilon^\infty\underbrace{\left(\frac{1}{r}\frac{d}{dr}\left(r\frac{dG_0}{dr}\right)+G_0(r)\right)}_{=0}\phi(r,\theta)\right)drd\theta \tag{A.13}$$

$$= 2\pi\frac{2c}{\pi}\phi(0) \Rightarrow c = \frac{1}{4}\frac{2m}{\hbar^2}. \tag{A.14}$$

B. Matlab implementation of the net refinement algorithm

netref.m

```
function [nTRI,coords] = netref(TRI,coords,fun)
%NEWNETREF Refine a triangular mesh
%{
Inputs:
-TRI    : dim=[3,N] an array of indices of vertices
            that make up the triangles.
            Each line corresponds to one triangle.
            Either clockwise or counter-clockwise.
-coords : dim=[2,M] an array of coordinates of vertices.
            Each line corresponds to one vertex.
-fun    : @(x,v) a function that controlls edge lengths
            at point x along direction v.

Outputs:
-nTRI   : dim=[3,NN] the refined mesh triangle array.
-coords : dim=[2,MM] the vertices of the refined mesh.
%}

extTRI = [TRI,TRI(:,1)];
% TRI array with the first column repeated for easier indexing.
tcoords = []; % temporary coordinate array

p = zeros(0,4); % Array of new vertices on the old edges.
% p(1:2,:) indices of the endpoints of an edge,
% p(3:4,:) indices of the first and last new vertices in coords.
k = length(coords); % Index where the new vertices of coords start
% Now we start looping over the old array of triangles
%to add new vertices on edges where applicable.
for i = 1:size(TRI,1)
    for j = 1:3
        iprev = logical((extTRI(i,j) == p(:,2)) ...
            &(extTRI(i,j+1) == p(:,1)));
        % If we have calculated on this edge before,
        % iprev has exactly one nonzero value
        if any(iprev)
            p = [p;extTRI(i,j),extTRI(i,j+1), p(iprev,4), p(iprev,3)];
```

```
        else
            % We calculate the coordinates of the new verices
            nn = newnodes(coords(extTRI(i,j),:), ...
                coords(extTRI(i,j+1),:),fun);
            ni = size(nn,1);
            % Add the new nodes to the temporary coordinate array
            tcoords = [tcoords;nn];
            if ni == 0 % No new node
                p = [p;extTRI(i,j),extTRI(i,j+1), NaN, NaN];
            else
                p = [p;extTRI(i,j),extTRI(i,j+1), k+1, k + ni];
            end
            k = k + ni;
        end
    end
end
% Add all the new nodes to the old coordinate array
coords = [coords;tcoords];
nTRI = [];
locTRIe = cell(1,3);
% Cell holding all the new and old vertices on each edge of a triangle

for i = 1:size(TRI,1)
    for j = 1:3
        iprev = ((extTRI(i,j) == p(:,1)) & (extTRI(i,j+1) == p(:,2)));
        if isnan(p(iprev,3)) % No new vertices added to edge
            locTRIe{j} = [extTRI(i,j),extTRI(i,j+1)];
        elseif p(iprev,3) == p(iprev,4) % One new vertice on edge
            locTRIe{j} = [extTRI(i,j), p(iprev,3), extTRI(i,j+1)];
        else
            locTRIe{j} = [extTRI(i,j), ...
                p(iprev,3):sign(p(iprev,4)-p(iprev,3)):p(iprev,4),...
                extTRI(i,j+1)];
        end
    end
    % Now we arrange the order of the cells of locTRIe so that
    %locTRIe{newo(3)} contains the fewest new vertices while
    %maintaining a clockwise or counter-clockwise ordering.
    [~,ii] = min([length(locTRIe{1}), length(locTRIe{2}),...
        length(locTRIe{3})]);
    newo = circshift(1:3,[0,-ii]); % The new order of the edges
    %The main function
    [ttTRI,coords] = netrefrec(locTRIe{newo(1)},locTRIe{newo(2)},...
        locTRIe{newo(3)},coords,fun);

    nTRI = [nTRI;ttTRI];
end
end
```

newnodes.m

```
function nodes = newnodes(P1,P2,f)
%NEWNODES Make new nodes on an edge.
%{
Inputs:
-P1    : dim=[1,2] Starting coordinates of the edge
-P2    : dim=[1,2] End coordinates of the edge
-f     : @(x,v) function controlling spacing between new nodes
Outputs:
-nodes  : dim=[2,X] The nodes added to the edge.
%}

Q = P1;
v = (P2-P1)/norm(P2-P1); %Unit array of the edge
Qi = 0;
while (all(Q>=min(P1,P2)) && all(Q<=max(P1,P2))) || (Qi == 0)
    Q = Q+v*f(Q,v);
    Qi = Qi+1;
end
Q = P2;
rQi = 0;
while (all(Q>=min(P1,P2)) && all(Q<=max(P1,P2))) || (rQi == 0)
    Q = Q-v*f(Q,-v);
    rQi = rQi+1;
end

i = max(Qi,rQi);
w = (P2-P1)/i;
nodes = [P1(1)+w(1)*(1:i-1)',P1(2)+w(2)*(1:i-1)'];
end
```

netrefrec.m

```
function [nTRI,coords] = netrefrec(a,b,c,coords,f)
%NEWNETREFMAIN Main loop of netref
%{
Inputs:
-a,b,c  : array containg vertices on edge.
          c contains the fewest vertices.
-coords : dim=[2,M] an array of coordinates of vertices.
          Each line corresponds to one vertex.
-f      : @(x,v) a function that controlls edge lengths
          at point x along direction v.

Outputs:
-nTRI   : dim=[3,NN] the refined mesh triangle array.
-coords : dim=[2,MM] the vertices of the refined mesh.
%}
na = length(a); nb = length(b); nc = length(c);
nv = [na,nb,nc];
if sum(nv == 2) > 1 %Check if two edges contain only 2 vertices each
    if sum(nv == 2) == 3 % Trivial case
```

```
        nTRI = [a(1),b(1),c(1)];
    else % Only one edge contains more than two vertices.
        % We create the new triangles by connecting the vertices
        % of this edge to the opposing corner.
        if na>nb
            nTRI = zeros(na-1,3);
            for j = 1:na-1
                nTRI(j,:) = [a(j) a(j+1) b(2)];
            end
        else
            nTRI = zeros(nb-1,3);
            for j = 1:nb-1
                nTRI(j,:) = [b(j) b(j+1) c(2)];
            end
        end
    end
else
    % Connect the vertices in A and B farthest away from each other
    % (excluding endoints) and find new points on that edge.
    nodes = newnodes(coords(b(nb-1),:),coords(a(2),:),f);
    nn = [b(nb-1),(1+size(coords,1)):(size(nodes,1)+size(coords,1)),...
        a(2)];
    coords = [coords;nodes];
    % Now we have to again arrange the edges left so that
    % locTRIe{newo(3)} contains the fewest vertices.
    [~,ii] = min([na-1,nb-1,length(nodes)]);
    newo = circshift(1:3,[0,-ii]); % The new order of the edges
    locTRIe = {a(2:end),b(1:end-1),nn};
    % Now we call the main loop recursivly
    [temp1TRI,coords] = netrefrec(locTRIe{newo(1)},...
        locTRIe{newo(2)},locTRIe{newo(3)},coords,f);
    % The new edge and edge c form a trapeziod.
    % We connect the vertices on hose edges to form new triangles.
    temp2TRI = traptri(c,nn,coords);

    nTRI = [temp1TRI;temp2TRI];
end

end
```

traptri.m

```
function nTRI = traptri(a,b,coords)
%TRAPTRI Make new nodes on an edge.
%{
Inputs:
-a,b    : dim=[1,2] edges containing extra vertices
-coords : dim=[1,2] an array of coordinates of vertices.
          Each line corresponds to one vertex

Outputs:
```

```
—nTRI : dim=[3,NN] the refined triangles.
%}

na = length(a); nb = length(b);
k = 1;
ia = 1; ib = 1;
% We loop over the vertices on each edge and connect them to form
% triangles.
while ia<na || ib<nb
    if ia == na
        nTRI(k,:) = [b(ib+1),b(ib),a(ia)];
        ib = ib + 1;
    elseif ib == nb
        nTRI(k,:) = [a(ia),a(ia+1),b(ib)];
        ia = ia + 1;
    else
        % We check to see wich connection between vertices creates a
        % triangle with the shortest edges.
        l1 = norm(coords(a(ia+1),:)—coords(b(ib),:));
        l2 = norm(coords(a(ia),:)—coords(b(ib+1),:));
        if l1>l2
            nTRI(k,:) = [b(ib+1),b(ib),a(ia)];
            ib = ib + 1;
        else
            nTRI(k,:) = [a(ia),a(ia+1),b(ib)];
            ia = ia + 1;
        end
    end
    k = k+1;
end
end
```