Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Iterative Methods for Solving the Schrödinger Equation on a Rectangular Scattering Region

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

ANNE MARKENSTEIJN

Delft, Nederland
Mei 2014

BSc verslag TECHNISCHE WISKUNDE

"Iterative Methods for Solving the Schrödinger Equation on a Rectangular Scattering Region"

ANNE MARKENSTEIJN

Technische Universiteit Delft

Begeleiders

Dr. A. R. Akhmerov
Dr. N. V. Budko
Dr. M. T. Wimmer

Overige commissieleden

Prof. dr. ir. A. W. Heemink Dr. B. van den Dries

Dr. Y. M. Blanter

Mei, 2014 Delft

Abstract

In this bachelor thesis, we demonstrate the application of iterative methods to the solution of the Schrödinger equation defined in a rectangular mesoscopic scattering region. We find that the Induced Dimension Reduction method (IDR(s)) is the best iterative method for large systems whereas the General Minimalized Residual method (GMRES) is the best method for small systems, out of the four tried methods: GMRES, restarted GMRES (GMRES(l)), the BiConjugate Gradient Stabilized method (BiCGStab) and IDR(s). We also show that, although preconditioning can improve the convergence behaviour of GMRES, the preconditioner proposed in this thesis, which is similar to a shifted Laplacian preconditioner, is not optimal. Furthermore, we observe that the dependence of the number of iterations N on the spatial extent of the domain, is related to the spatial decay factor of the corresponding Green's function. We also find that N is very small if the energy of the system is chosen to be outside the bandwidth of the spectrum of the Hamiltonian, and we show that this is due to the fact that the system matrix becomes diagonally dominant. Finally we show that introducing a random potential into the system results in an increase of the number of iterations with respect to a clean system for a system in the diffusive or in the ballistic regime, and we find that the number of iterations required to reach convergence becomes constant when the system enters the localized regime.

Contents

1 Introduction

Numerical analysis is a widely used method to solve the Schrödinger equation in a mesoscopic tight binding model of different geometries. These systems are now mainly solved by using direct methods, which solve a linear system of equations in a finite sequence of operations, and in the absence of rounding errors these methods will result in the exact solution. The main problem with direct methods is that they use a lot of memory to compute the required operations.

In this bachelor thesis, we will investigate the use of iterative methods to solve for a part of the Green's function corresponding to the Schrödinger equation defined in a rectangular scattering region. Iterative methods solve the system by approximating the solution, so that with every iteration, the approximation comes closer to the exact solution. In general, the operations used to compute the approximate solution require less memory than the operations used by a direct method. Because they require less memory, these methods can be used to solve systems that are too big for direct methods.

In order to investigate the application of iterative methods to solve the Schrödinger equation in a tight binding model, we will consider a rectangular scattering region with two leads attached to it. We will first derive a linear system of equations, in Section 2, to solve for a part of the Green's function corresponding to this scattering region. In Section 3 we will analyse the spectra of all the matrices that make up the system matrix of our problem. Using the properties found from the spectral analysis, we will apply four different iterative methods in Section 4. We will consider the Generalized Minimal Residual method (GMRES) in Section 4.1, restarted GMRES (GMRES(l)) in Section 4.2, the Biconjugate Gradient Stabilized methods (BiCGStab) in Section 4.3 and finally we will consider the Induced Dimension Reduction method (IDR(s)) in Section 4.5. To accelerate the convergence of GMRES, we will consider the application of a preconditioning matrix in Section 4.4. Furthermore, we will compare the performance of these different iterative methods in Section 5. Finally, we will introduce a random potential into our scattering region resulting in a disordered system and discuss the performance of iterative methods for a disorered system in section 6.

2 System

As mentioned in the introduction, we consider a square quantum billiard, which consist of two infinite leads connected to one conductor, see Figure 1. The leads are assumed to be ballistic conductors [1] (i.e. conductors with no scattering). This means that waves can enter freely from the leads into the conductor, and after the waves have scattered inside the conductor they can leave again through one of the leads. Because the scattering only takes place inside the conductor and not in the leads, the conductor is also referred to as the scattering region. In general, the electrons inside the scattering region obey the Schrödinger equation [1]:

$$\varepsilon\Psi = H_{op}\Psi + S \tag{1}$$

In which ε is the energy of the wave function Ψ, S is a source term representing a wave coming in from one of the leads and H_{op} is the Hamiltonian operator

$$H_{op} = \frac{-\hbar^2\nabla^2}{2m} + U(\mathbf{r}) \tag{2}$$

with e the electronic charge $(-1.6 \cdot 10^{-19}C)$, m is the effective mass, $\hbar = \frac{h}{2\pi}$ in which h is Planck's constant $(6.63 \cdot 10^{-34}Js)$ and U is the confining potential.
The formal solution of this problem can be written as:

$$\Psi = GS \tag{3}$$

where G is the corresponding Green's function.

$$G = [\varepsilon - H_{op}]^{-1} \tag{4}$$

Until the boundary conditions are prescribed, this problem is not uniquely solvable. We choose to have Dirichlet boundary conditions $\Psi = 0$ on the sides of the scattering region without leads, and open boundaries on the sides of the scattering region that are attached to the leads, see Figure 1. There is a unique solution that satisfies both the

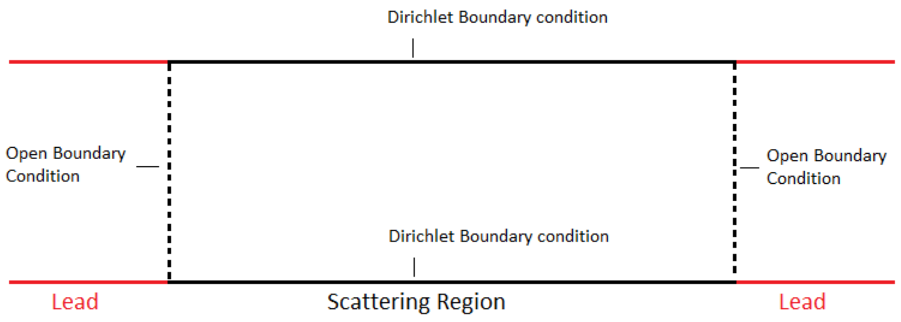


Figure 1: The scattering region (black) with the boundary conditions and part of the infinite leads (red).

Dirichlet boundary conditions and the open boundary conditons; the retarted Green's function G^R. In a one dimensional case, the retarded Green's function corresponds to two waves going from the point of excitation to minus and plus infinity. We incorporate the boundary conditions into the equation itself, which is done by adding an infinitesimal (positive) imaginary part η to the energy [1]. If we insert the imaginary part, we obtain the following retarded Green's function (since only the retarded Green's function is considered, G^R is simply denoted by G and will be referred to as the Green's function):

$$G = [\varepsilon - H_{op} + i\eta]^{-1} \tag{5}$$

In order to calculate the Green's function, we write equation 5 as a differential equation for the Green's function [1],

$$[\varepsilon - H_{op}(\mathbf{r}) + i\eta]G(\mathbf{r};\mathbf{r'}) = \delta(\mathbf{r} - \mathbf{r'}) \tag{6}$$

in which $\mathbf{r}$ is an arbitrary point inside the scattering region and $\mathbf{r'}$ is the excitation point.
We solve this differential equation by discretizing the spatial coordinates to lattice points (see Figure 2), which turns the Green's function and the Hamiltonian operator into matrices

$$G(\mathbf{r};\mathbf{r'}) \to G(i,j) \tag{7}$$

$$H_{op} \to H \tag{8}$$

where the indices i and j represent points on the discrete lattice and H represents the matrix notation of the differential operator H_{op}. Equation 5 then becomes a matrix

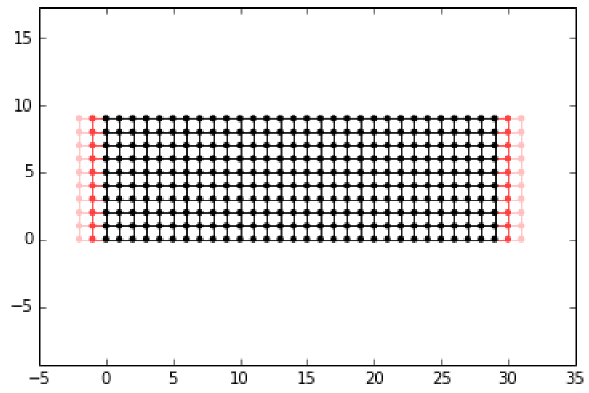


Figure 2: Discretized representations of the system with a width of 10 lattice points, and a length of 30 lattice points. The black points represent the scattering region and the red points represent the leads.

equation

$$[(\varepsilon + i\eta)I - H]G = I \tag{9}$$

with I the identity matrix. To be able to solve for G, we must have the matrix representation H of H_{op}. In order to make such a representation, a two dimensional case is considered. The Hamiltonian operator (see equation 2) then becomes:

$$H_{op} = -\frac{\hbar^2}{2m}\nabla^2 + U(x,y) \tag{10}$$

To obtain the matrix representation, we let H_{op} work on a arbitrary function $f(x,y)$. We then make a uniform discrete lattice in which the points are located at $x = k\Delta x$ and $y = l\Delta x$ with $k, l \in \mathbb{Z}$. This gives:

$$[H_{op}f]_{x=k\Delta x, y=l\Delta x} = \left[-\frac{\hbar^2}{2m}\nabla^2 f\right]_{x=k\Delta x, y=l\Delta x} + U_{k,l}f_{k,l} \tag{11}$$

with $f_{k,l} = f(x = k\Delta x, y = l\Delta x)$ and $U_{k,l} = U(x = k\Delta x, y = l\Delta x)$. To approximate the second derivatives of f, we use the finite difference method. Assuming Δx is small enough, the derivatives can be approximated by (see [3]):

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2}|_{x=k\Delta x, y=l\Delta x} &= \frac{1}{(\Delta x)^2}\Delta^2_{0,x}f_{k,l} + \mathcal{O}((\Delta)^2)\\ \frac{\partial^2 f}{\partial y^2}|_{x=k\Delta x, y=l\Delta x} &= \frac{1}{(\Delta x)^2}\Delta^2_{0,y}f_{k,l} + \mathcal{O}((\Delta)^2)\end{aligned}$$

in which $\Delta^2_{0,x}$ is the central difference operator in the x-direction and in which $\Delta^2_{0,y}$ is the central difference operator in the y-direction. If we substitute these approximations in equation 11 we get:

$$[H_{op}f]_{x=k\Delta x, y=l\Delta x} = (U_{k,l} + 4t)f_{k,l} - tf_{k-1,l} - tf_{k+1,l} - tf_{k,l-1} - tf_{k,l+1} \tag{12}$$

where we defined $t \equiv \frac{\hbar^2}{2m(\Delta x)^2}$. From this expression, a matrix representation for the two dimensional Hamiltonian operator can be obtained. We get a matrix H in which all the elements are either $4t+U$, $-t$ or 0; all diagonal elements $h_{\gamma,\gamma}$ are equal to $4t+U$ and an off-diagonal element $h_{\gamma,\delta}$ equals $-t$ if a lattice point (i_γ, j_γ) and another lattice point (i_δ, j_δ) are either horizontal or vertical neighbours, otherwise an off-diagonal element equals 0. Since being neighbours is a commutative relation [3](if (i_γ, j_γ) is a neighbour of (i_δ, j_δ), then (i_δ, j_δ) is a neighbour of (i_γ, j_γ)) we have that $h_{\gamma,\delta} = h_{\delta,\gamma}$ for all $\gamma, \delta = 1, 2, \ldots, n$ and therefore H is a symmetric matrix.

Even though we now have a matrix representation of the Hamiltonian operator, we still cannot get the Green's function by simply inverting $[(\varepsilon - i\eta)I_H]$ because this matrix has infinite size. Truncating the matrix at some point is not a solution, because than you're solving a closed system with reflecting boundaries instead of the system with open boundaries that we are interested in. In order to overcome this problem, we are going to divide the overall Green's function into four submatrices; one submatrix $G_l \in \mathbb{C}^{\infty\times\infty}$ for the leads, one submatrix $G_s \in \mathbb{C}^{n\times n}$ for the scattering region, where n is the number of lattice points in the scattering region, and two submatrices $G_{ls} \in \mathbb{C}^{\infty\times n}$ and $G_{sl} \in \mathbb{C}^{n\times\infty}$ for the coupling between the leads and the scattering region, which turns equation 9 into [1]:

$$\left[\begin{array}{cc} G_l & G_{ls} \\ G_{sl} & G_s \end{array}\right]\left[\begin{array}{cc} (\varepsilon + i\eta)I - H_l & T_l \\ T_l^H & \varepsilon I - H_s \end{array}\right] = I \tag{13}$$

in which the matrix $[(\varepsilon + i\eta)I - H_l]$ represents the leads, $[\varepsilon I - H_s]$ represents the scattering region and the matrix $T_l \in \mathbb{C}^{\infty\times n}$ is the coupling matrix between the leads and the scattering region (We use $(...)^H$ to denote the conjugate transpose). This coupling matrix is only non-zero if point l_i in the leads is adjacent to point i in the scattering region, and for these adjacent points T_l has value t. Since we are only interested in the part of the Green's function inside the scattering region, we are going to derive an explicit expression for G_s. From equation 13 we obtain:

$$[(\varepsilon + i\eta)I - H_l]G_{ls} + [T_l]G_s = 0 \tag{14}$$

and

$$[\varepsilon I - H_s]G_s + [T_l^H]G_{ls} = I \tag{15}$$

From equation 14 we get

$$G_{ls} = -F_l T_l G_s \tag{16}$$

with $F_l = [(\varepsilon + i\eta)I - H_l]^{-1}$ the Green's function for the isolated semi-infinite leads [1](since the overall Green's function was the retarded function, F_l is also the retarded function). If we now substitute equation 16 into equation 15, we obtain

$$[H_s - \varepsilon I + T_l^H F_l T_l]G_s = -I \tag{17}$$

This gives the Green's function for the entire scattering region without the infinite leads, but we are only interested in part of the scattering region. In particular, we are interested in the points of the scattering region which are adjacent to points in the leads (the black points next to a red point in Figure 2). To get the part of the Green's function we are interested in, we define a matrix $P_L \in \mathbb{C}^{n\times m}$, with n again the number of lattice points in the scattering region and m the number of lattice points in the scattering region that are next to a lattice point in one of the leads. The entries of P_L are zero, except for the entries (y, z) of P_L for which the corresponding lattice point (i, j) in the scattering region is adjacent to a point (l_i, l_j) in one of the leads, those entries have value -1. Since P_L essentially is a matrix which only contains some columns of $-I$ (only those we are interested in), we can rewrite the matrix equation to obtain the part of the Green's function we are interested in:

$$[H_s - \varepsilon I + T_l^H F_l T_l]G = P_L \tag{18}$$

where G is that part of the overall scattering Green's function G_s we are interested in. Using the same matrix P_L we can rewrite T_l as $T_l = T_l' P_L^H$, with $T_l \in \mathbb{C}^{\infty\times m}$. Substituting this in equation 18 and defining the self-energy matrix $H_E \equiv T_l' F_l T_l'^H$ we obtain :

$$[H_s - \varepsilon I + P_L H_E P_L^H]G = P_L \tag{19}$$

We use this equation to determine G, the part of the Green's function we are interested in. From now on, we will write this equation as a linear system

$$Ax = b \tag{20}$$

with $A = [H_s - \varepsilon I + P_L H_E P_L^H]$, x column j of matrix G and b column j of matrix P_L.

3 Spectral Analysis

In order to choose the right iterative method to solve this system and to be able to apply preconditioning, it is necessary to know the spectrum and the properties of the individual matrices and of the total system matrix A of equation 19. To be able to say something about the spectra and properties, we first define the matrix $\Gamma \equiv i(H_E - H_E^H)$ and show that Γ must be positive semidefinite. We will not give a thorough proof of this, but rather give a feeling of why Γ must be positive semidefinite.

Since the matrices H_s and H_E must keep the physical meaning of Hamiltonians, some constraints are imposed on the eigenvectors. For example, each eigenvector Ψ of a Hamiltonian H must remain normalisable at all times, therefore they must obey:

$$\frac{\partial|\Psi|^2}{\partial t} \leq 0 \tag{21}$$

Using $|\Psi|^2 = \Psi^H\Psi$ and the product rule, we can write this as

$$\frac{\partial|\Psi|^2}{\partial t} = \frac{\partial\Psi^H}{\partial t}\Psi + \Psi^H\frac{\partial\Psi}{\partial t} \tag{22}$$

We now rewrite the general time dependent Schrödinger equation to:

$$\frac{\partial\Psi}{\partial t} = \frac{1}{i\hbar}H\Psi = \frac{-i}{\hbar}H\Psi \tag{23}$$

where H is the effective Hamiltonian. In our system we have:

$$H = H_s + P_L H_E P_L^H \tag{24}$$

We can use the rewritten of the time dependent Schrödinger equation to obtain:

$$\frac{\partial\Psi^H}{\partial t} = (\frac{\partial\Psi}{\partial t})^H = (\frac{-i}{\hbar}H\Psi)^H = \frac{+i}{\hbar}\Psi^H H^H \tag{25}$$

Substituting the rewritten time dependent Schrödinger equation (equation 23) and equation 25 into the product rule (equation 22) we get:

$$\frac{\partial|\Psi|^2}{\partial t} = \left(\frac{i}{\hbar}\Psi^H H^H\right)\Psi + \Psi^H\left(\frac{-i}{\hbar}H\Psi\right) \tag{26}$$

As mentioned before, equation 21 must hold for all wave functions. So simplifying above expression and using the constraint from equation 21 gives:

$$-i\Psi^H(H - H^H)\Psi \leq 0 \quad \forall\ \Psi \tag{27}$$

or

$$\Psi^H i(H - H^H)\Psi \geq 0 \quad \forall\ \Psi \tag{28}$$

Using the effective Hamiltonian of our system (see equation 24) and using the fact that the system Hamiltonian H_s is real and symmetric, we get

$$i(H - H^H) = i(H_s + P_L H_E P_L^H - (H_s + P_L H_E P_L^H)^H) = P_L i(H_E - H_E^H)P_L^H = P_L\Gamma P_L^H \tag{29}$$

Substituting this in equation 28 gives

$$\Psi^H P_L \Gamma P_L^H \Psi \geq 0 \quad \forall\, \Psi \tag{30}$$

Since this is true for al Ψ, it follows that Γ is positive semidefinite.

Now that we have shown that Γ is positive semidefinite, we can use this to obtain constraints on the spectrum of the system matrix and on the spectra of the separate matrices of the system matrix. First look at the self-energy matrix H_E, which is a square complex $m \times m$ matrix, where m is the number of lattice points which are attached to a lead. Since Γ is positive semidefinite, we have

$$x^H \Gamma x \geq 0 \quad \forall\, x \in \mathbb{C}^m \tag{31}$$

Since equation 31 holds for all x, we can choose x such that it is an eigenvector of H_E, i.e. $H_E x = \lambda x$ with λ an eigenvalue of H_E, to get:

$$x^H \Gamma x = i x^H (H_E - H_E^H) x = i x^H H_E x - i x^H H_E^H x = i x^H \lambda x - i x^H \overline{\lambda} x \geq 0$$

where $\overline{\lambda}$ denotes the complex conjugate of λ. Using that $x^H x = |x|^2$ this simplifies to

$$x^H \Gamma x = i\lambda |x|^2 - i\overline{\lambda}|x|^2 \geq 0$$

Since λ is a complex number, we can write it as $\lambda = \lambda' + i\lambda''$ where λ' denotes the real part of λ and λ'' denotes the complex part of λ. Using this notation, we obtain

$$x^H \Gamma x = i[\lambda' + i\lambda'' - \overline{(\lambda' + i\lambda'')}]|x|^2 = 2 \cdot i^2 \cdot \lambda'' |x|^2 = -2\lambda''|x|^2 \geq 0$$

But since $|x|^2 \geq 0$, we must have $\lambda'' \leq 0$. Therefore the spectrum of H_E lies in the lower half plane, see Figure 3.

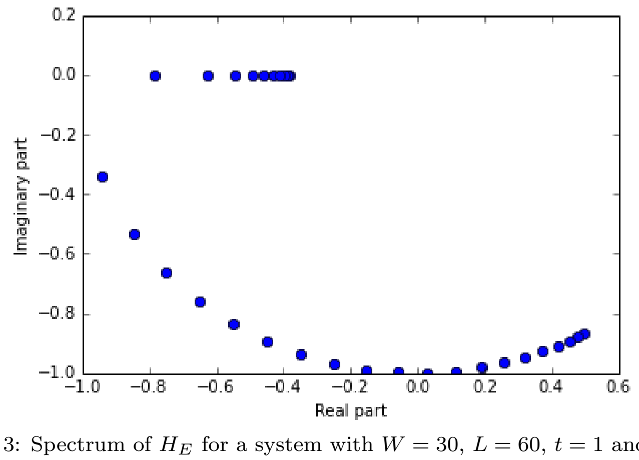


Figure 3: Spectrum of H_E for a system with $W = 30$, $L = 60$, $t = 1$ and $\varepsilon = 2$.

Looking at the anti-hermitian part of $M \equiv P_L H_E P_L^H$, which is a $n \times n$ complex

matrix, with n the number of lattice points, something similar can be shown.
First we rewrite the anti-hermitian part:

$$\frac{i}{2}(M - M^H) = \frac{i}{2}(P_L H_E P_L^H - P_L H_E^H P_L^H) = \frac{i}{2} P_L (H_E - H_E^H) P_L^H = \frac{1}{2} P_L \Gamma P_L^H$$

Then we choose y such that $P_L^H y = x$, or $y^H P_L = x^H$, to obtain:

$$\frac{i}{2} y^H (M - M^H) y = \frac{1}{2} y^H P_L \Gamma P_L y = \frac{1}{2} x^H \Gamma x$$

But we know that $x^H \Gamma x \geq 0$ for all x, which means that $\frac{i}{2} y^H (M - M^H) y \geq 0$ for all y. So we now choose an y such that it is an eigenvector of M, i.e. $My = \lambda y$ with λ an eigenvalue of M. Using the same simplifications that were used when looking at H_E gives:

$$\frac{i}{2} y^H (M - M^H) y = \frac{1}{2} \cdot (-2) \cdot \lambda'' |y|^2 = -\lambda'' |y|^2 \geq 0$$

Since $|y|^2 \geq 0$, we must have $\lambda'' \leq 0$. Therefore the spectrum of M lies in the lower half plane, see Figure 4. Comparing the spectrum of M (Figure 4) with the spectrum of

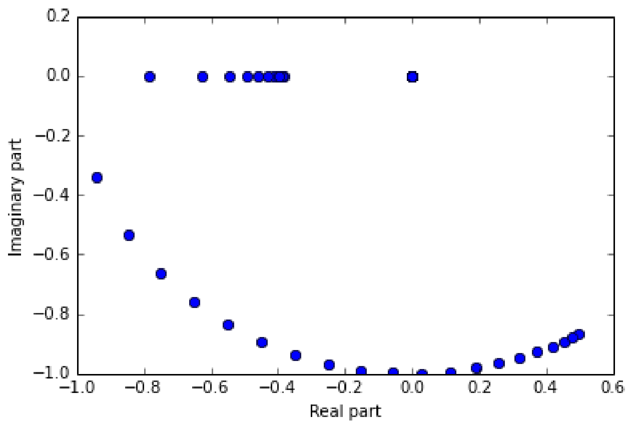


Figure 4: Spectrum of $P_L H_E P_L^H$ for a system with $W = 30$, $L = 60$, $t = 1$ and $\varepsilon = 2$.

H_E (Figure 3), we can see that they are identical, expect that the spectrum of M has an additional zero, which can be explained by the fact that P_L is essentially a projector.

Finally, we show that the same boundary holds for the spectrum of the system matrix $A = [H_s - \varepsilon I + P_L H_E P_L^H] = [H_s - \varepsilon I + M]$. Since H_s is real and symmetric and ε is a (real) scalar, we get the following for the anti-hermitian part of A:

$$\frac{i}{2}(A - A^H) = \frac{i}{2}[H_s - \varepsilon I + M - (H_s^H - E I^H + M^H)] = \frac{i}{2}(M - M^H)$$

We have already shown that $\frac{i}{2} y^H (M - M^H) y \geq 0$ for all y, so $\frac{i}{2} y^H (A - A^H) y \geq 0$ for all y. Choosing an y such that it is an eigenvector of A, i.e. $Ay = \lambda y$ with λ an eigenvalue of A, and using the same simplifications as before, we get:

$$\frac{i}{2} y^H (A - A^H) y = -\lambda'' |y|^2 \geq 0$$

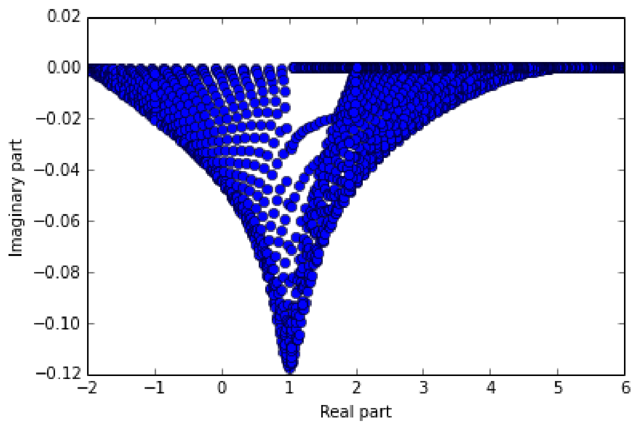


Figure 5: Spectrum of A for a system with $W = 30$, $L = 60$, $t = 1$ and $\varepsilon = 2$.

Since $|y|^2 \geq 0$, we must have $\lambda'' \leq 0$. Therefore the spectrum of A lies in the lower half plane.

In the first part of this section we have shown some boundaries for the imaginary part of the spectra, but it is also possible to show boundaries for the real part of the spectra. To obtain these boundaries, we first look at the system Hamiltonian H_s for a clean system, i.e $U \equiv 0$. As shown in section 2, the system Hamiltonian for such a system has entries $4t$ on the diagonal, and entries $-t$ or 0 elsewhere. To be more precise, every row of the Hamiltonian can contain at most four entries $-t$, since every lattice point can have at most four neighbours. This means that for H_s we have $h_{ii} = 4t$ and:

$$\sum_{j \neq i} |h_{ij}| \leq 4t \tag{32}$$

If we now use Gershgorin's circle theorem, we get:

$$|\lambda_{H_s} - h_{ii}| = |\lambda_{H_s} - 4t| \leq \sum_{j \neq i} |h_{ij}| \leq 4t \tag{33}$$

Rewriting equation 33 gives the following boundaries for the eigenvalues of the Hamiltonian:

$$0 \leq \lambda_{H_s} \leq 8t \tag{34}$$

These boundaries are also called the bandwidth of the system Hamiltonian and they are shown in Figure 6.

Looking at a part of the system matrix without the self-energy matrix, i.e looking at $[H_s - \varepsilon I]$, we can obtain similar boundaries. Since εI is a real diagonal matrix, we get (again with Gershgorin's circle theorem) for a clean system:

$$-\varepsilon \leq \lambda_{[H_s - \varepsilon I]} \leq 8t - \varepsilon \tag{35}$$

Numerical experiments show that the same boundaries hold for the real part of the eigenvalues of the entire system matrix A, see Figure 5.

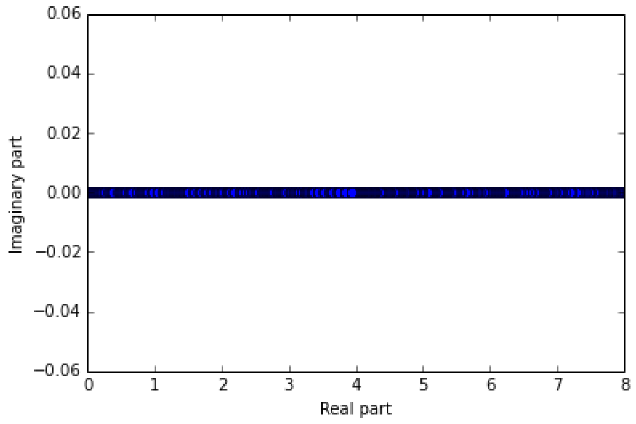


Figure 6: Spectrum of H_s for a system with $W = 30$, $L = 60$, $t = 1$ and $\varepsilon = 2$.

4 Iterative methods

As mentioned in the introduction, the Schrödinger equation is now mainly solved by using direct methods. These methods attempt to solve the problem by a finite sequence of operations in such a way that in the absence of rounding errors it would result in the exact solution. Iterative methods on the other hand, use an initial guess to generate successive approximations to a solution. Even in the absence of rounding errors, these methods would, in general, still not give the exact solution. But iterative methods are, in general, less expensive, meaning that they are able to solve larger systems than direct methods. In this section, four different iterative methods are discussed. All four methods are based on projection methods onto the Krylov subspace:

Considering a general linear system

$$Ax = b \tag{36}$$

where A is an $n \times n$ matrix and x and b are vectors of length n; a projection method seeks an approximate solution x_m from a m-dimensional subspace $x_0 + K_m$ of $\mathbb{C}^n$by using the following condition

$$b - Ax_m \perp L_m \tag{37}$$

in which L_m is another m-dimensional subspace of $\mathbb{R}^n$ [4]. This condition is called the Petrov-Galerkin condition. The iterative methods considered in this section use the Krylov subspace for K_m, which is a subspace of the form:

$$K_m(A, v_1) = span\{v_1, Av_1, A^2v_1, \dots, A^{m-1}v_1\} \tag{38}$$

with v_1 a vector of length n.

4.1 GMRES

As we have shown in the sections above, the system matrix A is not symmetric, hermitian or normal, and apart from the boundaries we found on the eigenvalues of A, the spectrum is rather spread out. Therefore, we try to solve the system using the Generalized Minimal Residual Method (GMRES), which is a method that has no requirements for A and which can solve most problems.

GMRES is based on a projection method onto the Krylov subspace. This method takes $L_m = AK_m$ in which K_m is the Krylov subspace with $v_1 = r_0/|r_0|_2$, where $r_0 = b - Ax_0$ is the initial residual vector. Taking $L_m = AK_m$ results in a method which minimizes the 2-norm of the residual vector $b - Ax$ over all $x \in x_0 + K_m$ [4]. This norm can also be written as:

$$|b - Ax|_2 = |b - A(x_0 + V_m y)|_2 = |\beta e_1 - \overline{H}_m y|_2 \tag{39}$$

with V_m the $n \times m$ matrix of the orthogonal basis vectors of K_m, y any vector of length m, $\beta = |r_0|_2$, e_1 the first column of the $n \times n$ identity matrix and $\overline{H}_m$ the $(m+1) \times m$ Hessenberg matrix with nonzero entries h_{ij} obtained when finding an orthogonal basis for the Krylov subspace (in general, the column vectors of the Krylov subspace are not orthogonal). For this orthogonalisation process, any orthogonalisation process can be used, for example Arnoldi or Gram-Schmidt.

The 2-norm of the residual vector (equation 39) is minimized if $x_m = x_0 + V_m y_m$, where $y_m = \text{argmin}_y |\beta e_1 - \overline{H}_m y|_2$. So GMRES basically works in three steps. First it determines an orthogonal basis for the Krylov subspace using for example Arnoldi. During this orthogonalisation process, the matrices V_m and $\overline{H}_m$ are formed. Secondly, GMRES computes y_m, using the $\overline{H}_m$ obtained by the orthogonalisation process. Finally, GMRES uses this y_m and the matrix V_m obtained earlier to compute the final approximation x_m. (For the entire derivation and the complete algorithm see [4]).

The convergence of this method for our system is plotted in Figure 7. Since the the right hand side of the system P_L is actually a matrix, the system is solved per column of P_L. This figure shows that the convergence behaviour of GMRES is very poor, since most of the iterative steps do not decrease the relative residual by much, it needs a number of iterations almost equal to the system size to reach convergence. Since GMRES orthogonalises the vector of the Krylov subspace, all these vectors must be stored. Therefore, GMRES requires a memory storage of approximately $\mathcal{O}(mn)$ with m the number of iterations required and n the system size. For our case we have $m \approx n$ and therefore a memory requirement of approximately $\mathcal{O}(n^2)$.

4.2 Restarted GMRES

Because of the memory requirements, GMRES is rather impractical if the system becomes big. One way to solve this, is to use restarted GMRES (GMRES(l)). This method basically does the same as non-restarted GMRES, except that the algorithm restarts after l iterations if convergence is not reached. GMRES(l) then sets $x_0 = x_l$ and start again until convergence is reached. The advantage of restarting is that instead of having to save all the vectors of the Krylov subspace, only l vectors are required. Therefore, the memory requirements for restarted GMRES are reduced to $\mathcal{O}(ln)$. One

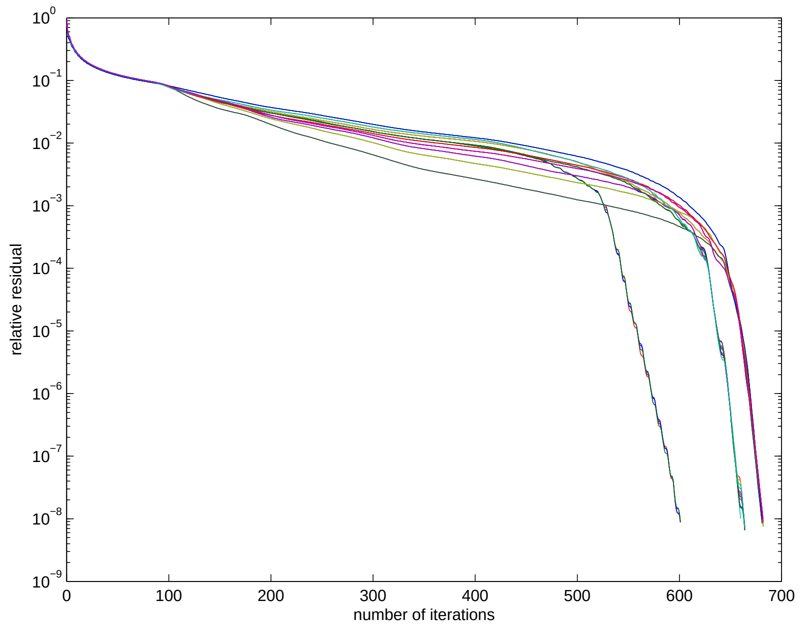


Figure 7: Convergence plot of GMRES for a system of size 1000 ($W = 20$, $L = 50$), $t = 1$, $\varepsilon = 2$ and the tolerance is 10^{-8}. Each color represents the convergence for one column of P_L.

drawback of restarted GMRES is that it does not guarantee convergence if A is not positive semi-definite, whereas GMRES always convergences in at most n steps [4]. Another difficulty of GMRES(l) is finding the optimal l.

The convergence of this method for our system is plotted in Figure 8. Even tough the convergence behaviour for GMRES(l) is linear instead of the poor behaviour shown for GMRES, the number of iterations required is worse (approximately 10 times the number of iterations required for non-restarted GMRES).

4.3 BiCGStab

As mentioned above, GMRES shows very poor convergence behaviour. Therefore we try another method, the Biconjugate Gradient Stabilized method (BiCGStab), since this method is also used to solve systems with non-symmetrical system matrices.
BiCGStab is a Krylov subspace projection process with

$$K_m = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}$$

where $v_1 = r_0/||r_0||$, and

$$L_m = span\{w_1, A^H w_1, \ldots, (A^H)^{m-1}w_1\}$$

with w_1 any vector which obeys $(v_1, w_1) = 0$, where (x,y) denotes the inner product of x and y.
BiCGStab is based on the Biconjugate Gradient method (BCG) which uses the same subspaces K_m and L_m (see [4], for the entire derivation of BiCGStab and for the

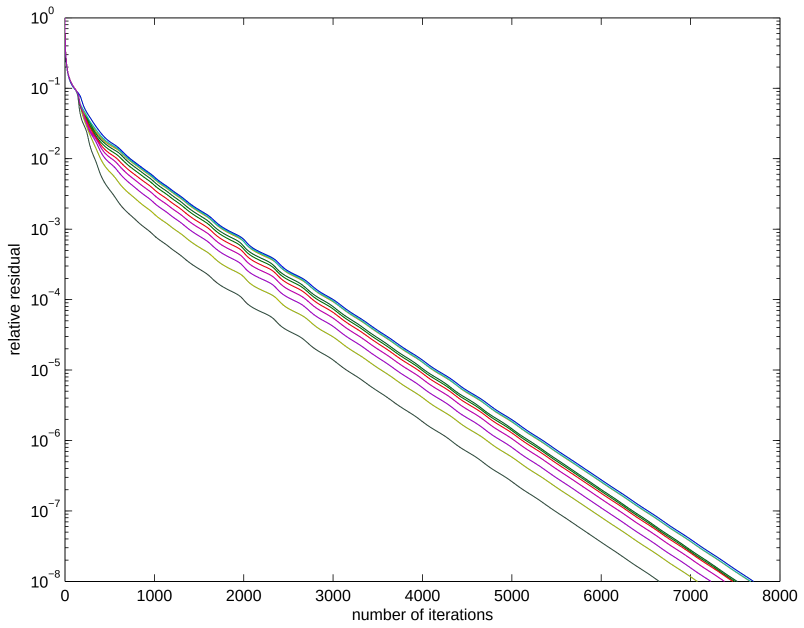


Figure 8: Convergence plot of GMRES(l) with $l = 10$ for a system of size 1000 ($W = 20$, $L = 50$), $t = 1$, $\varepsilon = 2$ and the tolerance is 10^{-8}. Each color represents the convergence for one column of P_L.

complete algorithms of both BiCGStab and BCG). BCG is meant for solving systems with a nonsymmetric system matrix A, and therefore uses both A and A^H. This means that BCG solves the original system $Ax = b$, but it implicitly also solves the dual system $A^H x^* = b^*$. The BCG method writes the solution vector and the residual vector of the original system as follows

$$\begin{aligned} x_{j+1} &= x_j + \alpha_j p_j \\ r_{j+1} &= r_j - \alpha_j A p_j \end{aligned}$$

with $j \leq m$. Here α_j is a scalar and p_j is a vector which indicates the search direction for the next residual vector. This search direction is a linear combination of r_j and p_{j-1}:

$$p_{j+1} = r_{j+1} + \beta_j p_j \tag{40}$$

For the dual system there are similar expressions for the residual vector and the search direction.

$$\begin{aligned} r^*_{j+1} &= r^*_j - \alpha_j A^H p^*_j \\ p^*_{j+1} &= r^*_{j+1} + \beta_j p^*_j \end{aligned}$$

BCG then requires that the residual vectors are biorthogonal and that the search directions are biconjugate with respect to A, i.e

$$\begin{aligned} (r_j, r^*_i) &= 0, \text{ for } i \neq j \\ (Ap_j, p^*_i) &= 0, \text{ for } i \neq j \end{aligned}$$

Applying this constraint to r_j and p_j and using some simplifications, results in an expression for both α and β:

$$\alpha_j = \frac{(r_j, r_j^*)}{(Ap_j, p_j^*)}$$
$$\beta_j = \frac{(r_{j+1}, r_{j+1}^*)}{(r_j, r_j^*)}$$

So when an initial r_0, r_0^*, p_0 and p_0^* are chosen, these relations can be used to determine the solution vector x_m.
A drawback of the BCG method is that it uses both A and A^H and that it is not very stable. The aim of the BiCGStab method is to solve both of these problems. In order to do this, BiCGStab uses the fact that the residual vector and the search direction of the BCG algorithm can be written as

$$r_j = \phi_j(A)r_0$$
$$p_j = \pi_j(A)r_0$$

with ϕ_j a polynomial of degree j which satisfies $\phi_j(0) = 1$ and $\pi_j(A)$ a polynomial of degree j. In order to stabilize the method and get rid of A^H, BiCGStab uses different expression for both the residual vector and the search direction:

$$r_j = \phi_j(A)\psi_j(A)r_0$$
$$p_j = \psi_j(A)\pi_j(A)r_0$$

where $\phi_j(A)$ and $\pi_j(A)$ are the polynomials found from the BCG method and $\psi_j(A)$ is a polynomial which is defined recursively at each step with the goal of stabilizing the method:

$$\psi_{j+1}(t) = (1 - \omega_j t)\psi_j(t) \tag{41}$$

where ω_j is a scalar. Defining $s \equiv r_j - \alpha_j Ap_j$ and using these equations results in the following expressions.

$$r_{j+1} = (I - \omega_j A)s_j$$
$$p_{j+1} = r_{j+1} + \beta_j(I - \omega_j A)p_j$$

with

$$\omega_j = \frac{(As_j, s_j)}{(As_j, As_j)}$$
$$\alpha_j = \frac{(r_j, r_0^*)}{(Ap_j, r_0^*)}$$
$$\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}$$

The solution vector can be found by using the following expression:

$$x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j \tag{42}$$

So in order to solve a linear system of equations, BiCGStab first chooses initial values, $r_0 = b - Ax_0$, r_0^* is arbitrary and $p_0 = r_0$, then it iterates over the above expressions until convergence is reached. In one iteration, BiCGStab first calculates α_j, then s_j and then ω_j. Using these values, BiCGStab computes x_{j+1} and r_{j+1}, after which it calculates β_j. Finally BiCGStab calculates p_{j+1} and then it starts a new iteration.

The convergence of this method for our system is plotted in Figure 9. This figure shows that BiCGStab, unlike GMRES, shows almost linear convergence behaviour. But even though the convergence behaviour of BiCGStab is considerably better than for GMRES, the number of iterations required for BiCGStab to reach convergence is approximately 4 times the system size whereas GMRES requires at most a number of iterations equal to the system size. Comparing BiCGStab to GMRES(l), we can see that BiCGStab shows similar convergence behaviour to GMRES(l), except that BiCGStab requires approximately half the number of iterations of GMRES(l). Since

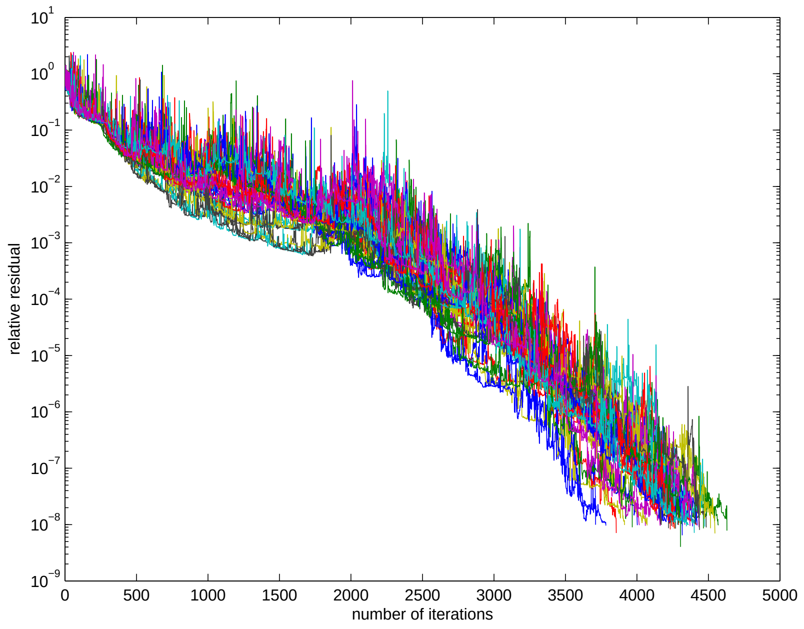


Figure 9: Convergence plot of BiCGStab for a system of size 1000 ($W = 20$, $L = 50$), $t = 1$, $\varepsilon = 2$ and the tolerance is 10^{-8}. Each color represents the convergence for one column of P_L.

BiCGStab only has to store x_j, p_j and r_j which are all of size $1 \times n$, it requires $\mathcal{O}(3n)$ of memory. Therefore BiCGStab might be preferred over GMRES for large systems, despite the large number of iterations.

4.4 Preconditioning

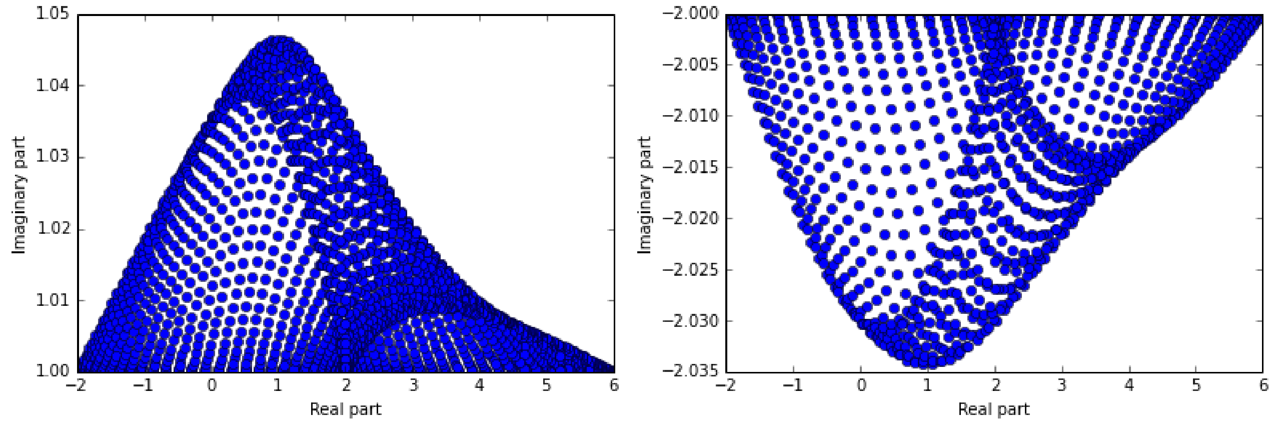
BiCGStab is considered as a way of improving the convergence behaviour. Another way of doing this for any Krylov based iterative solver is using preconditioning. This is a method that tries to change the original linear system into one that has the same solution, but is easier to solve. Changing the system is done by multiplying with a preconditioning matrix M, also called a preconditioner. Instead of trying to solve the original system, we will try to solve the preconditioned system

$$M^{-1}Ax = M^{-1}b \tag{43}$$

The Krylov subspace for the preconditioned system is a subspace of the form

$$K_m = span\{M^{-1}v_1, M^{-1}AM^{-1}v_1, \dots, (M^{-1}A)^{m-1}M^{-1}v_1\} \tag{44}$$

This means that we require a solution of a linear system with the matrix M, i.e of $Mx = v_1$, at each step of the iterative solver. In order to make preconditioning worthwhile, we require that this linear system $Mx = v_1$ is solved in a very small number of iterations [4].



(a) $\varepsilon = 2 - 2i$ (b) $\varepsilon = 2 + 2i$

Figure 10: Spectrum of A for a system with negative imaginary part (a) and positive imaginary part (b), both with $W = 30$, $L = 60$ and $t = 1$

Instead of looking at a system with strictly real energy ε, which we have done so far, we are going to consider a system in which the energy is allowed to be imaginary, and we can than write this energy as $\varepsilon = \alpha_1 + \beta_1 i$ with $\alpha_1, \beta_1 \in \mathbb{R}$. Since we now have an imaginary value for the energy, the boundaries as found in section 3 for the spectrum of the system matrix A do no longer hold. Numerical experiments (see Figure 10) show the following new boundaries for the real part of the spectrum of A:

$$\alpha_1 \leq \lambda'_A \leq 8t - \alpha_1 \tag{45}$$

and the following boundaries for the imaginary part of A:

$$\begin{cases} \lambda''_A \geq -\beta_1, & \text{if } \beta_1 < 0 \\ \lambda''_A \leq -\beta_1, & \text{if } \beta_1 \geq 0 \end{cases} \tag{46}$$

On this more general system, we are going to try a preconditioner of the form

$$M = H_s + P_L H_E P_L^H - \varepsilon_2 I \tag{47}$$

which is similar to the shifted Laplacian preconditioner proposed in [2]. Here, ε_2 is a complex scalar which can be written as $\varepsilon_2 = \alpha_2 + i\beta_2$ with $\alpha_2, \beta_2 \in \mathbb{R}$. As shown in [2], the eigenvalues σ of the preconditioned system matrix, i.e of $M^{-1}A$, lie in a half-plane

$$-\beta_1\sigma' + (\alpha_1 - \alpha_2)\sigma'' + \beta_1 \geq 0 \tag{48}$$

if $\beta_2 = 0$. This can be seen in Figure 11. If $\beta_2 \neq 0$ the shape of the spectrum of the

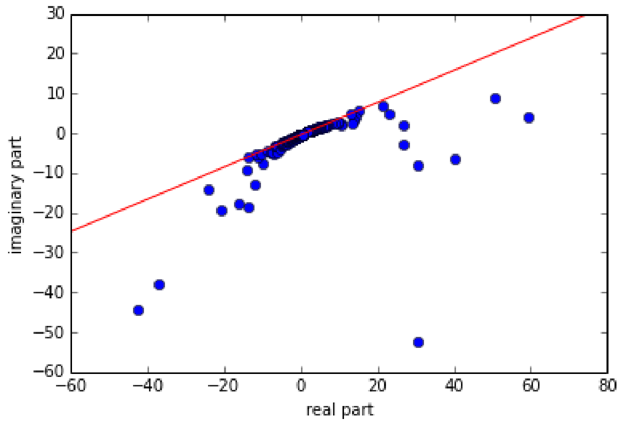


Figure 11: The spectrum of $M^{-1}A$ for system with $W = 10$, $L = 30$, $t = 1$, $\varepsilon = 2 - i$ and $\varepsilon_2 = 2|\varepsilon|$. The red line shows the edge of the half plane.

preconditioned system also depends on the spectrum of A itself. For $\lambda''_A \geq 0$, i.e for $\beta_1 < 0$, the spectrum of $M^{-1}A$ lies inside or on a circle with center c and radius R if $\beta_2 < 0$, and the spectrum lies outside or on this circle if $\beta_2 > 0$ [2]. For the center and the radius of the circle, the following relation holds:

$$c = \frac{\varepsilon - \overline{\varepsilon_2}}{\varepsilon_2 - \overline{\varepsilon_2}} = \left(\frac{\beta_2 + \beta_1}{2\beta_2}, \frac{\alpha_2 - \alpha_1}{2\beta_2}\right) \tag{49}$$

and

$$R = \left|\frac{\varepsilon_2 - \varepsilon}{\varepsilon_2 - \overline{\varepsilon_2}}\right| = \sqrt{\frac{(\beta_2 - \beta_1)^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2}} \tag{50}$$

These two cases can be seen in Figure 12 For $\lambda''_A \leq 0$, i.e for $\beta_1 \geq 0$, the spectrum of $M^{-1}A$ lies inside or on a circle with center c and radius R if $\beta_2 > 0$, and the spectrum lies outside or on this circle if $\beta_2 < 0$ [2]. This is shown in Figure 13.

According to [2], the following holds for the norm of the residual after the k-th iterations using GMRES

$$\frac{|r_k|_2}{|r_0|_2} \leq a\left(\frac{R}{|c|}\right)^k \tag{51}$$

with a a constant. This means that the further away the circle is from the origin and the smaller the circle is, the better the convergence behaviour. In the case we are most interested in, which is the case with strictly real E, this causes a problem. For E real we have $\beta_1 = 0$ and therefore

$$|c| = \frac{(\beta_2 + \beta_1)^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2} = \frac{\beta_2^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2} \tag{52}$$

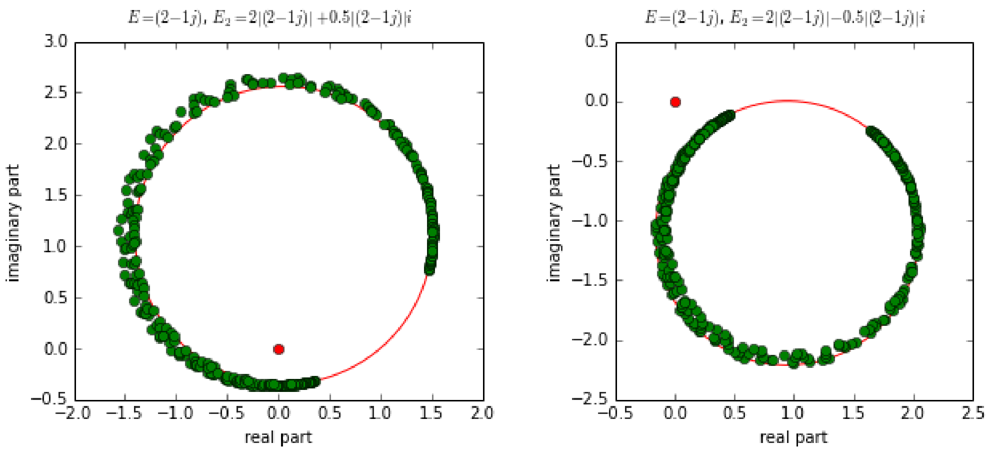


Figure 12: Spectra of the preconditioned system with $W = 10$, $L = 30$ and $t = 1$. The red line show the circle with center c and radius R. The red dot shows the origin.

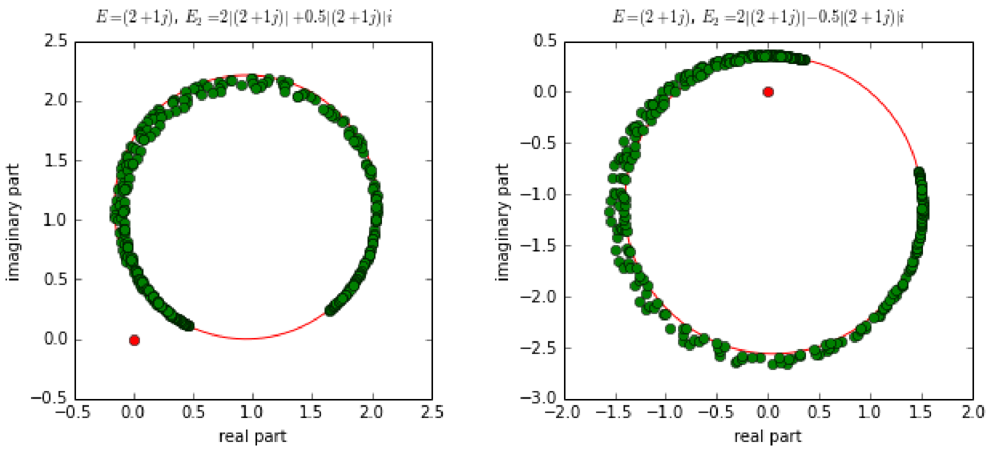


Figure 13: Spectra of the preconditioned system with $W = 10$, $L = 30$ and $t = 1$.The red line show the circle with center c and radius R. The red dot shows the origin.

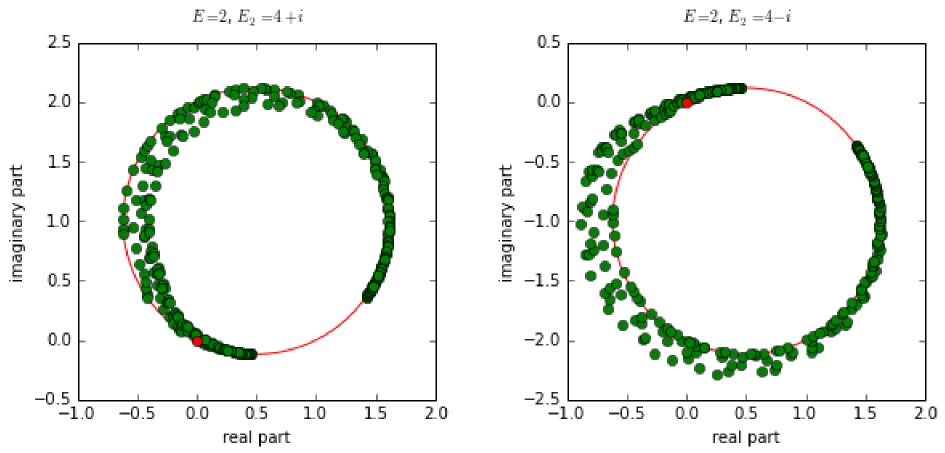
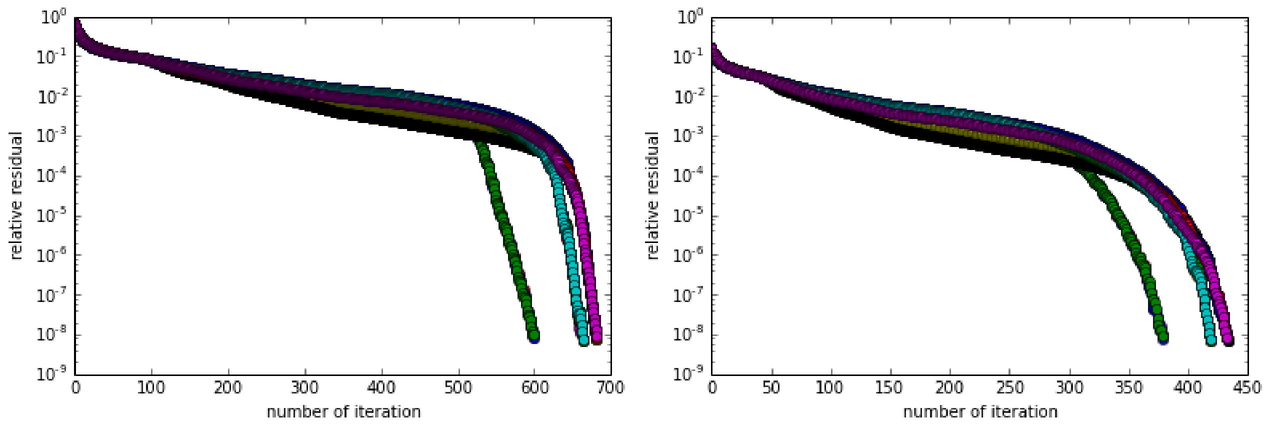


Figure 14: Spectra of the preconditioned system with $W = 10$, $L = 30$ and $t = 1$.The red line show the circle with center c and radius R. The red dot shows the origin.



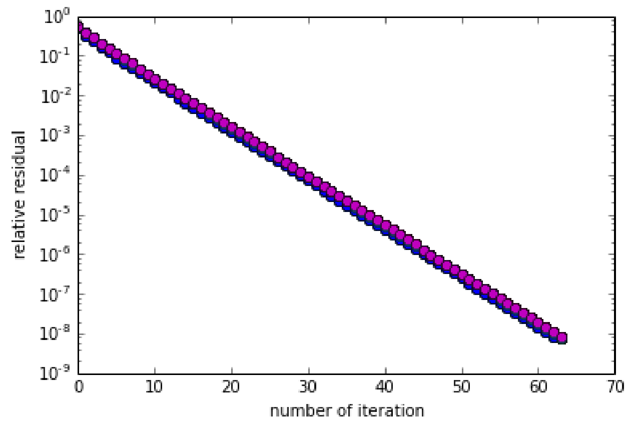
(a) Unpreconditioned system (b) Preconditioned system

Figure 15: Convergence plots for the unpreconditioned system (a) and the preconditioned system (b) with $W = 20$, $L = 50$, $E = 2$, $t = 1$, $\varepsilon_2 = 2i$ and the tolerance is 10^{-8}. Each color represents the convergence for one column of P_L.

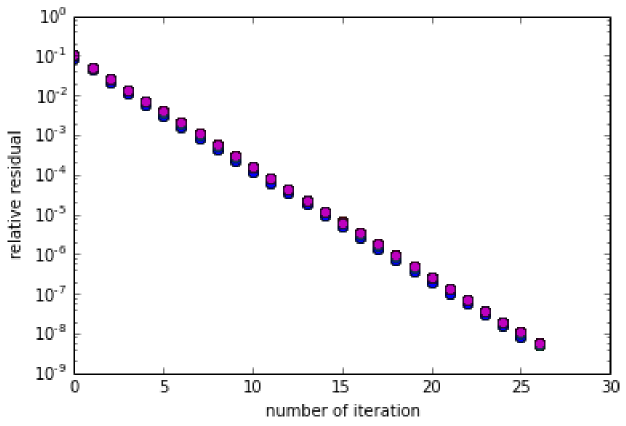
and

$$R = \frac{(\beta_2 - \beta_1)^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2} = \frac{\beta_2^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2} \tag{53}$$

From this we obtain $|c| = R$, so the distance from the origin to the centre of the circle is equal to the radius, which means the origin always lies on the circle (see Figure 14). Furthermore we can see that $\frac{R}{|c|} = 1$, so $\frac{|r_k|_2}{|r_0|_2} \leq a$, which means that the residual norm does not necessarily get smaller after the k-th iteration. Looking at Figure 15, we can see that the convergence behaviour is not improved, even though a smaller amount of iterations is needed. The smaller amount of iterations needed by the preconditioned system is due to the fact that the spectrum of the preconditioned system is less spread out than the spectrum of the unpreconditioned system. But since the convergence behaviour itself does not change, there might be a preconditioner that is better suited for a system with strictly real energy.



(a) Unpreconditioned system



(b) Preconditioned system

Figure 16: Convergence plots for the unpreconditioned system (a) and the preconditioned system (b) with $W = 20$, $L = 50$, $\varepsilon = 2 - i$, $t = 1$, $\varepsilon_2 = -|\varepsilon|i$ and the tolerance is 10^{-8}. Each color represents the convergence for on column of P_L.

For a system with imaginary ε and $\beta_1\beta_2 > 0$, the origin lies outside the circle [2]. In that case, this choice of preconditioner might improve the convergence behaviour. Looking at Figure 16, we can see that for imaginary values of energy, the preconditioned system show no change in convergence behaviour apart from needing a smaller amount of iterations to convergence. Therefore this precondioner might also not be the most optimal choice in the case of imaginary energy.

4.5 IDR(s)

As we have seen, BiCGStab shows better convergence behaviour than GMRES, but the number of iterations required for BiCGStab is large. Furthermore, the considered preconditioner does not improve the convergence behaviour of GMRES enough. Therefore, we try the Induced Dimension Reduction method (IDR(s)). This method is expected to behave the same as BiCStab or even better if s becomes larger.
IDR(s) is based on generating residuals that are forced to be in subspaces of decreasing dimension (see [5] for the entire derivations and complete algorithm). These subspaces are nested such that $\cdots \subset G_j \subset G_{j-1} \subset \cdots \subset G_0$ with G_0 the full Krylov subspace $K_n(A, v_0)$. They are related to each other by

$$G_j = (I - \omega_j A)(S \cap G_{j-1}) \tag{54}$$

where S is any fixed subspace of $\mathbb{C}^n$ such that G_0 and S do not share a nontrivial invariant subspace of A, and where ω_j is a nonzero scalar. Since $G_j = \{0\}$ for some $j \leq n$, forcing the residual vectors to be in these subspaces G_j ensures that the residual becomes smaller.
In general, a iterative method that uses Krylov subspaces can be described by the

following recursive relations for the solution vector x_m and the residual vector r_m:

$$r_{m+1} = r_m - \alpha A v_m - \sum_{l=1}^{\hat{l}} \gamma_l \Delta r_{m-l} \tag{55}$$

$$x_{m+1} = x_m + \alpha v_m - \sum_{l=1}^{\hat{l}} \gamma_l \Delta x_{m-l} \tag{56}$$

where α is a scalar, $\Delta u_k = u_{k+1} - u_k$ is the forward difference operator, $\hat{l}$ is an integer which is called the depth of the recursion, v_m is a vector in $K_m(A, r_0) \backslash K_{m-1}(A, r_0)$ and $\gamma_1, \ldots, \gamma_{\hat{l}}$ are coefficients. IDR(s) generates residual r_m that are forced to be in the subspaces G_j, with j non-decreasing when m increases. We have $r_{m+1} \in G_{j+1}$, if:

$$r_{m+1} = (I - \omega_{j+1} A) v_m = v_m - \omega_{j+1} A v_m \tag{57}$$

with $v_m \in G_j \cap S$. If we choose

$$v_m = r_m - \sum_{l=1}^{\hat{l}} \gamma_l \Delta r_{m-l} \tag{58}$$

we get the following recursion relation for the residual

$$r_{m+1} = r_m - \omega_{j=1} A v_m - \sum_{l=1}^{\hat{l}} \gamma_l \Delta r_{m-l} \tag{59}$$

which is in accordance with the recursion relation found for a general iterative method that uses Krylov subspaces. If we now define $S \equiv N(Q^H)$, the null-space of a $n \times s$ matrix $Q = (q_1\ q_2\ \ldots\ q_s)$, where n is the system size and $q_1, q_2, \ldots, q_s \in \mathbb{C}^n$ are linearly independent vectors, we must have $v_m \in S = N(Q^H)$, since $v_m \in S \cap G_j$. Therefore v_m satisfies

$$Q^H v_m = 0 \tag{60}$$

From this equation, we can obtain an expression for the coefficients $\gamma_1, \ldots, \gamma_{\hat{l}}$. If we substitute the expression for v_m from equation 58 into equation 60, we obtain:

$$Q^H v_m = Q^H r_m - \sum_{l=1}^{\hat{l}} \gamma_l Q^H \Delta r_{n-l} = 0 \tag{61}$$

which can be rewritten as a matrix equation:

$$\left[Q^H \Delta r_{m-1} | \; Q^H \Delta r_{m-2} | \; \ldots | \; Q^H \Delta r_{m-\hat{l}} \right] c = Q^H r_m \tag{62}$$

where $c = (\gamma_1\ \gamma_2\ \ldots \gamma_{\hat{l}})^T$. This matrix equation is an $s \times \hat{l}$ linear system of equations for the coefficients $\gamma_1, \ldots, \gamma_{\hat{l}}$. Since the vectors $q_1, q_2, \ldots, q_s \in \mathbb{C}^n$ are linearly independent, this system is uniquely solvable if $\hat{l} = s$. This means that in order to calculate the first vector in G_{j+1}, i.e. calculate r_{m+1}, we must have $s + 1$ vectors in G_j (namely, the s

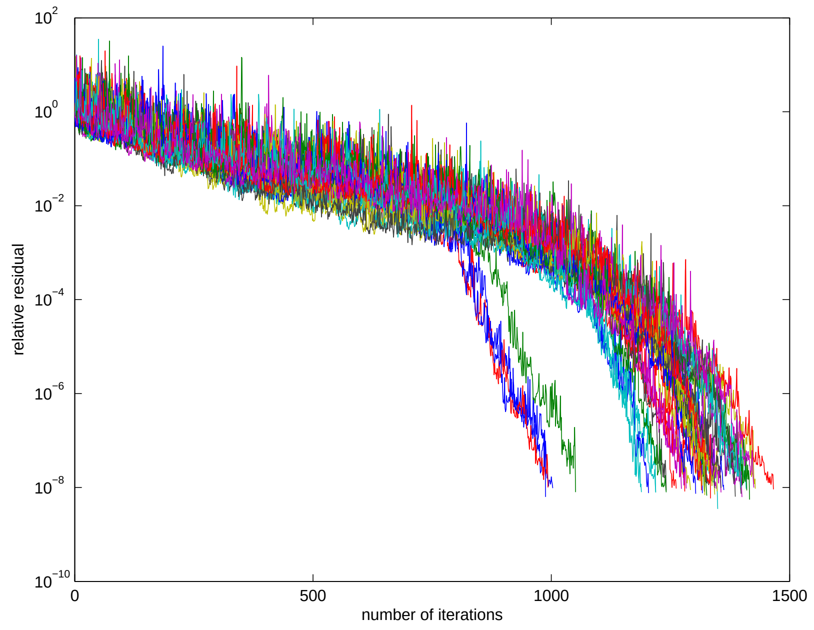


Figure 17: Convergence plot of IDR(8) for a system of size 1000 ($W = 20$, $L = 50$), $t = 1$, $\varepsilon = 2$ and the tolerance is 10^{-8}. Each color represents the convergence for one column of P_L.

vectors needed for $\left[Q^H \Delta r_{m-1} |\ Q^H \Delta r_{m-2} |\ \ldots |\ Q^H \Delta r_{m-s}\right]$ plus one vector r_m for the right hand side of equation 62).
If we define the following matrices:

$$\Delta R_m = [\Delta r_{m-1} |\ \Delta r_{m-2} |\ \ldots |\Delta r_{n-s}] \tag{63}$$
$$\Delta X_m = [\Delta x_{m-1} |\ \Delta x_{m-2} |\ \ldots |\Delta x_{n-s}] \tag{64}$$

We can then compute $r_{m+1} \in G_{j+1}$ by first calculating c from $(Q^H \Delta R_m)c = Q^H r_m$, then computing $v = r_m - \Delta R_m c$ and finally computing $r_{m+1} = v - \omega_{j+1} Av$. The solution vector $x_m + 1$ can be obtained from:

$$x_{m+1} = x_m - \Delta X_m c + \omega_{j+1} v \tag{65}$$

So we only need the scalar ω_{j+1}. For the calculation of the first residual in G_{j+1}, the scalar ω_{j+1} may be chosen freely, but it must remain the same for the calculations of all the other residual vectors in G_{j+1}. For G_0 we choose $\omega = \frac{v^H r_m}{v^H v}$ and for all the other G_j we choose $\omega = \frac{t^H v}{t^H t}$ with $t = Av$.

The convergence of this method for our system is plotted in Figure 17. This figure shows that IDR(8) has convergence behaviour between that of GMRES and that of BiCGStab, as expected (see also Figure 18). Since IDR(s) essentially only has to store ΔR_m which has size $n \times s$ and ΔX_m which also has size $n \times s$, the memory requirements for IDR(s) are approximately $\mathcal{O}(2sn)$.

5 Performance of iterative methods for a clean system

In order to choose the best iterative method for a system, we will compare the convergence histories of the different methods and look into the dependence for the convergence on the system size and on the energy. Because of the large amount of iterations needed for restarted GMRES, we will not consider this method. Looking at Figure 18

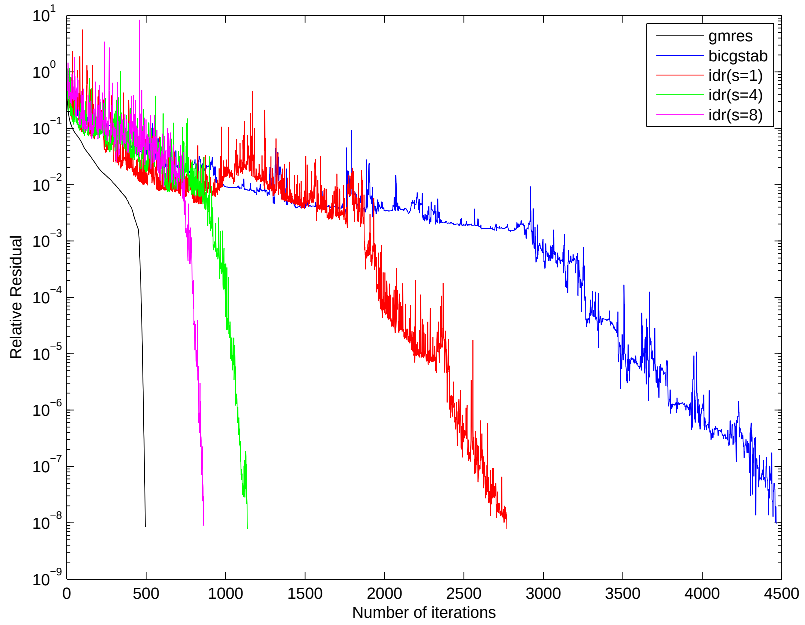


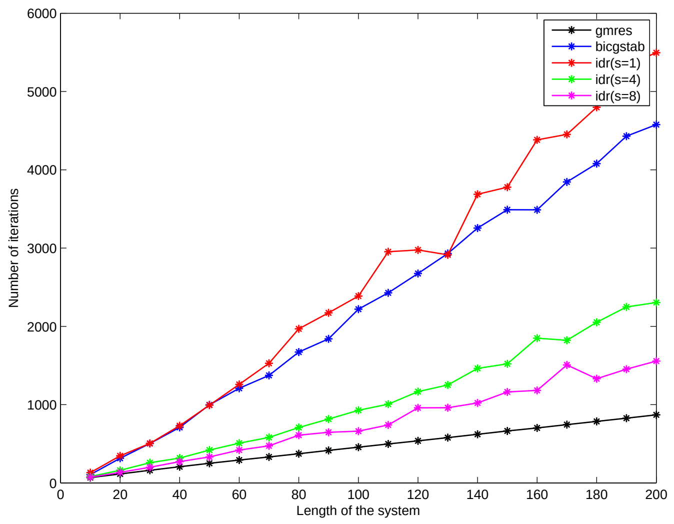
Figure 18: Convergence history of GMRES, BiCGStab and IDR(s) for a system with $W = 20$, $L = 50$, $\varepsilon = 2$, $t = 1$ and the tolerance is 10^{-8}. The convergence for only one column of P_L is shown.

we can see that BiCGStab and IDR(1) show similar convergence behaviour. As s gets larger, the convergence history of IDR(s) approaches that of GMRES, which requires the smallest number of iterations. But since the memory requirement for GMRES are $\mathcal{O}(n^2)$, and for IDR(s) they are $\mathcal{O}(2sn)$, we prefer IDR(s) over GMRES if the system is large while we prefer GMRES for small systems.

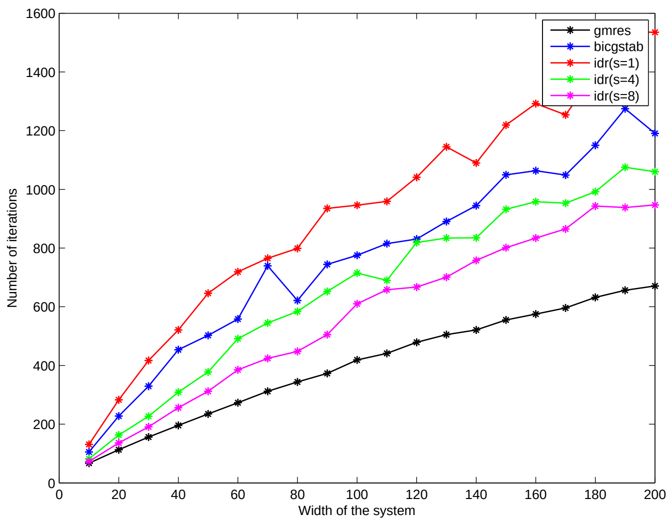
5.1 System Size

In Figure 19, the number of iterations is plotted as a function of the system size for all methods except GMRES(l). In accordance with Figure 18, we can see in Figure 19 that BiCGStab and IDR(1) require approximately the same amount of iterations and that the number of iterations needed for IDR(s) approach the number of iterations needed for GMRES if s gets larger.

If we look at Figure 20b, we can see that the Green's function is nonzero throughout the entire scattering region. We therefore expect the number of iterations N to depend on WL for $W \ll L$, see Figure 21. If we look at Figure 22 and Figure 19b, we can



(a) Dependence of iterations on length ($W = 10$)



(b) Dependence of iterations on width ($L = 10$)

Figure 19: Dependence of iterations of different methods on the length (a) and the width (b) of a system with $t = 1$, $\varepsilon = 1$ and the tolerance is 10^{-8}.

see that, for $W \gg L$, the number of iterations has a shape similar to $\sqrt{W}$. Figure 20a shows that the Green's function decays (in the W direction) at a rate of $\frac{1}{\sqrt{W}}$, which might correspond to the $\sqrt{W}$ found from Figure 22 and Figure 19b. This gives:

$$N \sim \begin{cases} \sqrt{W}, & \text{if } W \gg L \\ WL, & \text{if } W \ll L \end{cases} \tag{66}$$

In other words, the number of iterations shows similar behaviour to the spatial decay factor of the corresponding Green's function.

5.2 Energy

In Figure 23, the number of iterations as a function of energy is plotted for all the methods except GMRES(l). We can see that BiCGStab and IDR(1) require approximately the same amount of iterations, as was the case for the dependence on the system size. Also in accordance with the dependence on the system size is that IDR(s) approaches GMRES as s gets larger. Considering memory requirements, we again prefer IDR(s) over GMRES for a larger system. Looking at Figure 23, we can see that the number of iterations is very small if $\varepsilon < 0$ or $\varepsilon > 8t$. In section 3 we found for the system Hamiltonian the following boundaries

$$0 \leq \lambda'_{H_s} \leq 8t \tag{67}$$

If we now take ε to be outside the bandwidth (i.e $\varepsilon < 0$ or $\varepsilon > 8t$) we get that

$$|a_{ii}| = |4t - \varepsilon| > 4t = \sum_{j \neq i} a_{ij} \tag{68}$$

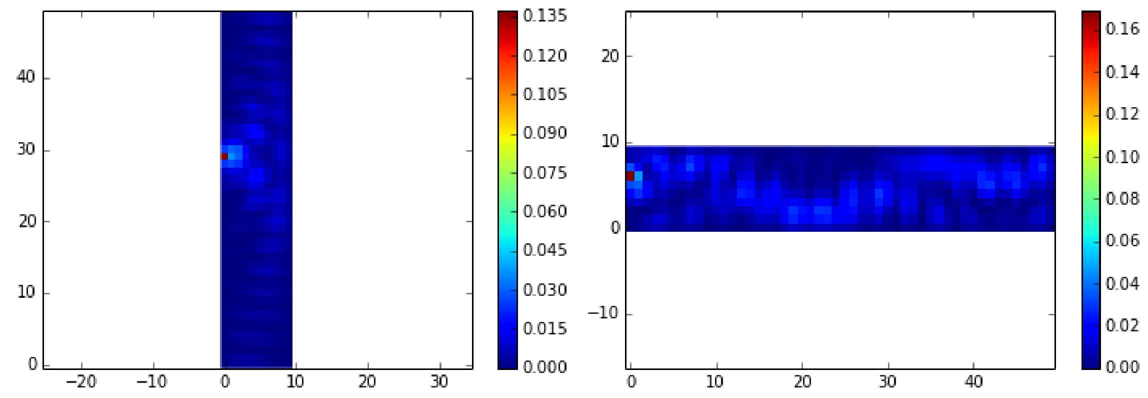
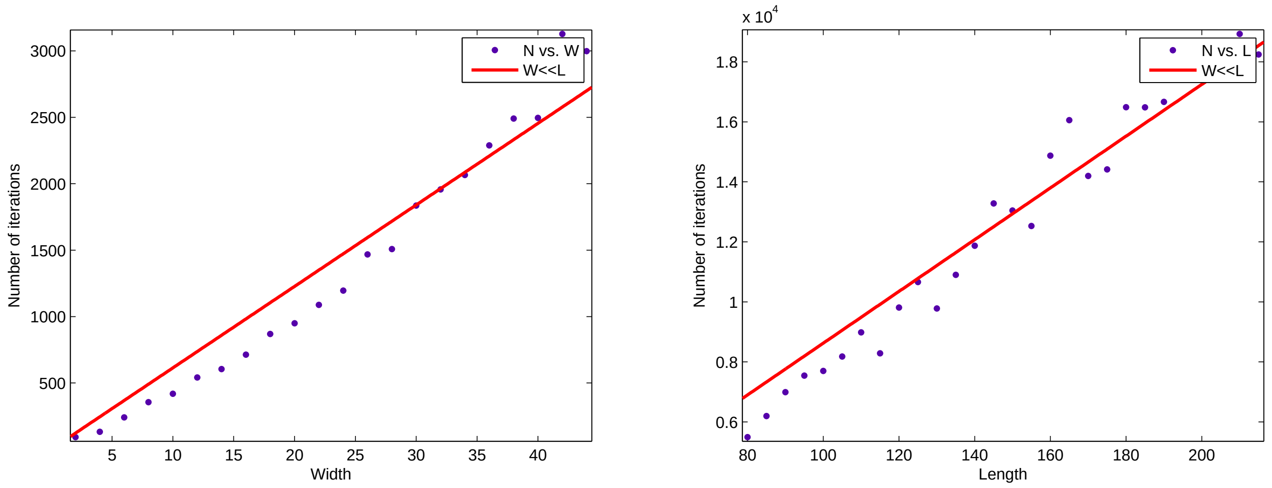

(a) Green's function for $W \gg L$ (b) Green's functions for $W \ll L$

Figure 20: The absolute value squared of the solution vectors of $Ax = b$ for a system (a) with $W = 50$, $L = 10$ and a system (b) with $W = 10$, $L = 50$. Both systems have $t = 1$ and $\varepsilon = 1$.



(a) $L = 60$ (b) $W = 60$

Figure 21: Fitted curves for the region where $W \ll L$ and a system with $\varepsilon = 1$ and $t = 1$. The data points show the convergence for IDR(8).

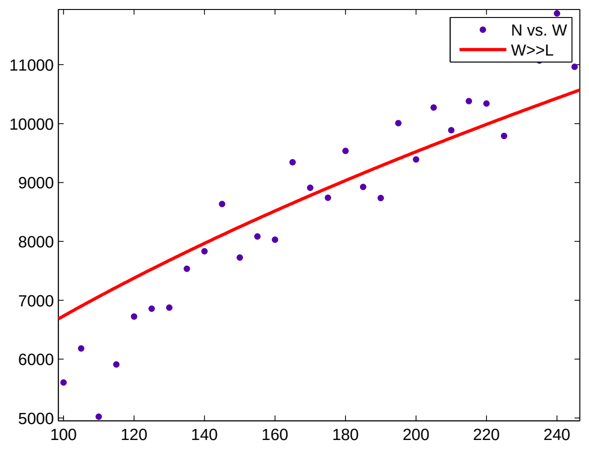
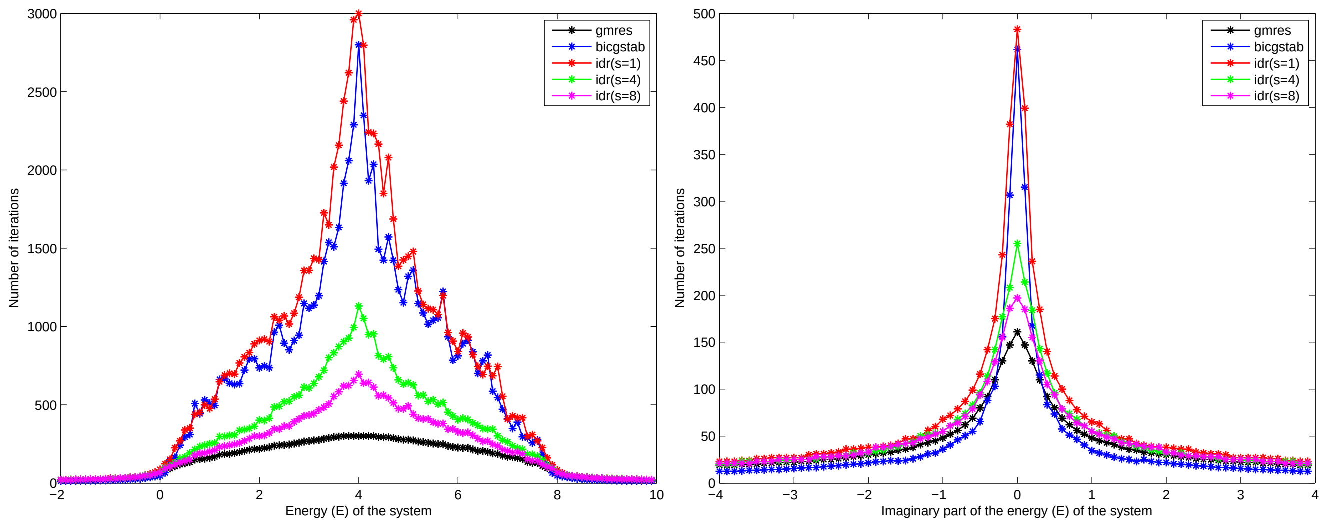


Figure 22: Fitted curve for the region where $W \gg L$ and a system with $L = 60$, $\varepsilon = 1$ and $t = 1$. The data points show the convergence for IDR(8).



(a) Real energy (b) Imaginary energy with $\varepsilon' = 1$

Figure 23: Dependence of iterations of different methods on the energy of a system with $W = 10$, $L = 30$, $t = 1$ and the tolerance is 10^{-8}.

with a_{ij} is an element of the system matrix A on column j and row i. From this we can conclude that the system matrix A becomes diagonally dominant, which results in fast convergence.

6 Performance of iterative methods for a disordered system

Until now, we have only considered a clean system (i.e $U(x, y) \equiv 0$). In this section, we will consider a system with a random potential, which means the system becomes disordered. For a disordered system, the wavesfunction decay exponentially:

$$\Psi \sim e^{-x/\xi} \tag{69}$$

where x is the coordinate in horizontal direction and ξ is the localization length. If we consider a general semiconductor, or a general scattering region, we can distinguish three different types: a ballistic conductor, a diffusive conductor and an insulator. A ballistic conductor has no scattering, which results in a conductor with zero resistance. A diffusive conductor shows some scattering (i.e has a finite resistance), but the electrons can still pass through the conductor. In an insulator, the scattering or disorder is too big for the electrons to pass trough the entire conductor, and we say that the wave functions become localized. A conductor is said be in the ballistic regime if $L \ll l_m$ where $l_m$is the mean free path, it is said to be in the localized regime if $L \gg \xi$, and a conductor is said to be in the diffusive regime if it has a length between the mean free path and the localization length [1]. The following holds for the mean free path and the for the localization length:

$$l_m \approx 48\frac{\sqrt{\varepsilon}}{U^2} \tag{70}$$

$$\xi \approx Ml_m = \frac{W\sqrt{\varepsilon}}{\pi}l_m \tag{71}$$

with $M = \frac{W\sqrt{\varepsilon}}{\pi}$ the number of transverse modes.

In order to investigate the performance of the iterative methods for a disordered system, we consider a disordered system with $W = 10$ and changing length and solve this system with IDR(8). We can rewrite the disordered system in the following form:

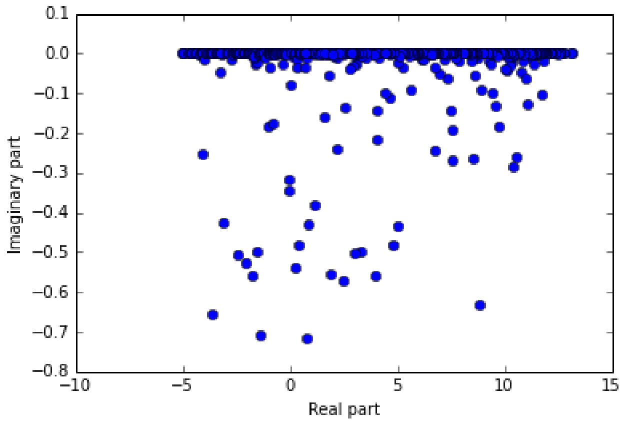
$$\tilde{A}x = b \tag{72}$$

in which $\tilde{A}$ is the system matrix of the disordered system which is defined as

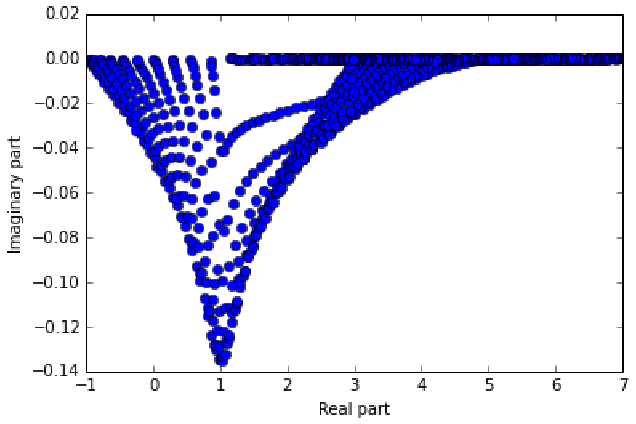
$$\tilde{A} \equiv H_s + P_L H_E P_L^H - D \tag{73}$$

with H_s the Hamiltonian of a clean system, $P_L H_E P_L^H$ the self-energy matrix of a clean system and D a diagonal matrix with entries d_{ii}, $i = 1, \ldots, n$. These entries are obtained by drawing from a uniform distribution $d_{ii} \in [\varepsilon - U/2, \varepsilon + U/2]$.
Using the boundaries found in section 3 for the real part of the spectrum of $H_S - P_L H_E P_L^H$ and using Gershgorin's circle theorem we can write

$$|\lambda_{H_S+P_L H_E P_L^H} - a_{ii}| = |\lambda_{H_S-P_L H_E P_L^H} - 4t| \leq \sum_{j\neq i} |a_{ij}| \leq 4t \tag{74}$$



(a) Spectrum of a disordered system



(b) Spectrum of a clean system

Figure 24: Spectra of a clean (a) and disordered (b) system, both with $W = 20$, $L = 50$, $t = 1$, $\varepsilon = 1$ and $U = 15$.

where a_{ij} is an element on row i and column j of the matrix $H_S - P_L H_E P_L^H$. From this we can obtain a boundary for the spectrum of the disordered system. Notice that $\tilde{a}_{ij} = a_{ij}$ for $i \neq j$ since we only add a diagonal matrix, and therefore for every $i = 1, \dots n$ we have

$$|\lambda_{\tilde{A}} - \tilde{a}_{ii}| = |\lambda_{\tilde{A}} - 4t - d_{ii}| \leq \sum_{j \neq i} |\tilde{a}_{ij}| \leq 4t \tag{75}$$

This gives the following boundaries for the real part of the eigenvalues of $\tilde{A}$:

$$\min_i \{d_{ii}\} \leq \lambda'_{\tilde{A}} \leq 8t + \max_i \{d_{ii}\} \tag{76}$$

where λ' denotes the real part of λ.
Looking at these boundaries (equation 76), we can conclude that the spectrum of the disordered system is more spread out than the spectrum of a clean system, see also Figure 24, and we can conclude that this spreading becomes larger for bigger U. We therefore expect that the number of iterations necessary to solve the disordered system is larger than the number of iterations required for a clean system, and that the number of iterations grows if U gets bigger.
But if $L \gg \xi$, localization occurs, which results in exponentially decaying wave functions. Therefore the wave function becomes smaller than the tolerance δ of the iterive method if x becomes bigger than some x_l, i.e:

$$\Psi|_{x_l} \sim e^{-x_l/\xi} = \delta \tag{77}$$

For $x \geq x_l$ we have that $\Psi \leq \delta$, and therefore the initial vector $x_0 = 0$ of the iterative method is within the tolerance for this region, and the number of iterations N is equal to zero. For the entire region where $x < x_l$ we have that:

$$N \sim cWx_l \tag{78}$$

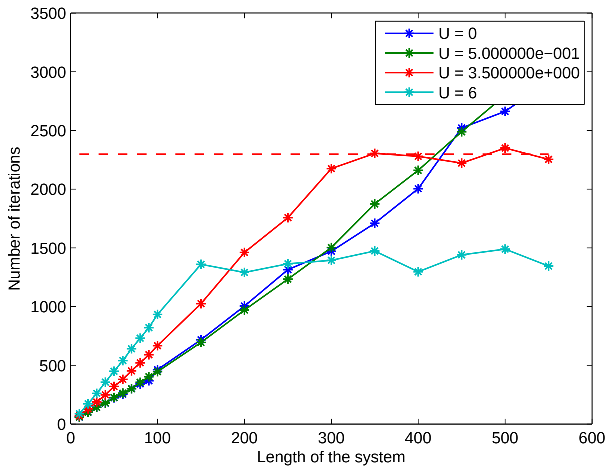


Figure 25: Convergence plot of IDR(8) with a tolerance of 10^{-8} for a disordered system with $W = 10$, $t = 1$ and $\varepsilon = 0.5$. The number of iterations is averaged over 50 realisations of the disorder. The red dotted line shows the expected number of iterations for $U = 3.5$

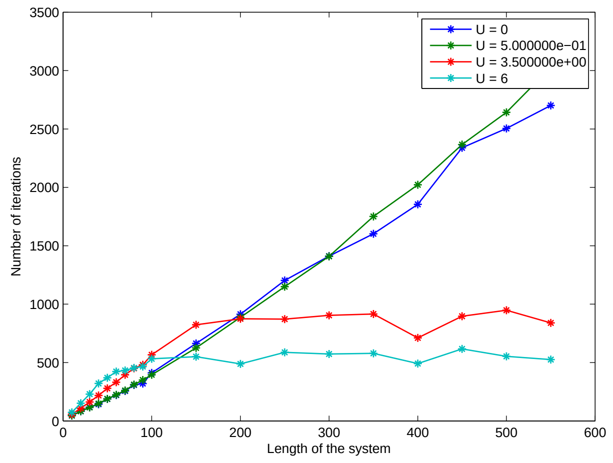
since $W \ll x_l$ (see equation 66), with c a constant that depends on the iterative method. For GMRES we have that $c \approx 1$ since GMRES convergences in at most as many steps as the system size. Looking at Figure 19a and Figure 19b we can see that IDR(8) requires approximately 2 times the number of iterations, and therefore $c \approx 2$ for IDR(8). If we now rewrite equation 78 and substitute it in equation 77, we get:

$$e^{\frac{-N}{cW\xi}} \sim \delta \tag{79}$$

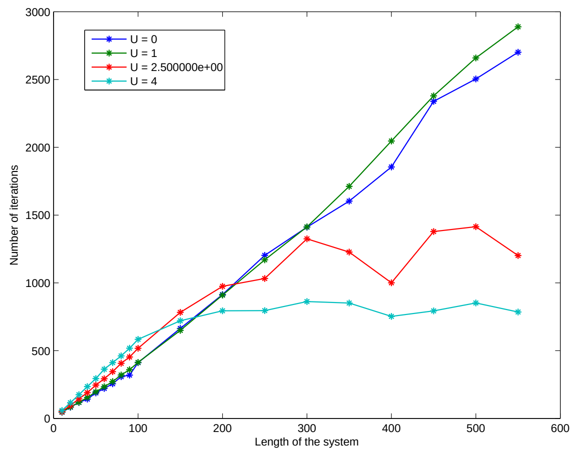
which can be rewritten into an expression for the number of iterations N:

$$N \sim -cW\xi \ln \delta \tag{80}$$

We therefore expect the number of iterations N to become independent of the length if the system becomes localized.
The dependence of the number of iterations on the length of the system for different disorders is plotted in Figure 26 and figure 25. In these figures we can see that the number of iterations grows as the length of the system becomes larger, until a certain length is reached after which the number of iterations becomes independent of the length. We can also see that this point is reached sooner for larger values of U, which corresponds to the fact that the localization length becomes smaller for larger values of U. Furthermore we can see that a system with disorder in the ballistic and diffusive regime requires more iterations to solve than a system with zero disorder in the same regimes, and that the number of iterations required in these regimes is bigger for larger U. Finally, we can see that equation 80 holds. If we for example look at $U = 3.5$, we have a mean free path of $l_m \approx 3$ and a localization length of $\xi \approx 7$, so using equation 80 with $\delta = 10^{-8}$ and $c = 2$ we obtain the red dotted line shown in Figure 25. In Table 1 we can see that equation 80 is also a good approximation for the other values of disorder and tolerance.



(a)



(b)

Figure 26: Convergence plot of IDR(8) with a tolerance of 10^{-4} for a disordered system with $W = 10$, $t = 1$ and $\varepsilon = 0.5$. The number of iterations is averaged over 50 realisations of the disorder.

U	N_t	N_e
1	$2.8 \cdot 10^4$	-
3.5	$2.6 \cdot 10^3$	$2.4 \cdot 10^3$
6	$8.1 \cdot 10^2$	$1.4 \cdot 10^3$

(a) $\delta = 10^{-8}$

U	N_t	N_e	U	N_t	N_e
0.5	$5.6 \cdot 10^4$	-	3.5	$1.3 \cdot 10^3$	$9 \cdot 10^2$
1	$1.4 \cdot 10^4$	-	4	$8.3 \cdot 10^2$	$8 \cdot 10^2$
2.5	$2.2 \cdot 10^3$	$1.3 \cdot 10^3$	6	$4.1 \cdot 10^2$	$5 \cdot 10^2$

(b) $\delta = 10^{-4}$

Table 1: The theoretical number of iterations N_t and the (approximate) number of iterations N_e found with numerical experiments for different values of disorder for a system with $W = 10$, $\varepsilon = 0.5$, $t = 1$ and $c = 2$. Table (a) shows the number of iterations for a tolerance $\delta = 10^{-8}$ and table (b) shows the number of iterations for a tolerance $\delta = 10^{-4}$

7 Conclusions and Discussion

In this Bachelor thesis, we investigated the use of iterative methods to solve a square quantum billiard problem. Four different methods were used: GMRES, GMRES(l), BiCGStab and IDR(s). Even though it was possible to solve the system with all four methods, we have seen that GMRES has a bad convergence history despite requiring the least amount of iterations and that GMRES(l) and BiCGStab require a large (i.e. a couple times the system size) number of iterations to reach convergence. Furthermore we have seen that IDR(1) behaves similar to BiCGStab and that IDR(s) approaches GMRES for growing s, both in convergence behaviour and in the required number of iterations. When taking into account the memory requirements of all the methods, we concluded that IDR(s) is preferred over GMRES for large systems, while GMRES is the best suited options of these four methods for small systems.

As for a more detailed description of the convergence behaviour of all the methods in general, we have seen that the number of iterations N shows a linear dependency on both W and L for a system with $W \ll L$ and we have seen that N shows a non-linear dependence on W, namely $N \sim \sqrt{W}$, for a system with $W \gg L$. These dependencies seem to correspond to the spatial decay factor of the corresponding Green's function for the same systems. Whether there is an actual relation between the shape of the solution and the number of iterations N would require further research.
In addition to the dependence of N on the system size, we have seen that the system matrix becomes diagonally dominant if we choose E outside of the bandwidth of the Hamiltonian (i.e, outside of $0 \leq E \leq 8t$). This diagonal dominance results in fast convergence of all the methods.

In order to improve the convergence behaviour of GMRES, we tried to apply a preconditioner similar to a shifted Laplacian. We have seen that this preconditioner shifts the spectrum to either a half-plane or to a circle depending on the exact choice of the preconditioner. For imaginary E, we have seen that the preconditioner can be chosen such that the origin lies outside the circle, resulting in faster convergence. But since the convergence behaviour itself is not improved, the choice of a preconditioner similar to a shifted Laplacian, might not be the optimal choice. Looking at a system with real E, we have shown that the origin always lies on the circle, which means the convergence behaviour of GMRES can not be improved by this choise of preconditioner, even though the number of iterations required is somewhat reduced. This reduction in number of iterations is due to the fact that the spectrum of the preconditioned system is less spread out than the spectrum of the unpreconditioned system. From this we can conclude that the preconditioner considered in this thesis does not improve the convergence behaviour enough. Whether there is a preconditioner that is better suited for the system considered in this thesis would require further research.

Finally, we considered the convergence behaviour of IDR(8) for a disordered system. We have shown that a disordered system requires more iterations to solve than a clean system in the diffusive and in the ballistic regime. Furthermore we have shown that the required number of iterations is larger if the disorder is larger. We have also seen that the number of iterations becomes independent on L if the system is in the localized regime, and that the point where the system enters the localized regime is reached sooner for larger disorders.

Appendix A Python Code

For the figures and for the numerical experiments, the following code was used. The linear system was made in python using the package Kwant, after which the different matrices of this system were save so that they could later be uploaded to Matlab.

```
from __future__ import division
import kwant
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
import scipy.sparse.linalg as sla
import random
import copy
import time
from scipy import io
from math import sqrt


### Make the actual scattering region and the lead ###
def MakeSystem(W,L):
    a = 1
    t = 1

    global sys
    lat = kwant.lattice.square(a)
    sys = kwant.Builder()

    sys[lat.shape((lambda pos: 0 <= pos[0] < L and 0 <= pos[1] < W),
    (0, 0))] = 4 * t
    sys[lat.neighbors()] = -t

    lead = kwant.Builder(kwant.TranslationalSymmetry((-a, 0)))
    lead[lat.shape((lambda pos: 0 <= pos[1] < W), (0, W//2))] = 3 * t
    lead[lat.neighbors()] = -t

    sys.attach_lead(lead)
    sys.attach_lead(lead.reversed())

    kwant.plot(sys)
    sys = sys.finalized()

### Get the linear system from the quantum billiard ###
def LinearSystem(e):
    leadsin = [0,1]
    global lead_info
    linsys, lead_info = kwant.solvers.default.hidden_instance.
    _make_linear_sys(sys, leadsin, energy = e, check_hermiticity
    = False, realspace = True)

    global lhs
    global rhs
    global n
    global m

    #lhs is a numpy.sparse.csc_matrix, containing the left hand
    #side of the system of equations.
    lhs = linsys.lhs


    #rhs is a list of matrices with the right hand side, with
    #each matrix corresponding to one lead mentioned in
    #'in_leads'.
    rhs0 = linsys.rhs[0]
    rhs1 = linsys.rhs[1]

    #You must have the whole rhs to use GMRES (or any other
    #solver) so first get full rhs
    rhs = sp.sparse.hstack((rhs0,rhs1))
    rhs = rhs.todense()


    n = lhs.shape[0]
    m = rhs.shape[1]

### Get all seperate matrices of the linear system ###
def GetMatrices():
        #Create H_s
        H_s = sys.hamiltonian_submatrix(sparse = True)

        #Create H_E
        #lead_info is a list of objects which contains one entry for
        #each lead.
    HE0 = lead_info[0]
    HE1 = lead_info[1]

    O_block = np.zeros((HE0.shape[0],HE0.shape[0]), dtype =
    complex)
    HE0_ex = np.hstack((HE0, O_block))
        HE1_ex = np.hstack((O_block, HE1))
    HE = np.vstack((HE0_ex, HE1_ex))
```

```
        #Create dense and sparse E
    E = e * np.eye(n, dtype = complex)

    diag = e * np.ones(n)
    E_sparse = sp.sparse.dia_matrix(([diag],[0]), shape = (n, n),
     dtype = complex).tocsr()

        #Create dense and sparsePHP
    PHP = rhs.dot(HE.dot(rhs.getH()))

    PHP_sparse = lhs - H_s + E_sparse

    return H_s, HE, E, E_sparse, PHP, PHP_sparse

### Show spectrum of matrices ###
def Spectrum(A): #it takes a dense matrix
        eigvl = np.linalg.eigvals(A)

    plt.plot(eigvl.real, eigvl.imag, 'o')
    plt.xlabel('Real_part')
    plt.ylabel('Imaginary_part')
    plt.show()

### Make lists of different matrices to upload to Matlab.
(NB. only that part of the code was run that needed to be
uploaded to Matlab). ###

# Lists for different lengths
RightHandSide = []
LeftHandSide = []
Length = []

W = 60
e = 1

for L in np.arange(1,45,2): #(1,45,2) for W>> L, (80,220,5) for W<< L
    MakeSystem(W,L)
    LinearSystem(e)
    b = rhs[:,(m-1)]
    RightHandSide.extend([b])
    LeftHandSide.extend([lhs])
        Length.extend([L])

sp.io.savemat('Path_to_matlab_folder/RightHandSide.mat',
mdict={'RightHandSide': RightHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/LeftHandSide.mat',
mdict={'LeftHandSide': LeftHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/Length.mat',
mdict={'Length': Length}, oned_as = 'column')

# Lists for different widths
RightHandSide = []
LeftHandSide = []
Width= []

L = 60
e = 1

for W in np.arange(10,250,10):
    MakeSystem(W,L)
    LinearSystem(e)
    b = rhs[:,(m-1)]
    RightHandSide.extend([b])
    LeftHandSide.extend([lhs])
    Width.extend([W])

sp.io.savemat('Path_to_matlab_folder/RightHandSide.mat',
mdict={'RightHandSide': RightHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/LeftHandSide.mat',
mdict={'LeftHandSide': LeftHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/Width.mat',
mdict={'Width': Width}, oned_as = 'column')

# List for different energies
# Real E
RightHandSide = []
LeftHandSide = []
Energy = []

L = 30
W = 10

for e in np.arange(-2., 10., 0.1): #-10, 10, 0.1
    E = e
    MakeSystem(W,L)
    LinearSystem(E)
    b = rhs[:,(m-1)]
    RightHandSide.extend([b])
    LeftHandSide.extend([lhs])
```

```
        Energy.extend([E])

sp.io.savemat('Path_to_matlab_folder/RightHandSide.mat',
mdict={'RightHandSide': RightHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/LeftHandSide.mat',
mdict={'LeftHandSide': LeftHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/Energy.mat',
mdict={'Energy': Energy}, oned_as = 'column')

#Imaginary E
RightHandSide = []
LeftHandSide = []
Energy = []

L = 30
W = 10

for e in np.arange(-4., 4., 0.1):
    E = complex(1,e)
    MakeSystem(W,L)
    LinearSystem(E)
    b = rhs[:,(m-1)]
    RightHandSide.extend([b])
    LeftHandSide.extend([lhs])
    Energy.extend([E.imag]) #since the real part is fixed,
    #plot it against the imaginary part.

sp.io.savemat('Path_to_matlab_folder/RightHandSide.mat',
mdict={'RightHandSide': RightHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/LeftHandSide.mat',
mdict={'LeftHandSide': LeftHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/Energy.mat',
mdict={'Energy': Energy}, oned_as = 'column')

# Lists for a disordered system. The disorder matrix D is added
# in matlab
RightHandSide = []
HP = []
Length = []

W = 10
e = 0.5 #make sure this is a double, not an integer

a = np.arange(10,100,10)
c = np.arange(100,560,50) #(150,500,30)
```

```
Range = np.concatenate((a,c))

for L in Range:
        MakeSystem(W,L)
        LinearSystem(e)
        H0, HE, E, E_sparse, PHP, PHP_sparse = GetMatrices()
        b = rhs[:,(m-1)]
        RightHandSide.extend([b])
        HP.extend([H0-PHP_sparse])
        Length.extend([L])

sp.io.savemat('Path_to_matlab_folder/RightHandSide.mat',
mdict={'RightHandSide': RightHandSide}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/HP.mat', mdict={'HP': HP},
oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/Length.mat',
mdict={'Length': Length}, oned_as = 'column')
sp.io.savemat('Path_to_matlab_folder/e.mat', mdict={'e': e},
oned_as = 'column')

### Preconditioning ###
#GMRES
def callback_g(rk):
    rho_n = np.linalg.norm(rk) / np.linalg.norm(B[:,0])
    global k_g
    global column_rho_g
    global column_iter_g
    column_rho_g.extend([rho_n])
    column_iter_g.extend([k_g])
    k_g += 1

def GMRES(A, b, tolerance, m=None):
    if m is None:
        m = b.shape[1]
    start_g = time.clock()

    n = A.shape[0]

    global B
    B = b

    rho_g = []
    iter_g = []

    global x_gmres
    x_gmres = np.zeros((n,m), dtype = complex)

    global k_g
    global column_rho_g
    global column_iter_g

    for i in xrange(m):
        k_g = 0
        column_rho_g = []
        column_iter_g = []
        y, info_g = sla.gmres(A, b[:, i], tol = tolerance,
        restart = n, callback = callback_g)
        x_gmres[:,i] = y
        rho_g.extend([column_rho_g])
        iter_g.extend([column_iter_g])

    elapsed_g = (time.clock() - start_g) / m
    print "Elapsed time of GMRES per column is %r seconds" % elapsed_g

    fig = plt.figure()
    for i in xrange(m):
        plt.semilogy(iter_g[i][:], rho_g[i][:], 'o')
        plt.xlabel('number of iteration')
    plt.ylabel('relative residual')
    plt.show()

#Create the circle
def circle(phi, a1, a2, b1, b2):
        r = sqrt(((b2-b1)**2+(a2-a1)**2)/(2*b2)**2)
        cx = (b2+b1)/(2*b2)
        cy = (a2-a1)/(2*b2)
        return r*np.cos(phi)+cx, r*np.sin(phi)+cy

#Create the line (which is the boundary of the half-plane)
def l(t, a1, a2, b1, b2):
    return (b1*(t-1))/(a1 - a2)
def Preconditioning(e, z, W, L):
    MakeSystem(W, L)
    LinearSystem(e)
    H_s, HE, E, E_sparse, PHP, PHP_sparse = GetMatrices()
    Spectrum(lhs.todense())

    I = np.eye(n, dtype = complex)

    #You have to change this one by hand
    e2 = '-|%r|i'  %e

    P1 = H_s + PHP - z * I

    a1 = e.real
    b1 = e.imag
    a2 = z.real
    b2 = z.imag

    P = P1.getI()
    lhs_pre = copy.deepcopy(lhs) #is sparse
    b_pre = copy.deepcopy(rhs)
    pre_sys = P.dot(lhs_pre.todense())

    ew = np.linalg.eigvals(pre_sys)
    eigenw = np.linalg.eigvals(lhs_pre.todense())

    title = '$E = %r$, $E_2 = %s$' % (e, e2)
    if b2 == 0:
        t = np.arange(-60,75,0.1)
        fig = plt.figure()
        plt.plot(ew.real, ew.imag, 'o')
        plt.plot(t, l(t, a1, a2, b1, b2), 'r')
        fig.suptitle(title)
        plt.xlabel('real part')
        plt.ylabel('imaginary part')
        plt.show()

    else:
        fig = plt.figure()
        ax = fig.add_subplot(111,aspect='equal')
        phis=np.arange(0,6.28,0.01)
        ax.plot( *circle(phis, a1, a2, b1, b2), c='r',ls='-' )
        plt.plot(ew.real, ew.imag, 'o')
        plt.plot(0,0,'or')
        fig.suptitle(title)
        plt.xlabel('real part')
        plt.ylabel('imaginary part')
        plt.show()

    return P, b_pre, lhs_pre

W = 20
L = 50
e = complex(2,2)
```

```
z = complex(0, -abs(e))
P, b_pre, lhs_pre = Preconditioning(e,z,W,L)
```

Appendix B Matlab Code

Convergence History for different methods:

```
clc
clear all
close all

load lhs.mat
load rhs.mat

b = size(rhs);
m = b(2);
n = length(rhs);
max_it = 10*n;
tol = 1e-8;

% GMRES
for i = 1:m
    [x_g, info_g, r_g, i_g, res_g] = gmres(lhs, rhs(:,i), n, tol, max_it);
    iter_g = [0:1:(length(res_g)-1)];
    it_g{i} = iter_g;
    r = res_g./norm(rhs(:,i));
    rho_g{i} = r;
end

figure
for i = 1:m
    semilogy(it_g{i}, rho_g{i})
    hold all
end
xlabel('number of iterations')
ylabel('relative residual')

% Restarted GMRES
restart = 10;
for i = 1:m
    [x_gr, info_gr, r_gr, i_gr, res_gr] = gmres(lhs, rhs(:,i),
    restart, tol, max_it);
    iter_gr = [0:1:(length(res_gr)-1)];
    it_gr{i} = iter_gr;
    r = res_gr./norm(rhs(:,i));
    rho_gr{i} = r;
end

figure
```

```
for i = 1:m
    semilogy(it_gr{i}, rho_gr{i})
    hold all
end
xlabel('number␣of␣iterations')
ylabel('relative␣residual')

% BiCGStab
for i = 1:m
    [x_b, info_b, r_b, i_b, res_b] = bicgstab(lhs, rhs(:,i), tol, max_it);
    iter_b = [0:1:(length(res_b)-1)];
    it_b{i} = iter_b;
    r = res_b./norm(rhs(:,8));
    rho_b{i} = r;
end

figure
for i = 1:m
    semilogy(it_b{i}, rho_b{i})
    hold all
end
xlabel('number␣of␣iterations')
ylabel('relative␣residual')

% IDR with 8
for i = 1:m
    [x_i_a, info_i_a, r_i_a, i_i_a, res_i_a] = idrs(lhs, rhs(:,i), 8,
    tol,        max_it);
        iter_i = [0:1:(length(res_i_a)-1)].';
        it_i{i} = iter_i;
        r = (res_i_a./norm(rhs(:,8))).';
        rho_i{i} = r;
end

figure
for i = 1:m
        semilogy(it_i{i}, rho_i{i})
        hold all
end
xlabel('number␣of␣iterations')
ylabel('relative␣residual')
```

Code to make plots of the dependence of the number of iterations on the length, and the convergence history for the different methods (to get the plots for the dependence on the width, change 'Length' with 'Width'):

```
clc
clear all
close all

load rhs.mat
load lhs.mat

load RightHandSide.mat
load LeftHandSide.mat
load Length.mat

tol = 1e-8; % default is 1e-8
restart = 8;

%non-restarted gmres
iterations_g = [];
time_g = [];
for i = 1:length(Length)
    tic;
    n = length(RightHandSide{i});
    max_it = 10*n;
    [x_g, info_g, r_g, i_g, res_g] = gmres(LeftHandSide{i},
    RightHandSide{i}, n,         tol, max_it);
    iter_g = [0:1:(length(res_g)-1)];
    iterations_g = [iterations_g; i_g(2)];
    rho_g = res_g./norm(RightHandSide{i});
    time_g = [time_g; toc];
end

%BiCGStab
iterations_b = [];
time_b = [];
for i = 1:length(Length)
        tic;
        n = length(RightHandSide{i});
        max_it = 10*n;
        [x_b, info_b, r_b, i_b, res_b] = bicgstab(LeftHandSide{i},
        RightHandSide{i},        tol, max_it);
        iter_b = [0:1:(length(res_b)-1)];
        iterations_b = [iterations_b; i_b];
        rho_b = res_b./norm(RightHandSide{i});
        time_b = [time_b; toc];
```

```
end

% IDR with 1
iterations_i_a = [];
time_i_a = [];
for i = 1:length(Length)
        tic;
        n = length(RightHandSide{i});
        max_it = 10*n;
        [x_i_a, info_i_a, r_i_a, i_i_a, res_i_a] = idrs(LeftHandSide{i},
        RightHandSide{i}, 1, tol, max_it);
        iter_i_a = [0:1:(length(res_i_a)-1)].';
        iterations_i_a = [iterations_i_a; i_i_a];
        rho_i_a = (res_i_a./norm(RightHandSide{i})).';
time_i_a = [time_i_a; toc];
end


% IDR with s = 4 (= default)
iterations_i_b = [];
time_i_b = [];
for i = 1:length(Length)
    tic;
    n = length(RightHandSide{i});
    max_it = 10*n;
    [x_i_b, info_i_b, r_i_b, i_i_b, res_i_b] = idrs(LeftHandSide{i},
    RightHandSide{i}, 4, tol, max_it);
    iter_i_b = [0:1:(length(res_i_b)-1)];
    iterations_i_b = [iterations_i_b; i_i_b];
    rho_i_b = res_i_b./norm(RightHandSide{i});
    time_i_b = [time_i_b; toc];
end


% IDR with s = 8
iterations_i_c = [];
time_i_c = [];
for i = 1:length(Length)
        tic;
        n = length(RightHandSide{i});
        max_it = 10*n;
        [x_i_c, info_i_c, r_i_c, i_i_c, res_i_c] = idrs(LeftHandSide{i},
        RightHandSide{i}, 8, tol, max_it);
    iter_i_c = [0:1:(length(res_i_c)-1)];
    iterations_i_c = [iterations_i_c; i_i_c];
    rho_i_c = res_i_c./norm(RightHandSide{i});
    time_i_c = [time_i_c; toc];
end


%%% Length %%%

%%% Plots %%%
% dependence of N on Length
figure
plot(Length, iterations_g, '-k*', Length, iterations_b, '-b*',
Length,          iterations_i_a, '-r*', Length, iterations_i_b,
'-g*', Length, iterations_i_c, '-m*', 'MarkerSize', 6,
'LineWidth', 1)
legend('gmres', 'bicgstab', 'idr(s=1)', 'idr(s=4)', 'idr(s=8)')
xlabel('Length_of_the_system')
ylabel('Number_of_iterations')

% convergence history for different methods
figure
semilogy(iter_g.', rho_g.', 'k', iter_b.', rho_b.', 'b',
iter_i_a, rho_i_a, 'r',          iter_i_b, rho_i_b, 'g', iter_i_c,
rho_i_c, 'm' )
legend('gmres', 'bicgstab', 'idr(s=1)', 'idr(s=4)', 'idr(s=8)')
xlabel('Number_of_iterations')
ylabel('Relative_Residual')
```

Code to make plots of the dependence of the number of iterations on the energy of the system:

```
clc
clear all
close all

load rhs.mat
load lhs.mat
load RightHandSide.mat
load LeftHandSide.mat
load Energy.mat

tol = 1e-8; % default is 1e-8
restart = 8;


%non-restarted gmres
iterations_g = [];
time_g = [];
for i = 1:length(Energy)
    tic;
    n = length(RightHandSide(i,:));
    max_it = 10*n;
    [x_g, info_g, r_g, i_g, res_g] = gmres(LeftHandSide{i},
    RightHandSide(i,:).', n, tol, max_it);
    iter_g = [0:1:(length(res_g)-1)];
    iterations_g = [iterations_g; i_g(2)];
    rho_g = res_g./norm(rhs(:,8));
    time_g = [time_g; toc];
end

%BiCGStab
iterations_b = [];
time_b = [];
for i = 1:length(Energy)
    tic;
    n = length(RightHandSide(i,:));
    max_it = 10*n;
    [x_b, info_b, r_b, i_b, res_b] = bicgstab(LeftHandSide{i},
    RightHandSide(i,:).', tol, max_it);
    iter_b = [0:1:(length(res_b)-1)];
    iterations_b = [iterations_b; i_b];
    rho_b = res_b./norm(rhs(:,8));
    time_b = [time_b; toc];
end
```

```
% IDR with 1
iterations_i_a = [];
time_i_a = [];
for i = 1:length(Energy)
    tic;
    n = length(RightHandSide(i,:));
    max_it = 10*n;
    [x_i_a, info_i_a, r_i_a, i_i_a, res_i_a] = idrs(LeftHandSide{i},
    RightHandSide(i,:).', 1, tol, max_it);
    iter_i_a = [0:1:(length(res_i_a)-1)].';
    iterations_i_a = [iterations_i_a; i_i_a];
    rho_i_a = (res_i_a./norm(rhs(:,8))).';
    time_i_a = [time_i_a; toc];
end


% IDR with s = 4 (= default)
iterations_i_b = [];
time_i_b = [];
for i = 1:length(Energy)
    tic;
    n = length(RightHandSide(i,:));
    max_it = 10*n;
    [x_i_b, info_i_b, r_i_b, i_i_b, res_i_b] = idrs(LeftHandSide{i},
    RightHandSide(i,:).', 4, tol, max_it);
    iter_i_b = [0:1:(length(res_i_b)-1)];
    iterations_i_b = [iterations_i_b; i_i_b];
    rho_i_b = res_i_b./norm(rhs(:,8));
    time_i_b = [time_i_b; toc];
end


% IDR with s = 8
iterations_i_c = [];
time_i_c = [];
for i = 1:length(Energy)
    tic;
    n = length(RightHandSide(i,:));
    max_it = 10*n;
    [x_i_c, info_i_c, r_i_c, i_i_c, res_i_c] = idrs(LeftHandSide{i},
    RightHandSide(i,:).', 8, tol, max_it);
    iter_i_c = [0:1:(length(res_i_c)-1)];
    iterations_i_c = [iterations_i_c; i_i_c];
    rho_i_c = res_i_c./norm(rhs(:,8));
    time_i_c = [time_i_c; toc];
end


%%% Energy %%%

%%% Plots %%%
figure
plot(Energy, iterations_g, '-k*', Energy, iterations_b, '-b*',
Energy,         iterations_i_a, '-r*', Energy, iterations_i_b,
'-g*', Energy, iterations_i_c, '-m*',    'MarkerSize', 6,
'LineWidth', 1)
legend('gmres', 'bicgstab', 'idr(s=1)', 'idr(s=4)', 'idr(s=8)')
xlabel('Imaginary part of the energy (E) of the system')
ylabel('Number of iterations')
```

Code to make plots of the dependence of the number of iterations on the length of the system for a disordered system:

```
clc
clear all
close all

tol = 1e-8; % default is 1e-8

%%% Changing length for fixed disorders. Adding the disorder in matlab %%%
load Length.mat
load RightHandSide.mat
load HP.mat
load e.mat
U = [0,0.5,3.5,6];
s = size(U);

%%% IDR with s = 8 %%%
for i = 1:s(2) %number of different disorders considered
    av_it_i = [];
    av_it_g =[];
    for l = 1:length(Length) %number of disorders
        iterations_i_c = [];
        iterations_g = [];
        for k = 1:50 %number of realisations over which is averaged
            [i,Length(l),k]
            n = length(RightHandSide{l});
            max_it = 10*n;

            disorder = diag(unifrnd(e-U(i)/2,e+U(i)/2,[n,1]));
            A = HP{l} - disorder;
            [x_i_c, info_i_c, r_i_c, i_i_c, res_i_c] = idrs(A,
            RightHandSide{l}, 8, tol, max_it);
            iterations_i_c = [iterations_i_c; i_i_c];
        end
        av_i = mean(iterations_i_c);
        av_it_i = [av_it_i, av_i];
        end
        it_i{i} = av_it_i;;
    LegendinfoU{i} = sprintf('U = %d',U(i));
end


figure
for j = 1:s(2)
        plot(Length, it_i{j}, '-*', 'MarkerSize', 6,  'LineWidth', 1)
        hold all
end
legend(LegendinfoU)
title('Dependence␣of␣convergence␣of␣IDR(8)␣on␣lenght␣of␣a
disordered␣system␣(tol␣=␣1e−8,␣W␣=␣20,␣t␣=␣1,␣E␣=␣0.39)')
xlabel('Length␣of␣the␣system')
ylabel('Number␣of␣iterations')
```

References

[1] Datta, *Electronic Transport in Mesoscopic Systems*, Cambridge University Press 1995, p.132 - p.149, p.196 - p.200

[2] M. B. Van Gijzen, Y. A. Erlangga, C. Vuik, *Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian*, SIAM 2007

[3] Iserles, *A First Couse in the Numerical Analysis of Differential Equations*, Cambridge University Press 1996, p112 - p119

[4] Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company 1996, p.158 - p.159, p.210-p.211, p.215-p.220

[5] P. Sonneveld, M. B. Van Gijzen, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, SIAM 2008