Large scale simulation of wave-packet propagation via Krylov subspace methods and application to graphene

Simulation von Wellenpaketen mit Krylov-Methoden und Anwendung auf Graphen

Victor Häfner
Institut für Theorie der Kondensierten Materie
Institut für Nanotechnologie,
Karlsruhe Institute of Technology,
76131 Karlsruhe, Germany
E-mail: victor.haefner@gmail.com

Referent: Prof. Dr. (apl.) F. Evers
Korreferent: Prof. Dr. K. Busch
Prüfexemplar

November 11, 2011

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, November 11, 2011

Acknowledgments

An dieser Stelle will ich meiner Verlobten Polina Stoyanova danken die mich das Jahr über immer unterstützt hat.

Ich danke Ferdinand Evers für die Geduld und hervorragende Betreuung, dem wissenschaftlichem Umfeld in dem ich Arbeiten durfte und nicht zuletzt für all das vermittelte Wissen.

Ich will auch Christian Seiler für die langen Diskusionen und vielen Ratschlägen danken, sowie für das Korrekturlesen, und auch allen anderen der Arbeitsgruppe um Ferdinand Evers.

Contents

List of Figures

Zusammenfassung

Die Zielsetzung dieser Diplomarbeit ist es, eine numerische Methode zu implementieren, um quantenmechanische Wellenpakete zu propagieren. Dies wollen wir für möglichst lange Zeiten und große Systeme ohne Beschränkungen der zugrunde liegenden Geometrie durchführen. Dadurch würde ermöglicht, die Dynamik von Quantenteilchen und somit Effekte von Bandstruktur, Rändern oder Inhomogenitäten zu erforschen. Die eingesetzte Methode basiert auf der so genannten *Arnoldi* -Methode, eine von mehreren Methoden, die zur Familie der Krylov-Methoden gehört. Der Krylov-Raum ist ein um einen Quantenzustand aufgebauter Unterraum, in den wir den Hamiltonian hineinprojizieren. Dadurch können wir ohne Probleme diagonalisieren. Dies ermöglicht es uns, den Zeitentwicklungsoperator zu berechnen, ohne die volle Matrix des Hamiltonoperators diagonalisieren zu müssen. Dadurch müssen dass wir nicht mit der vollen Matrix arbeiten und brauchen daher wesentlich weniger Speicher. Wir erreichen so die nötigen Systemgrößen und die nötige Energieauflösung, um quantenmechanische Lokalisierungseffekte zu berechnen. Wir haben unseren auf Tight-binding-Gitter ausgelegten Code parallelisiert; hierfür nutzen wir die MPI-Bibliothek. Wir konnten so für unsere Simulation die Supercomputer der Region bzw. Deutschlands nutzen, wie den HC3 am SCC in Karlsruhe oder den JUROPA am JSC in Jülich. Wir erreichen bis dato unerreichte Systemgrößen von 16348×16348 Gitterpunkten, mit einer Million Zeitschritten in weniger als 12 Stunden auf JUROPA. Mit diesem Erfolg haben wir den Weg bereitet für die Simulation der Quantendynamik effektiver Einteilchentheorien von ungeordneten Metallen.

Wir nutzen die von uns entwickelte numerische Methode um Graphen zu untersuchen. Die Kristallstruktur von Graphen ist zweidimensional und bildet ein hexagonales Gitter mit einer zweiatomigen Basis. Wir haben uns

die Zustandsdichte angeschaut, insbesondere mit zufälliger Unordnung in Form von unbesetzten Gitterplätzen. Solche Fehlstellen sind in der Realität zwar selten, jedoch von großem Interesse, da in Experimenten Adatome lokal die sp^2-Hybridisierung brechen können, indem sie das Kohlenstoff Atome in die sp^3-Hybridisierung zwingen. Effektiv wird somit ein p_z-Orbital aus dem π-Band entfernt, was man gut durch einen unbesetzten Gitterplatz approximieren kann.

Diese Art von Unordnung bricht gewisse Symmetrien des Dirac-Hamilton-Operators vom reinen Graphen, jedoch nicht alle. Das Material entspricht der Symmetrieklasse BDI, bei der Chiralität und Zeitumkehrinvarianz erhalten sind. Ausgehend von früheren analytischen Ergebnissen erwarten wir ein singulares Verhalten in der Nähe des Energienullpunktes dort, wo die chirale Symmetrie zum tragen kommt. Ein Pseudogap öffnet sich in dessem Mitte, Nullmoden erscheinen, die miteinander koppeln, das Pseudogap füllt sich auf. Dieses Verhalten wurde schon in früheren numerischen Arbeiten beobachtet, jedoch wurden die Details der Strukturen, die man für verschiedene Unordnungskonfigurationen beobachtet, nicht eingehend diskutiert. Wir ermöglichen dies mit unserer Simulation durch eine verbesserte Auflösung der Energie.

Anfangs haben wir Fehlstellen gleichmäßig in beide Graphen-Untergitter eingebracht. Der Dirac-Kegel ist aufgefüllt und die Zustandssumme weist eine integrable Divergenz auf. Die Form dieser Divergenz kann durchaus grob durch das analytische Verhalten $|E|^{-1}$ beschrieben werden. Jedoch scheinen unsere Daten nicht ohne weiteres mit der Energie-Abhängigkeit des Vorfaktors in Einklang zu bringen sein. Ein Grund dafür könnte sein das die analytische Beschreibung für ein Regime niedrigerer Energien gilt, in dem wir uns noch nicht befinden. Sollte dies der Fall sein, so haben wir eventuell

ein neues Übergangsregime in der Zustandsdichte entdeckt, das in den realen Niederenergiebereich übergeht, im thermodynamischen Limes.

Die Entwicklung der Zustandsdichte haben wir auch für in beiden Untergittern ungleich verteilten Fehlstellen studiert. Wir bestätigen bestehende qualitative Ergebnisse für Unordnung in nur einem Untergitter. Zusätzlich können wir auch niederenergetische Asymptoten mit $|E|^2$-Verhalten identifizieren, und ein Zwischenregime mit nicht universellem effektivem Exponenten $|E|^{1/z}$. Wie der asymptotische Bereich im Niederenergiebereich und im Limes des voll unkompensierten Falls verschwindet bleibt offen. Wir schlagen ein Phasendiagramm vor, dass das Verhalten vom Exponenten $1/z$ beschreibt, insbesondere sein Vorzeichen.

Summary

The aim of this thesis is to implement a computational method that allows to propagate a quantum mechanical wave packet for long times in large systems with arbitrary geometries. In this way one can study the quantum dynamics of particles and thus the effect of band structure, edges or inhomogeneities. The technique is based on the *Arnoldi method* which belongs to the *Krylov-subspace methods*. This method consists of spanning a sub space around the current quantum state and projecting the Hamiltonian of the system in it. In this way the time evolution operator of the system is computed without diagonalizing the full Hamiltonian, which is absolutely essential for the system sizes and energy resolution required to study quantum localization effects. Our code is efficiently parallelized for tight-binding lattices with the MPI library and runs on the biggest clusters of the SCC (Karlsruhe) and the JSC (Jülich). We reach an unprecedented system size of 16348×16348 lattice sites with a million time steps observation time in only 12 hours computer time on the JUROPA cluster in Jülich. With this success, we have paved the ground for future studies of quantum dynamics in very broad classes of disordered tight binding models.

We have applied our method to the material graphene. The crystal structure of graphene is a 2D honeycomb lattice with a two atom unit cell. We studied the density of states (DoS) of graphene in the presence of randomly placed vacancies. This case is interesting to study because even though in experiments carbon vacancies are rare, adatoms can break locally the sp^2 hybridization by forming sp^3-bonds and thus effectively eliminate a p_z-orbital from the conjugated sheet. Thus, an approximate realization of vacancies is found.

Vacancy disorder breaks some of the symmetries of the Dirac Hamiltonian

of clean graphene, but not all of them. With vacancies present the material belongs to the symmetry group BDI, which respects a "chirality" symmetry and time reversal invariance. Based on earlier analytical investigations one expects that the DoS exhibits a singular shape near zero energy, where the chiral symmetry becomes active. Zero modes are filling into the pseudo gap of clean graphene which couple to each other thus filling the gap. Indeed, the overall singular shape has been observed already in earlier numerics. However, the detailed sub gap-structure was not studied before. The improved resolution of our code has made this possible.

We first have introduced the same amount of vacancies in both sub lattices. The Dirac cone is filled and the DoS exhibits an integrable divergency. While the strongest feature of this divergency is not inconsistent with an analytical prediction, $|E|^{-1}$, we find qualitative disagreement with the sub leading energy dependency of the pre factor. The computational result does not exhibit the analytically predicted scaling with the system size. In principle, a possible reason could be that the analytical result becomes valid at energy scales still too low to resolve with our approach. If this is the case, then we have identified a new cross-over regime in the DoS that evolves into the true low energy asymptotes in the thermodynamic limit.

The evolution of the DoS has also been studied with increasing mismatch between the impurity concentrations in both sub lattices. We confirm earlier qualitative findings about the evolution of the full gap at maximum mismatch, where vacancies are in one sub lattice only. In addition, we can also identify a low energy asymptotic with a $|E|^2$ behavior of the DoS and an intermediate regime with a non-universal effective exponent $|E|^{1/z}$. The fate of the low energy asymptotic in the limit of zero sub lattice mismatch remains a question open for future research. We have formulated a scenario in terms

of a phase diagram displaying the behavior of the effective exponent $1/z$, in particular its sign.

1 Introduction

Why computer simulations? A simulation is the imitation of a real physical system or process. Some specific key characteristics are modeled in an artificial environment. In computational sciences simulations are cultivated in order to look into the complex processes of nature, which often are not accessible by other means. Such simulations offer the advantage that they take place in a controlled setting. In particular, one can change the external factors at will, and explore even non-physical regimes. In this way, dependencies of the system properties for many variables can be investigated.

Aim of this thesis. The aim of this thesis is twofold. First and foremost a method was implemented and tested that allows to propagate a quantum mechanical wave packet for long times in large systems with arbitrary geometries. With the code thus developed the quantum dynamics of particles can be simulated and thus the effect of band structure, edges or large scale inhomogeneities may be studied. The technique relies upon what is known in the literature as *Krylov-subspace method*. Our specific implementation is optimized for the treatment of tight-binding lattices, where an efficient parallelization has been installed. As a scientifically relevant test case we have applied our method to the graphene material. The system sizes that we could efficiently deal with have already reached 16384×16384, a factor of two larger that what has been possible before (to the best of our knowledge). [24]

Why graphene? As first application of our simulation tool, we investigated the time evolution of a single particle state on a 2D hexagonal lattice. Such a lattice is well known, especially in the context of *graphene*, a single

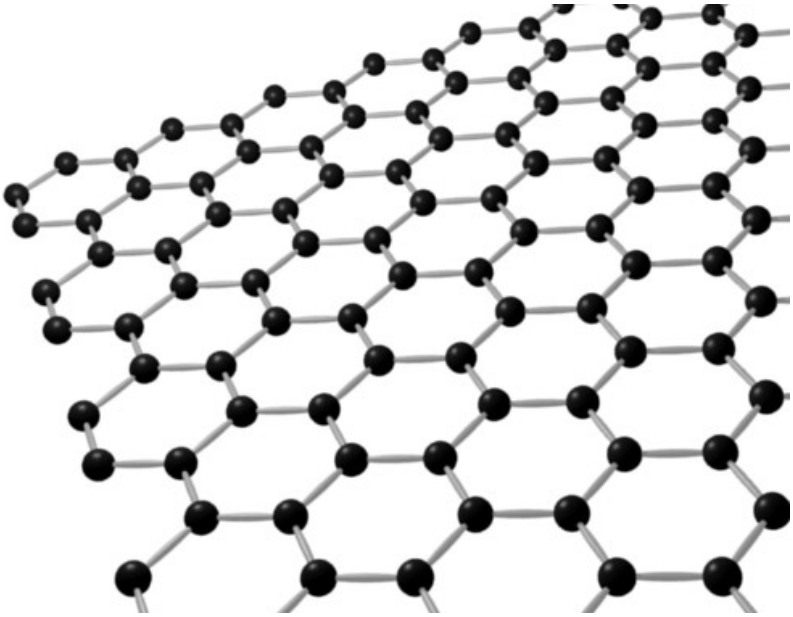


Figure 1.1: A clean graphene honeycomb lattice. The trigonal planar structure comes from the sp^2 hybridization.

sheet of graphite. Graphene is a carbon allotrope, it is a one-atomic thick layer of carbon atoms arranged in a honeycomb lattice. The distance between two carbons is 0.142 nm. This allotrope of carbon has sp^2-hybridized orbitals which leads to the planar structure. The p_z-orbital is perpendicular to the plane and forms with the p_z-orbitals from the other carbon atoms a half filled π-band. Multiple layers stacked onto each other, coupled by weak van der Waals forces, result in graphite, with a inter-planar spacing of 0.335 nm.

The Nobel prize 2010 was awarded to Andre Geim and Konstantin Novoselov "for groundbreaking experiments regarding the two-dimensional material graphene". The awardees wrote a review on graphene in 2009 with the title *The electronic properties of graphene*, where they cover most of the basics [5]. Graphene is currently one of the most active research fields in condensed matter physics.

Studies of graphene have already revealed a lot of new physics and promising applications. It is the first 2D material realized in an experiment; not so long ago nobody did even believe such material could exist. Graphene is a material with very interesting electronic properties due to its unusual electronic spectrum. It could have significant impact in future technologies: graphene-based electronics, composite materials. It has already been discussed that graphene powder for batteries can be produced in mass-production, cheaper than nanotubes [8].

Most interesting to many scientists is the Dirac cone shape in the band-structure, resulting in "relativistic" behavior of particles even at room temperature. [1] The transport properties of graphene are quite special due to this relativistic behavior of the electrons. In experiments a minimal conductance has been measured of a quantum unit $4e^2/h$, discussed in numerous papers [8, 5, 16]. More will be said on the electronic properties of graphene in section 6. Here, we would like to briefly explain our specific interest in the matter. Our focus is on graphene with disorder in the form of vacancies. In other words, we imagine a hexagonal lattice where links have been interrupted so a number of sites is totally disconnected from all the others. We are motivated to do so for two reasons.

(1) In reality, vacancies are rare, also they can attract charge and even spin and thus are not necessarily faithfully modeled by a vacancy in a tight binding lattice; nevertheless, a first impression may be gained. More relevant is the observation that some of the carbon atoms of a graphene sheet may be taken out of their sp^2-hybridization into an sp^3-hybridization by chemical reactants. Such situations may be a common encounter in real experiments, and therefore to model their effect on transport and other properties is of

[1] For example, it was proposed that Klein tunneling can be observed in experiments [3].

interest.

(2) Vacancy disorder in a tight binding lattice is of great interest in itself. Namely, vacancies leave some of the intrinsic symmetries of the hexagonal lattice intact. Specifically, they do not break the so called *chiral* symmetry, which in a sense reflects the fact that hopping is from one sub lattice to an other, always. [14] Hence, one can investigate dynamical properties at the point, where this symmetry is active, namely at the Dirac point at zero energy. Indeed, it is well known that vacancies when added to one sub lattice only, induce a mid gap state each ("zero mode") into the electronic spectrum. When vacancies are given into the second sub lattice also, the zero modes hybridize and give rise to a singular structure of the density of states near zero energy. In the tool developed in this thesis, we can achieve an energy resolution that allows us to investigate the sub gap-structure of the density of states in better detail than it was possible in previous attempts. [24]

1.1 The method implemented in this thesis

The method to propagate in time a wave packet on a lattice is an implementation of Krylov subspace methods. We further looked at various observables and investigated them with our numerics in the presence of disorder. The major achievement of this thesis is the implementation of this simulation technique. The software has been entirely implemented from scratch. We use subspace methods to avoid the full Hamilton matrix diagonalization. We also use standard tools like Lapack [1] functions for diagonalization. The density of states is accessible via a spectral method which requires the Fourier transformation of the correlation function $\langle\Psi(t_0)|\Psi(t)\rangle$. For this task we use the FFTW library [6].

The method is capable of simulating a quantum mechanical single particle

state in a discrete space. We do this by computing the direct numerical solution of the time-dependent Schrödinger equation. The wave function is defined on a set of sites in the Hilbert space, the dynamics being governed by the Hamilton of the system. For a specific physical system its Hamilton operator is needed, the rest of our simulation is independent of the system to study. The numerical propagation method is to compute the time evolution operator for every time-step to iteratively propagate the system with small time-steps.

Parallelization: We tackle the computational difficulties by parallelizing our code. The underlying numerical methods, Krylov subspace method with the GramSchmidt orthonormalization, have the advantage to scale very good in an parallel environment which allows very big system sizes. We distribute the work by splitting the lattice, we would like to to simulate our quantum state on, in multiple chunks. Every process simulates a piece of the system, communicating with its four neighbors to ensure the coherence of the lattice. Taking into account the computational resources at our disposal (e.g. Jülich, EUROPA cluster, 4096 cores, 12h computation time) this allows us to reach unprecedented systems sizes of 16384×16384 sites with an energy resolution better than 10^{-4} in terms of the non-interacting bandwidth. [2]

[2] **Alternative method.** *Split operator method:* The split-operator spectral method ([13, 10]) computes the propagated state vector in time. One starts with the Schrödinger equation $i\partial_t \Psi = -\frac{1}{2M}\nabla^2\Psi + V\Psi$ and approximates the solution to,

$$\Psi(t_0 + \Delta t) = e^{\frac{i\Delta t}{4M}\nabla^2} \cdot e^{-i\Delta t V} \cdot e^{\frac{i\Delta t}{4M}\nabla^2}\Psi(t_0) + \mathcal{O}(\Delta t^3) \tag{1.1}$$

In photonics this method is used to approximate the solution to the Maxwell's wave equation which has the same form as the Schrödinger equation. We avoid application of this method here, since it relies on the representation of the continuum dynamics in terms of a dispersion $\epsilon(\boldsymbol{p})$, where eigenfunctions of $\boldsymbol{p}$ are plane waves. To easily represent vacancies, we prefer to work with lattice-dynamics.

1.2 Organization of this thesis

The thesis is structured in 8 parts. After the introduction comes the first theoretical part which eludes the physical backgrounds of this simulation. Then follows an extensive part about the simulation method The part on the implementation reveals more details how exactly we translate the theory in algorithms, what milestones we passed, the improvements we made over time. Finally, we present the application to graphene, the model we use to describe the physics, our results concerning the density of states and our conclusions. The last section, summarizes our results, evaluates our method and discusses methodological improvements and other applications where the wave packet propagation might be useful.

Recursive Green's function method: Another method used is the recursive Green's function method [9, 15]. The Green's function on an atomic site has a recursive form,

$$G_{00}(E) = \cfrac{1}{E - \cfrac{5h^2}{E - \cfrac{\frac{37}{5}h^2}{E - \dots}}} \tag{1.2}$$

Such an expansion is cut off at some point which results for example in a sum over delta functions for the local density of states at a specific atomic site.

2 Wave packet simulation

In condensed matter physics, surface science and photonics, the simulation of wave packet propagation is widely used. It can provide information about the system dynamics over short and intermediate time scales and thus probes the the full energy spectrum at relatively high but also at lower energies. Hence the method is attractive especially in those situations, where an overview about all the spectrum is required. The energy resolution is limited by two time scales, the observation time, $t_{\rm obs}$, and the time interval, Δt, for the discretized time boosts. As usual, the band width in energy is limited by Δt^{-1} while the energy resolution is fixed by $t_{\rm obs}^{-1}$. Obviously, in order to study low energy excitations one should be able to afford very long observation times.

As a first attempt on time propagation of an initial state, Ψ_0, one might consider to start head on with the time dependent Schrödinger equation,

$$i\frac{\partial}{\partial t}\Psi(t) = H\Psi(t). \tag{2.1}$$

The initial state, $\Psi_0(\mathbf{r})$, could be chosen as a sharply peaked Gaussian wave packet so as to simulate motion of a quantum particle that has been relieved at $t = 0$ at the origin. The wave packet describes the distribution of the probability amplitude of position and momentum for a quantum mechanical time evolution; If one were to discretize (2.1) and hence follow a finite difference time evolution difficulties would be encountered at long observation times. One problem is, that the unitary time evolution is realized only in an approximate way, so that in the long run particle number is not conserved.

In general, approximation schemes are preferable that replace the exact unitary time evolution by another unitary time evolution, that can be repre-

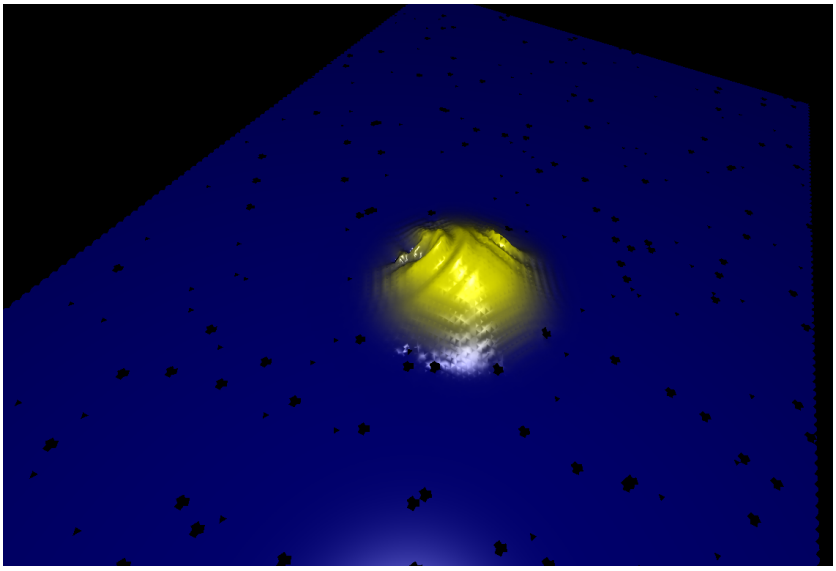


Figure 2.1: An initial state Gaussian wave packet, propagated for a few time steps on a graphene honeycomb lattice with 256x256 sites and 5% vacancy concentration. The initial state was a Gaussian wave packet strongly localized in the center of the lattice. We show how strong the interference effects deforms the wave packet in the presence of strong scatterers like vacancies.

sented in the computer. To see how this work we rewrite Eq. (2.1):

$$\Psi(t + \Delta t) = U(t + \Delta t, t)\Psi(t) \tag{2.2}$$

where we have introduced the time evolution operator $U(t, t')$.

$$U(t, t') = e^{-iH(t-t')} \tag{2.3}$$

We have now for the time evolution over the small time interval Δt the boost operator

$$dU = e^{-\imath H \Delta t} \tag{2.4}$$

To evaluate this operator exactly, e.g. via full diagonalization of H, is not easier then calculating the full time evolution operator $U(t)$. However, powerful schemes may be used, that evaluate the matrix exponential approximately while respecting unitarity. One of them, the split operator method, has already been mentioned in the introduction. A particularly scheme optimized for the very large but sparse matrices we are interested in is based on Krylov-subspace techniques. It will be explained in the next section.

In passing, we mention that propagating a quantum mechanical state in time, allows to calculate many system properties. We will mostly focus on the DoS, but e.g. the diffusion constant or the conductivity can also be obtained straight forwardly

2.1 Tight binding formalism

We would like to propagate a wave packet in time. How exactly the Hamilton operator driving the dynamics looks like depends on the nature of the lattice and the physics we would like to to describe. In our case we use tight binding models, designed to model the electronic structure of a real material near the Fermi energy. In second quantization we can write:

$$H = -t \sum_{<i,j>} (c_i^\dagger c_j + h.c.) \tag{2.5}$$

The operators $c_i^\dagger$ and c_i are the fermion creation and annihilation operators on the lattice site i, the brackets under the sum mean a summation over the nearest neighbors. In our case, the hopping parameter t is the only microscopic parameter – apart from the disorder concentration(s) to be introduced later. It dictates the energy scale, in particular the bandwidth, of

the dispersion relation (here only 1d result)

$$\varepsilon(k) = -2t\cos(ka) \tag{2.6}$$

where a is the lattice constant.

2.2 Graphene specific implementation of the Hamilton

As graphene is the most important application of our numerics we now formulate the corresponding tight binding Hamiltonian. From every site one can hop to exactly three other sites from the other sub-lattice, at least in a clean lattice. Of course it is impossible to simulate an infinite lattice, the behavior at the edges of the lattice needs to be defined. One possibility is to choose hard edges where the wave is reflected into the bulk, the particle would be confined in a box. This approach easily suffers from artifacts related to edge induced Friedel oscillations, however. Another approach would be absorbing edges: the wave would propagate out, disappearing from the lattice. This approach does not conserve particle number and energy. Hence, it is more troublesome to properly implement. In our case we follow standard practice and use periodic boundary conditions. Thus the effective topology of our system becomes torus-like.

We first present and explain the matrices one would need if one were to do explicit matrix operations. However as discussed above, the Krylov methods have the advantage that only the action of the matrix on a vector is needed. Thus the matrices written here are never explicitly represented in our computer code.

The tight binding Hamilton for graphene is as follows :

$$H = \begin{pmatrix} A & B & 0 & \dots & 0 & C \\ B & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & B \\ C & 0 & \dots & 0 & B & A \end{pmatrix} \tag{2.7}$$

The Hamilton is constructed from 3 different types of blocks, A, B and C :

$$A = \begin{pmatrix} 0 & t & 0 & \dots & 0 & t \\ t & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & t \\ t & 0 & \dots & 0 & t & 0 \end{pmatrix} \tag{2.8}$$

$$B = \begin{pmatrix} t & 0 & \dots \\ 0 & t' & \ddots \\ \vdots & \ddots & \ddots \end{pmatrix} , C = \begin{pmatrix} t' & 0 & \dots \\ 0 & t & \ddots \\ \vdots & \ddots & \ddots \end{pmatrix} \tag{2.9}$$

A, B, and C are of size $L \times L$, with L the system size. H is of size $L^2 \times L^2$. The block A is for the hopping between the sites from the same row. Both next diagonals are the hopping to the left and right, the blocks B and C are for the inter row hopping, if $t' = t$ we have a square lattice, to have a honeycomb lattice one only has to set t' to zero.

With theses preliminaries it is easy to define a routine to compute $H|\psi\rangle$.

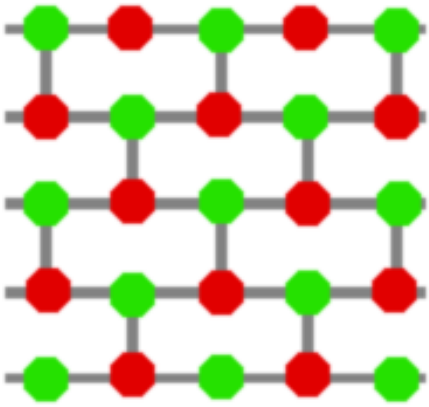


Figure 2.2: Graphene lattice represented as a brick wall, this is how the state is abstracted for the machine, the amplitude at every site is stored row wise in a one dimensional array. Green and red are the different sub-lattices.

We feed the source vector v_s and the destination vector v_d, every element of the destination vector is a sum over three terms corresponding to the three neighbors, Fig. 2.2.

$$v_{d,i} = v_{s,i-1} + v_{s,i+1} + v_{s,i\pm L} \tag{2.10}$$

The sign in the third term depends on which sub-lattice $v_{d,i}$ sits. With this routine we can now span the Krylov subspace (see next chapter). It is quite expensive as it operates on the full length of the state vector, but it is still much faster than a naive approach like storing the Hamilton matrix and implementing a generic matrix vector multiplication. This routine will be called $m - 1$ times, m being the dimension of the Krylov subspace. Our implementation of the Krylov subspace is presented and discussed in Sec. 3.

3 Krylov subspace methods

Our motivation is to propagate a quantum state in time. To compute the solution one can compute the time evolution operator, which is essentially computing a matrix exponent (the matrix being the Hamilton operator) and applying it to the state,

$$U(t, t') = e^{-iH(t-t')} \tag{3.1}$$

There are various methods to achieve this, called "matrix exponential methods", most require to diagonalize a matrix Ref. [15]. The most intuitive method is the exact diagonalization where the full matrix is diagonalized, it works for small systems and yields good results. Doing this for relatively big systems beyond 10^4 sites becomes increasingly difficult even with today's computational resources, not to mention that we would like to to work with more than 10^7 sites. Another method is the Taylor expansion,

$$e^A = I + A + A^2/2! + \dots \tag{3.2}$$

This method is one of the most expensive as it takes long to converge, where converging means adding the next order term does not bring much closer to the result. The problem with this method is the big amplitude of the fluctuations of the expansion. One optimization is to scale the exponent down,

$$e^A = e^{(A/m)*m} \tag{3.3}$$

with m a power of two. m is chosen so that $||A||/m < 1$, after the Taylor expansion the result needs to be squared until the result for the initial problem

is obtained. There are other similar methods, based also on others than the Taylor expansion, but they all have in common to store at least the whole matrix to perform matrix matrix operations. This is a major limitation for big systems, for n sites one needs a Hamilton matrix of size $n \times n$, this does not only use up a lot of memory but also matrix-matrix operations are costly.

Often one does not need the matrix exponent e^A but only the product with a vector $e^A v$. This is exactly our case as we would like to to apply the time evolution operator to the state vector. This allows us do use Krylov methods. They essentially consist of building a low dimensional subspace which we use to project the Hamilton into it. This way only a small matrix has to be diagonalized, in our numerics a 4×4 matrix. That subspace is build around the quantum state, in a way it approximates the time evolution just in the vicinity of the state. The resulting time evolution operator is an approximation, only valid within a small time step. After applying the time evolution operator, the subspace needs to be spanned again around the new state vector.

Krylov subspace methods are very effective numerical methods to diagonalize large (sparse-) matrices or solve large linear systems. First published by and named after Alexei Krylov in 1931, a Krylov subspace of dimension m, generated from a vector v with size s and a s-by-s matrix A, is the linear subspace spanned by $A^i v$ with i from 0 to m,

$$\{v, Av, Av, \ldots, A^{m-1}v\} \tag{3.4}$$

The size m of the subspace is one key parameter of the numerics and chosen depending on the numerical accuracy needed. To span such a subspace in this specific way has the obvious advantage of avoiding matrix-matrix multiplications. Also one can avoid storing the full matrix with an appropriate

subroutine if possible. [3]

3.1 Orthonormalization methods

When constructing a subspace, one has to make sure the basis vectors are orthogonal and normalized. This is important because working with a non-orthogonal basis is complicated. The implementation of the orthonormalization is crucial, the methods might be formally equivalent, but some are more stable as others.

Lets define the matrix M with columns the vectors $A^i v$. Orthonormalizing the vectors is formally equivalent to the decomposition $M = Q \cdot R$, where R is a triangular matrix and Q an orthogonal matrix. Q is the Matrix which columns are the orthonormalized vectors we seek.

Popular methods are for example the Householder transformation and the Givens rotation. The Householder transformation sequentially transforms M column by column in the triangular matrix R. The Givens rotation can be constructed in a way to zero any element of M, but only one at a time. It is obvious that this method is only suitable for sparse matrices, but easier to parallelize than the householder transformation.

The method we used in our numerics is the Gram-Schmidt orthonormalization, it has the advantage to yield a numerical bonus. The Gram-Schmidt

[3] It is interesting to note that to compute the eigenvectors of a matrix A, a simple way is to compute $A^m v$ for large m, the vector $A^m v$ converges with the power m to the eigenvector of A with the largest eigenvalue. To obtain the next biggest eigenvalue one repeats the procedure, but now the vector is chosen orthogonal to the eigenvectors already computed. This is also called the "power method" for finding eigenvalues. This method is numerically not very stable, since round-off errors induce parts from previous eigenvectors which greatly reduce accuracy. It is also a waste of computational effort as all intermediate results are discarded, only the last order of $A^m v$ is kept. This is not the case for the Krylov methods where all power of m enter the approximation.

formalism is as follows,

$$v_i = A^i v - \sum_{k=0}^{i-1} (e_k A^i v) e_k \quad e_i = \frac{v_i}{||v_i||} \tag{3.5}$$

For every basis vector any linear dependency to the other basis vectors is taken out. We will show that the term $e_k A^i v$ can be reused for the computation of the Hamiltonian, projected into the Krylov sub-space.

3.2 Arnoldi methods

Arnoldi with normal Gram-Schmidt: In our numerics we span the Krylov basis and orthonormalize them with the Gram-Schmidt algorithm from above. This method is called the Arnoldi method. It is easy to implement, but numerically not very stable [2].

```
v_0 is random and normalized
for i from 0 to m-2
    v_{i+1} = H v_i
    for k from 0 to i
        r = v_{i+1} - (e_k v_{i+1}) e_k
    e_{i+1} = r / ||r||
```

We use a Krylov basis with a size of 4, going beyond makes the simulation unstable. A small Krylov spaces means we have to use small time steps. The observables we would like to to compute like the DoS are quite sensible to numerical instability, big asymmetrical artifacts appear. We show in the appendix how the artifacts look like induced by too big time steps.

An alternative method which is formally equivalent to the method above is the Arnoldi method with modified Gram-Schmidt. It is more stable which

means the Krylov basis can be bigger, this means the computation of one time step gets much more expensive, but the time steps can also increase. The trade off between Krylov dimension and time step is important. Also one needs to consider if the Hamilton routine is the bottleneck or the orthonormalization. If the Hamilton routine is more expensive, then a bigger time step can be rewarding, but if the Gram-Schmidt is more expensive then a bigger Krylov space gets to costly. This is why, as for us the Gram-Schmidt is quite expensive, we stick with the Arnoldi method with normal Gram-Schmidt. Another important point is the parallelization which is much easier to implement for the normal Gram-Schmidt. We will still present the modified Gram-Schmidt version for completeness, as for a different physical application, meaning another Hamiltonian, the bottleneck might shift towards the Hamilton routine.

Arnoldi with modified Gram-Schmidt: We present the method based on ref. [2, chap. 9]. Instead of computing all the $v_i = H^i v$ and then orthogonalizing them, the Hamiltonian multiplication is operated on an already orthogonal vector $v_{i+1} = He_i$ which is then orthogonalized.

$$v_i = \sum_{k=0}^{i} \beta_k e_k \tag{3.6}$$

$$= H^i v = HH^{i-1}v = H(\sum_{k=0}^{i-1} \beta_k e_k) \tag{3.7}$$

The only part that is kept is the orthogonal part $r_i = \beta_i e_i$. So instead of orthogonalizing $v_{i+1} = Hv_i$ one can orthogonalize $H\beta_i e_i$ or even He_i to

obtain e_{i+1}.

$$\beta_{i+1} e_{i+1} = H e_i - \sum_{k=0}^{i} e_k (e_k H e_i) \tag{3.8}$$

The algorithm looks like this :

```
v_0 is random and normalized
for i from 0 to m-2
    r = H e_i
    for k from 0 to i
        r = r - (e_k r) e_k
    e_{i+1} = r / ||r||
```

The difference to the previous method is that the matrix multiplication is performed on a vector that has already been orthonormalized in the previous iteration, this makes it very stable.

3.3 Hamilton projection in the Krylov subspace

The orthonormalized Krylov basis vector (with normal Gram-Schmidt) looks like,

$$|k\rangle = H^j |0\rangle - \sum_{k'=0}^{j-1} |k'\rangle \langle k'| H^j |0\rangle \tag{3.9}$$

We are at a point where they have been computed, we now analytically prepare the Hamilton operator projection onto the Krylov subspace. To compute the time evolution operator, the Hamilton operator has to be diagonalized. For this purpose we project it onto our low dimensional Krylov basis. The

matrix elements we seek are simply :

$$(H_k)_{ij} = \langle \Psi_i | H | \Psi_j \rangle \tag{3.10}$$

The naive approach would be to directly compute them, which is extremely slow and a big waste of computational effort. To optimize this we first combine 3.10 and 3.9 to,

$$\begin{aligned}
\langle i|H|j\rangle &= \left(\langle 0|H^i - \sum_{k=0}^{i-1}\langle 0|H^i|k\rangle\langle k|\right) H \left(H^j|0\rangle - \sum_{k'=0}^{j-1}|k'\rangle\langle k'|H^j|0\rangle\right) \\
&= \langle 0|H^{i+j+1}|0\rangle \\
&\quad - \sum_{k'=0}^{j-1}\langle 0|H^{i+1}|k'\rangle\langle k'|H^j|0\rangle \\
&\quad - \sum_{k=0}^{i-1}\langle 0|H^i|k\rangle\langle k|H^{j+1}|0\rangle \\
&\quad + \sum_{k=0}^{i-1}\sum_{k'=0}^{j-1}\langle 0|H^i|k\rangle\langle k|H|k'\rangle\langle k'|H^j|0\rangle
\end{aligned} \tag{3.11}$$

$$\langle i|H^j|0\rangle = \langle 0|H^{i+j}|0\rangle - \sum_{k=0}^{i-1}\langle 0|H^i|k\rangle\langle k|H^j|0\rangle \tag{3.12}$$

It is important to see that all the factors can be computed from $\langle 0|H^i|0\rangle$ in a recursive manner. Only those are expensive to compute. We use the vectors of the preliminary basis $H^i|0\rangle$, they are only accessible right before the Gram-Schmidt process. An important optimization was also to realize that the factor $\langle i|H^j|0\rangle$ also appear in the Gram-Schmidt process. Now that all important factors are computed, we can assemble the Hamilton matrix in the Krylov basis. In the next section we diagonalize this matrix and compute the time evolution operator, and then we can calculate observables.

4 Observables and Krylov subspace methods

In this section, we will discuss what observables we calculate with Krylov subspace methods. The last section explained what the Krylov methods are, here we will show how to use them. First we compute the time evolution operator,

$$U(t + \Delta t, t) = e^{-iH_k \Delta t} \tag{4.1}$$

H_k is the Hamilton operator in the Krylov basis, with a size $m \times m$ usually where $m = 4$, which is trivial to diagonalize. Notice that the state vector $|\Psi_t\rangle$ is the first basis vector of the Krylov basis, this means that the state vector in Krylov space is simply $(1, 0, 0, ...)^T$. We operate the time evolution operator on this, the resulting vector is then reconverted to real space, which results in one iteration.

We now have the possibility to propagate the quantum state vector in time. The whole dynamic of the system is contained therein. With the time propagated state vector, one can compute all kind of different observables. We will discuss only a few of them like the density of states. We computed them for different configurations of graphene, with and without disorder, the results are shown in Sec. 6.

4.1 Density of states

The density of states $D(E)$ is the number of available states for electrons per unit volume per unit energy of a solid $D(E) = \frac{dZ}{dE}$, with $Z(E)$ the total number of states up to the energy E.

The analytical expression for the DoS can be derived from the dispersion

relation $\epsilon(k)$. We derive it for the example of a free electron,

$$\epsilon(k) = \frac{\hbar^2 k^2}{2m} \tag{4.2}$$

we obtain for different dimensions,

$$\begin{aligned} 1D: \quad & D(\epsilon) = \tfrac{1}{\pi}\sqrt{\tfrac{2m}{\hbar\epsilon}} \\ 2D: \quad & D(\epsilon) = \tfrac{m}{\pi\hbar^2} \end{aligned} \tag{4.3}$$

Discrete DoS The density of states is only a continuous function of energy if the system is infinite. This is never the case for a physical system, but for reasonable big systems we are in the continuum limit, meaning the number of states becomes so large that the distance between two delta peaks of the spectral density is very small, compared to the resolution with which we look at the data.

The discrete nature of the spectrum of the DoS is one of the reasons to go to large system sizes, to reach the continuum limit with increasing resolution. Due to the Fourier transformation from real space to energy space, the longer the propagation time, the bigger the energy resolution. When approaching the resolution of the mean level spacing it will start resolving the delta peaks of the discrete spectrum of finite size systems. To avoid this one can either reduce the propagation time, or go to bigger system sizes, which means reducing the mean level spacing.

To obtain the formula for the DoS we use for our numerics, we start with the general definition, making use of the Fourier transformation of the delta

function ($\hbar = 1$),

$$\begin{aligned}
\hat{\delta}(t) &= FT(\delta(\epsilon)) = 1 \\
=> \delta(\epsilon) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{it\epsilon}\mathrm{d}t \\
&= \frac{1}{2\pi}\left(\int_{-\infty}^{0} e^{it\epsilon}\mathrm{d}t + \int_{0}^{\infty} e^{it\epsilon}\mathrm{d}t\right) \\
&= \frac{1}{2\pi}\int_{0}^{\infty}\left(e^{-it\epsilon} + e^{it\epsilon}\right)\mathrm{d}t \\
&= \frac{1}{\pi}\int_{0}^{\infty} cos(t\epsilon)\mathrm{d}t \\
&= \frac{1}{\pi}\,\mathrm{Re}\int_{0}^{\infty} e^{it\epsilon}\mathrm{d}t \qquad (4.4) \\
D(\epsilon) &:= \mathrm{tr}\,\delta(\epsilon - H) = \sum_{\alpha}\delta(\epsilon - E_\alpha) \\
&= \sum_{\alpha}\frac{1}{\pi}\,\mathrm{Re}\int_{0}^{\infty} e^{it\epsilon}e^{-itE_\alpha}\mathrm{d}t \qquad (4.5)
\end{aligned}$$

where $\sum_\alpha$ is over all the eigenstates of H. The trace of the Hamilton operator is expressed in the basis of the eigenstates, but it can also be expressed with any random states. This is exact only if all eigenstates contribute, this is the case if summing over an infinite number of random states [24],

$$|\psi_p\rangle = \sum_{\alpha} c_\alpha \psi_\alpha \quad , c_\alpha \in \mathbb{C} \qquad (4.6)$$

The DoS then becomes,

$$\begin{aligned}
D(\epsilon) &= \frac{1}{\pi}\,\mathrm{Re}\int_{0}^{\infty} e^{it\epsilon}\,\mathrm{tr}\,e^{-itH}\mathrm{d}t \\
&= \lim_{s\to\infty}\frac{1}{s}\sum_{p=1}^{s}\frac{1}{\pi}\,\mathrm{Re}\int_{0}^{\infty} e^{it\epsilon}\langle\psi_p|e^{-itH}|\psi_p\rangle\mathrm{d}t \qquad (4.7)
\end{aligned}$$

In practice one only need one state $s = 1$, at least when the system is big

enough,

$$\begin{aligned} D(\epsilon) &= \frac{1}{\pi} Re \int_0^\infty e^{it\epsilon} \langle \psi_0 | e^{-itH} | \psi_0 \rangle \mathrm{d}t \\ &= \frac{1}{\pi} Re \int_0^\infty e^{it\epsilon} C(t) \mathrm{d}t \end{aligned} \tag{4.8}$$

with the correlation function $C(t) = \langle \psi_0 | e^{-itH} | \psi_0 \rangle$. Big systems have very good self averaging, for small systems one needs averaging over multiple random starting vectors $|\psi_0\rangle$. As we will show in more details later on, the computation of the DoS is done in two steps. First the most expensive computation of the correlation function, and second the much faster discrete Fourier transformation.

4.2 Diffusion

Another observable we look into is the diffusion constant $\mathcal{D}$ of the system. The reason we are interested in the diffusion constant is that we were searching for a way to compute transport properties of the system [4]. The Einstein formula relates $\mathcal{D}$ with the conductivity [12],

$$\sigma = e^2 D \cdot \mathcal{D} \tag{4.9}$$

with D the density of states, e the elementary charge and $\mathcal{D}$ the diffusion constant. We already have the possibility to obtain the density of states for any disorder configuration, what is left is to compute the diffusion constant.

The diffusion of particles is related to the time evolution of wave packets.

[4] Other approaches use eigenstates, but eigenstates are not directly available with the Krylov method we implemented, one possibility is to compute so called quasi eigenstates [24]. But it was soon clear that this would be beyond the scope of this diploma thesis.

To compute the diffusion constant we start with a Gaussian wave packet. We construct it with following formula,

$$\Psi(\boldsymbol{r}, \boldsymbol{r_0}, \boldsymbol{k}, s) = Ne^{\frac{-(\boldsymbol{x}-\boldsymbol{x}_0)^2}{2s}} e^{i\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{x}_0)^2} \tag{4.10}$$

with N a normalization coefficient, $\boldsymbol{r_0}$ the center of the wave packet. $\boldsymbol{x}, \boldsymbol{x_0}$ are coordinates in the 2D array the carbon sites are stored such that,

$$(\boldsymbol{r} - \boldsymbol{r_0})^2 = \frac{3}{4}(x - x_0)^2 + (\frac{3}{2}(y - y_0) \pm \frac{1}{2})^2 \tag{4.11}$$

The diffusion constant can be calculated from,

$$\langle r^2(t) \rangle = \int \mathrm{d}\boldsymbol{r} r^2 |\Psi_t(\boldsymbol{r})|^2 \tag{4.12}$$

If a wave packet is strongly localized, meaning it does not propagate much for long times, then we have an insulating system,

$$\langle r^2(t) \rangle \approx \text{const} \tag{4.13}$$

If the wave packet travels diffusely through the system then we have a finite conductivity,

$$\langle r^2(t) \rangle \approx Dt \tag{4.14}$$

If a wave packet is localized or not depends on the disorder strength. The so called Anderson transition, the transition between a system with insulating and a system with metal characteristics, does not only depend on the amount of disorder but also on the dimensionality of the system. E.g. 1D system becomes insulating with any disorder strength.

5 Implementation

In this diploma thesis, one major aspect is the implementation of the wave packet propagation code. The initial idea in our group to implement a Krylov method for the computation of propagating wave packets came up during the PhD thesis of a colleague, Soumya Bera [4]. He was working on the zero modes introduced by vacancies in graphene, in this context he needed a tool to study the DoS of graphene.

In this section we will describe in detail how we implemented the simulation. Explain the workflow of the different logical modules, like the core simulation or the data structures. An important point is the validation of the simulation, the parameter we monitor. We also show how we compute the DoS and other observables.

The requirements of the simulation evolved with time. First a Krylov subspace method should be used to propagate wave packets in time. After this first milestone was achieved, we expanded the simulation to compute the density of states. This took the biggest amount of time, mainly due to the need for big system sizes to get first results. Considerable amount of time was also spend on the parallelization of the code, and running it on different clusters. The last phase of development was to analyze the diffusion of Gaussian wave packets. The development of the diffusion code is on going and at this point no physical results are available.

Workflow: The workflow is typical for simulations in physics. First parameters for the simulation are defined and loaded via config file or command line. The simulation is launched, a main routine is called sequentially until the desired number of steps is achieved. The number of steps depends on the experiment, one has to make a trade-off between the accuracy and the CPU

time needed. Data is extracted every time-step, stored in a buffer which is flushed to a binary file when it is full. In a second step, after completion of the time evolution, the extracted data must be post processed to obtain the final results. An example workflow is depicted in Fig. 5.1.

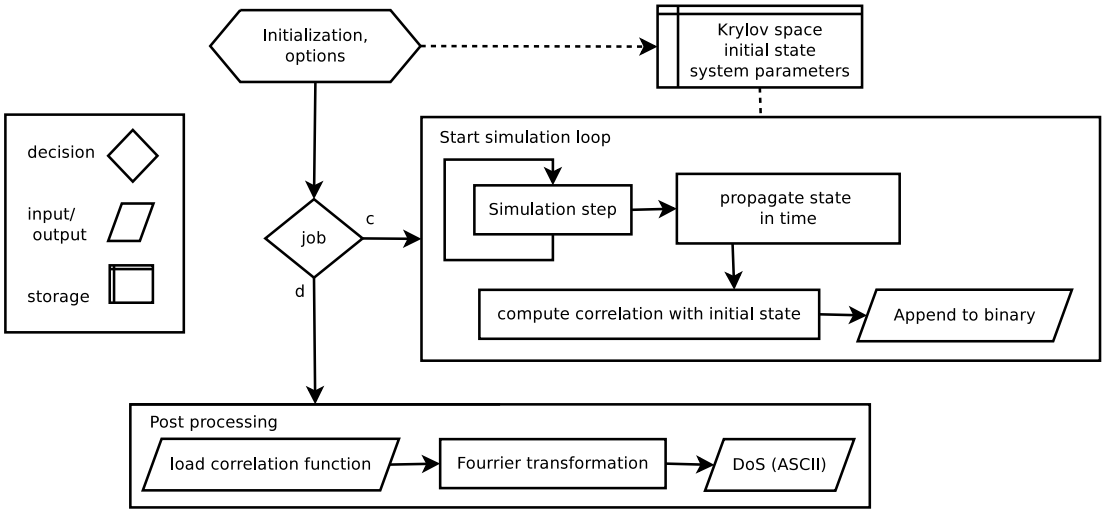


Figure 5.1: The workflow diagram for the computation of the DoS. First the simulation is started with the *job* parameter c. After completion we start the simulation again with $job = d$

Modules: In this software, one can distinguish 3 types of logical modules. The data handling, with the IO operations and the MPI wrapper, are quite important regarding parallelization. Then we have the core simulator consisting of the Krylov methods and the time evolution operator which are necessary for one iteration of time propagation. Finally some auxiliary modules for the different observables, like the DoS and the diffusion code, or the recording module for 3D visualization of the propagating wave packets.

5.1 Core simulation, Krylov methods and time evolution

The core simulation process is the propagation of the system state vector for one time-step. To do this, the Krylov subspace needs to be spanned every time-step from the Hamilton operator and the current state. This is the most expensive part of the simulation, and as such quite optimized. The workflow of the process is a as follows.

```
- span basis
- compute Hamilton in Krylov space
- diagonalize the Hamilton (Lapack method)
- compute time propagation operator
- apply the operator to the state,
   thous propagating it one time step further.
```

Computing the Krylov basis To span the Krylov basis one needs to define the matrix vector multiplication. Knowing the matrix elements is sufficient to work directly on the arrays and avoids to store the full matrix of the Hamilton operator. This is the reason we have a routine for the specific matrix in use, instead of using a matrix object and a generalized matrix vector multiplication. For our Hamilton routine and the implementation of the Krylov subspace see Appendix A.

With this method one can now span the Krylov basis iteratively. The current system state being the first basis state one needs to store m-1 additional vectors with m the desired dimension of the Krylov space. First again a quick overview of the work-flow of the algorithm.

```
- compute the preliminary basis
```

- compute the factors $\langle 0|H^i|0\rangle$,
  important for the computation
  of the Hamilton in Krylov space
- Gram Schmidt orthonormalization
- compute the factors $\langle k|H^i|0\rangle$
- compute the Hamilton in Krylov space

The theoretical background to most of those functions has been discussed in previous sections. The preliminary basis is like discussed above simply $B_{p,m} = \{\Psi_0, H\Psi_0, H^2\Psi_0, \ldots, H^m\Psi_0\}$. To compute the Hamilton operator in Krylov space, the factors $\langle 0|H^i|0\rangle$ for i from 0 to 2m have to be computed before the gram-smith orthonormalization, because we need the original vectors from the preliminary basis $H^i|\Psi_0\rangle$. Tho other factors $\langle k|H^i|0\rangle$ do only need the factors $\langle 0|H^i|0\rangle$ computed the step above. To finally compute the Hamilton in Krylov space is quite simple, all terms are already there.

The Hamilton matrix must now be diagonalized. This is done by the LAPACK method zheev [1] which computes the eigenvalues and eigenvectors which are used for the time evolution. The final time evolution operator is still a matrix in Krylov space, thous we apply it to the state vector in Krylov space which is simply $(1, 0, 0, 0...)^T$, this vector must be multiplied with the time evolution matrix. The resulting vector is converted back in the original space by,

$$\Psi(t) = \sum_{k=0}^{m} \Psi_{krylov}(t)\,[k] \cdot B_m\,[k] \tag{5.1}$$

5.1.1 Density of states

To compute the DoS of the system we use the methods described above to propagate the state in time. Every time step t_i we compute the correlation $C(t_i)$ between the initial and the current state. The obtained discrete and finite function has to be Fourier transformed to get the density of states. The initial state is constructed with random complex double precision numbers,

$$|\psi_0\rangle = \sum_\alpha c_\alpha \psi_\alpha \quad , c_\alpha \in \mathbb{C} \tag{5.2}$$

with $c_\alpha \in [-0.5, 0.5]$, and then normalized.

We apply some transformations to $C(t_i)$ before the Fourier transformation,

$$\hat{C}(t_i) = (-1)^i \cdot f_{HW} \cdot \frac{\Delta_t}{\pi} \cdot C(t_i) \tag{5.3}$$

with i the time step and Δ_t the width of the time step. The integral in (4.8) goes up to infinity, but our correlation function is finite. To alleviate the effects and reduce the artifacts due to the abrupt cut off, we multiply the correlation function with a window function $f_{HW} = 0.5(1+cos(pi*i/N))$. To multiply $C(t_i)$ by $(-1)^i$ is a trick of discrete Fourier transformations which centers the DoS around zero, else the negative values of the DoS would appear after the positive ones. The normalization $\frac{\Delta_t}{\pi}$ comes from (4.8).

The Fourier transformation is done by the FFTW [5] library [6]. The

[5] FFTW is designed to try to compute any input function of arbitrary length in a most optimized way possible. To achieve this, it has an object called a "plan", which contains all needed information for the transformation. To initialize such a plan is relatively expensive because it does quite some optimizations. Once generated, this plan can be reused, even for other input and output arrays if they are similar to the ones the plan was generated for. This can be very important if the Fourier transformation must be done every time-step, but this is not our case.

Fourier transformation we need is only the last step to our final result and takes only a few seconds. The results are stored in ASCII format on disk, easy for every plot program like Gnuplot or XMGrace to visualize.

5.1.2 Diffusion

Instead of choosing a random state, we choose now to build a Gaussian wave packet with zero average velocity. One can deduce the diffusion constant $\mathcal{D}$ with,

$$\langle r^2(t)\rangle \approx \mathcal{D}t \tag{5.4}$$

with $\langle r^2(t)\rangle = \int \mathrm{d}\boldsymbol{r} r^2 |\Psi_t(\boldsymbol{r})|^2$ where $r = 0$ at the center of the wave packet. We compute $\langle r^2(t)\rangle$ and store it. Further development of the diffusion code and the processing of the collected data is done in the diploma thesis of Johannes Schindler [21].

5.1.3 Data-structures

Primarily we store the basis vectors of the Krylov subspace, the first of them being the system state vector itself. For those vectors we have a separate class called *state*, which includes all IO operations, binary and ASCII, that we need to store and load them. The state class comes also with basic math methods like mult() and normalize().

Another important class is called *storage*. It contains the Krylov basis state vectors as well as a copy of the original state needed for the correlation function, and the correlation function itself. The amount a of memory needed

is mainly,

$$a = 16(m+1)k^2 \tag{5.5}$$

with m the size of the Krylov space, k^2 the system size and 16 the size in bytes of the complex data type with double precision.

The simulation parameters are stored in a simple class 'head', it comes with some basic IO operations to write and load itself to a binary file. There is also defined the naming convention for external files. To pass options to the application we use the boost_options library. It parses the command line and a config file *sim.cfg*.

5.1.4 MPI wrapper, parallelization

The first design was not parallelized, and it was not clear in the beginning if this would change. Only during the implementation of the DoS computation the need arose for bigger system sizes. But it was not yet the CPU capacity what capped the system size on a normal workstation, it was the virtual memory needed for the Krylov basis. The average workstation has between 1 and 4 GB of virtual memory. The system size we can simulate with this is around 3600×3600 sites per GB. (calculated with 5.5).

To parallelize the software not only gives a significant boost in speed but also reduces greatly the amount of memory needed per node, making the CPU the new bottleneck. Others limiting factors are specific to the cluster we can use. Obviously the number of nodes is limited depending on the size of the cluster. Also the wall time, the maximum time span allowed for the application to run, is limited. The biggest systems (16384×16384 sites) were computed on the JUROPA cluster in Jülich, with one million time steps in

less than 12 hours.

MPI: The workload of the application has to be distributed on multiple nodes. For this one needs a framework, to launch the application and to take care of the communication between the processes. In the example above on the JUROPA cluster we used 512 nodes with 8 CPUs per node. One of such a framework is MPI (Message Passing Interface) [7]. We give a detailed description on how we wrap the MPI calls in the appendix B.2.

The system is split in equal chunks, each chunk will be the responsibility of one process/core in the cluster. For this trick to be effective the simulation of the bulk should be self sufficient, this means the communication with other processes should be reduced to a minimum. The processes which require some communication between nodes are the exchange of the boundaries at the beginning of each time step, and vector operations like the dot product.

The example studied with this application is a 2D lattice. A graphene patch divided in $N \times M$ sub patches with $k \times k$ sites like a chessboard. The resulting grid (thus the naming of the class) of patches needs to exchange the 1D boundary of each patch with his 4 neighbors.

5.2 Validation

Before computing observables, one has to validate the numerics. It is also important to find the parameter regime for which the simulation works best. Often one has to accept a trade-off between accuracy and time needed for the computation. A last validation is to compute different observables which are easy to obtain and to compare them with literature and other published numerical results, this is done later on for graphene.

Time-line: To validate the simulation, the first step was to plot simulation parameter on a time-line, Fig. 5.2. The quality of the simulated state (in red) is computed from the propagated state in the m dimensional Krylov space. We sum over the last element of the Ritz vector, for the last three time-steps, which should be near zero if the Krylov space approximation is good enough. We also compute the quality of the diagonalization (in green). Diagonalizing

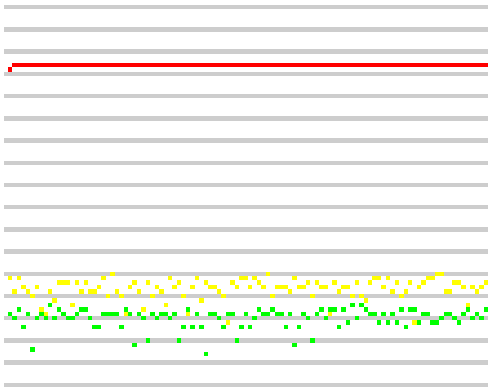


Figure 5.2: We show an example how we plot important parameter on a time line to monitor the simulation. From left to right the time steps and from top to bottom the value of the parameters on a log scale 10^{-p} with p the number of pixels from the top gray line. The green dots are all around 10^{-30}. In red the quality of the approximation for the propagated state vector, in green the quality of the diagonalization of the Hamilton matrix in Krylov space.

a matrix A means decomposing the matrix in three matrices $A = SDS^{-1}$. To check the quality of the diagonalization performed by the LAPACK library, we compute $||A - SDS^{-1}|| \approx 0$. This parameter mostly was around 10^{-30}.

5.3 Visualization

Initially the visualization of the wave packets was only two dimensional. It was a simple pixelated visualization with CImg for debugging purposes, Fig. 5.3. The purpose of the visualization was to validate the propagation of the Gaussian wave packets, and thous of the Krylov method. Seeing a quite realistic propagation of the wave packet was a major milestone.

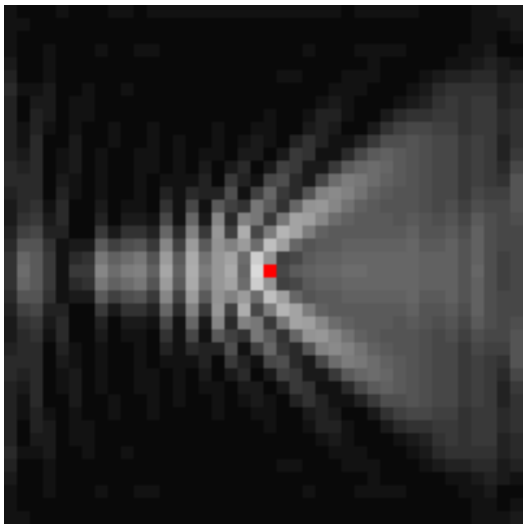


Figure 5.3: Snapshot of a Gaussian wave packet scattering on a single impurity in the middle of a patch of a square lattice. The initial setup was a Gaussian wave packet with a width of 2 and an initial momentum from left to right.

3D Visualization: The nature of our simulation makes it predestined for adding a visualization. Most suitable for a wave packet in graphene is a 3D representation, two dimensions for the lattice and the third dimension for the state amplitude at each site as shown in Fig. 5.4. The simulation was already running well on clusters, adding dependencies like OpenGL would have been detrimental to the portability. This is why we implemented the

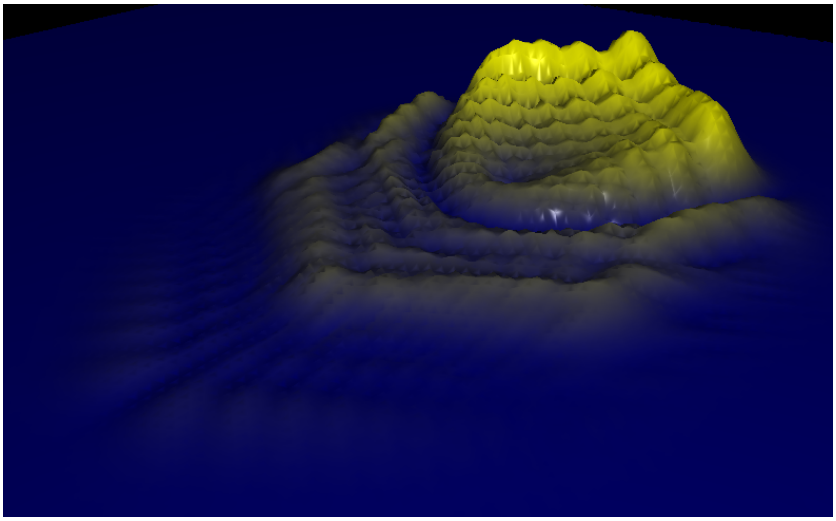


Figure 5.4: Gaussian wave packet, propagated for a few time-steps on a clean graphene honeycomb lattice 256x256 with a single impurity in the middle of the patch, the picture shows the wave packet from behind right after the scattering process.

visualization as a separate tool. The visualization is based on OpenGL, using the OpenGL toolkit GLUT as a window system and OpenSG as a scene-graph. The shader we implemented consists of a classical Phong shader, augmented by a displacement shader, normal recomputing and the coloring of the state depending on the amplitude. This makes it possible to visualize the data in real-time, allowing an easy manipulation of the view perspective for systems up to 512×512 sites. We also added the capability to capture and output frames for animations. The shader code can be found in the Appendix. Prior to the visualization, the data is gathered. We propagate the quantum state with our simulation and store it every N time-steps. The data is visualized afterward in real-time. We made animations for various setups like scattering and double split, using vacancies to build the structures. The aim of this feature is to get a better understanding of the wave packet propagation. Other possibilities are to compare the differences of propagation

on a square lattice, a honeycomb lattice and others. It is also conceivable to visualize exotic effects in graphene like the Klein tunneling. We also took the opportunity to use the virtual reality environment at our disposal which consists of a "cave", an immersive three walls projection system. It allows a profound experience of the wave packet propagation, like being at the shore of some alien beach.

6 Application to graphene

In this section we present our results regarding the application of our numerics to the two dimensional carbon allotrope graphene.

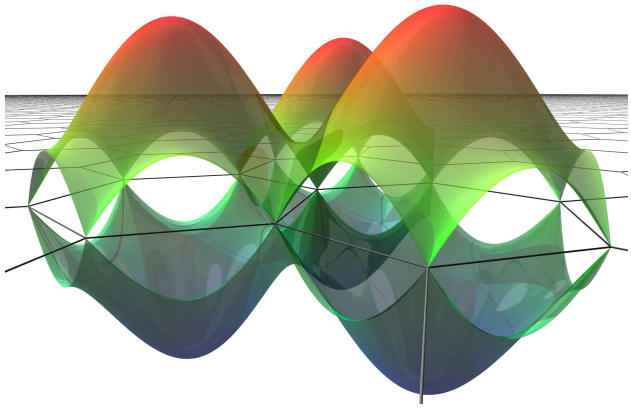


Figure 6.1: Electronic bands of the clean graphene, (courtesy from M. Schütt, Diplomarbeit, Institut für Theorie der Kondensierten Materie, KIT, 2009)

6.1 Graphene, a honeycomb lattice

The 2D hexagonal lattice is the origin of the Dirac cones in the band structure (Fig. 6.1). They allow to observe relativistic effects like the Klein paradox, at room temperature. To investigate how those intriguing effects carry over to transport in the presence of vacancies in graphene. Introducing vacancies is known to dramatically change the shape of the density of states around the Dirac point; zero modes appear exactly at zero energy. To investigate this for various configurations of disorder is the most important physical topic of this thesis. There have been already publications on this topic [4, 20, 19, 23, 11, 22, 24], but with our energy resolution we hope to resolve finer structures and the asymptotic low energy behavior induced by zero mode physics.

6.1.1 Electronic properties of clean graphene, the tight binding approach

As described above in Sec. 2, we use the tight binding approach to describe the electronic lattice model of graphene. The TB Hamiltonian looks like this.

$$H = -t \sum_{i,j} (c_i^\dagger c_j + h.c.) \tag{6.1}$$

We neglect the spin of the electrons, and the hopping to the next nearest neighbors. From this Hamiltonian one can derive the energy bands [5],

$$E_\pm(\boldsymbol{k}) = \pm t\sqrt{3 + f(\boldsymbol{k})} \tag{6.2}$$

$$f(\boldsymbol{k}) = 2\cos(\sqrt{3}k_y a) + 4\cos(\sqrt{3}/2k_y a)\cos(3/2k_x a) \tag{6.3}$$

If one expands this formula near the Dirac points with $\boldsymbol{k} = \boldsymbol{K} + \boldsymbol{q}$ and

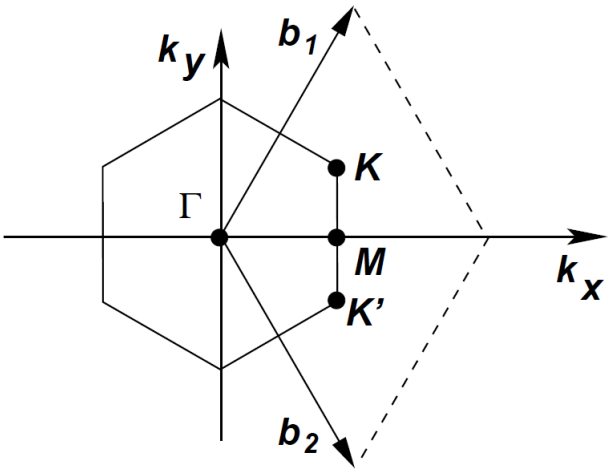


Figure 6.2: Brillouin zone of the clean graphene lattice, adapted from [5], $\boldsymbol{K}$ and $\boldsymbol{K'}$ are the locations of the Dirac cones

$|\boldsymbol{q}| << |\boldsymbol{K}|$ as seen on Fig. 6.2, one gets $E_\pm(\boldsymbol{q}) \approx \pm v_f|\boldsymbol{q}|$, with $v_f = 3ta/2$ the Fermi velocity, t the hopping parameter, and a the lattice constant. This is

a big difference to the usual case where $E(\boldsymbol{q}) = \frac{q^2}{2m}$ and $v = k/m = \sqrt{2E/m}$.

The electronic properties of graphene are quite remarkable as even for room temperatures they exhibit relativistic behavior. This is due to the shape of the band structure around two points in the Brillouin zone $\boldsymbol{K}$ and $\boldsymbol{K'}$, called Dirac points. Centered on each of those two points is a Dirac cone. This means the slope of the energy dispersion around the Dirac point is linear. This corresponds to the solution of the mass less Dirac equation of relativistic particles [5].

With the spinor $\Psi = (\Psi_1, \Psi_2)$ in proximity of the Dirac points, the Dirac equation for mass-less fermions is,

$$v\boldsymbol{p}\boldsymbol{\sigma}\Psi(\boldsymbol{r}) = \epsilon\Psi(\boldsymbol{r}) \tag{6.4}$$

with the velocity v of the fermion of energy ϵ, the momentum operator $\boldsymbol{p} = -i\hbar\partial/\partial\boldsymbol{r}$ and the σ_i the Pauli matrices with $\boldsymbol{\sigma} = \{\sigma_x, \sigma_y\}$. The Hamilton operator near the $\boldsymbol{K}$ point is,

$$H_{\boldsymbol{K}} = \hbar v_F \boldsymbol{\sigma}\boldsymbol{p} = -i\hbar v_F \begin{pmatrix} 0 & \partial_x - i\partial_y \\ \partial_x + i\partial_y & 0 \end{pmatrix} \tag{6.5}$$

with v_F the Fermi velocity. $\boldsymbol{\sigma}$ acts on the AB space of the hexagonal lattice, it corresponds to the pseudo-spin. The wave-function Ψ_K around the Dirac point is,

$$\Psi_{\pm,K}(\boldsymbol{k}) = \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-i\theta_K/2} \\ \pm e^{i\theta_K/2} \end{pmatrix} \tag{6.6}$$

the signs corresponds to the π and π^* bands with the Energies $E = \pm v_F k$ and the angle θ_K is the angle of the momentum k in momentum space. One

can see different characteristics of mass-less particles. When rotating the phase by 2π the wave-function changes sign. This indicates a Berry's phase of π which means that the wave-function is a two component spinor. Ψ_K is also an eigenstate of the helicity $\hat{h} = \boldsymbol{\sigma p}/p$,

$$\hat{h}\Psi_{\boldsymbol{K}}(\boldsymbol{r}) = \pm\frac{1}{2}\Psi_{\boldsymbol{K}}(\boldsymbol{r}) \quad . \tag{6.7}$$

The eigenvalues correspond to the eigenvalues of the pseudo-spin $\boldsymbol{\sigma}$. This means that electrons have positive and holes negative helicity. It is a good quantum number near the Dirac points. The equations for the Dirac point $\boldsymbol{K}'$ can be obtained by applying the time reversal symmetry $k \to -k$. The helicity is also the cause that eigenvalues from $H_{\boldsymbol{K}}$ come in pair, this can be seen when rewriting the Schrödinger equation,

$$\begin{pmatrix} 0 & p_- \\ p_+ & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Psi_2 \end{pmatrix} = E \begin{pmatrix} \Phi_1 \\ \Psi_2 \end{pmatrix} \tag{6.8}$$

with $p_\pm = \partial_x - i\partial_y$. This can be multiplied out to,

$$\left.\begin{aligned} p_-\Psi_2 &= E\Psi_1 \\ p_+\Psi_1 &= E\Psi_2 \end{aligned}\right\} \quad \Rightarrow \quad p_-p_+\Psi_1 = E^2\Psi_1 \tag{6.9}$$

This shows that all eigenvalues come in pairs, with equal amplitude and opposite sign. This is an important conclusion, that the helicity of the Dirac fermions conserves particle-hole symmetry, see also ref. [4, p. 59].

6.1.2 Long range disorder, vacancies and zero modes

There exist different kind of disorder as one can see from table 6.1.

Table 6.1: Different types of disorder in graphene. The rows are the different classes, either conserving C_z chirality leading to Gade-Wegner-type criticality, C_0 chirality, or Λ_z where the disorder is of long-range nature (the valleys are decoupled). In our numerics we use vacancies which conserves time invariance and chirality, which is the origin for zero modes. Adapted from Ref. [14].

Disorder	Symmetries	Class	Criticality
Vacancies, strong potential impurities	C_z, T_0	BDI	Gaede
Vacancies + RMF	C_z	AIII	Gaede
σ_3, $\tau_{1,2}$ disorder	C_z, T_0	CII	Gaede
Dislocations	C_0, T_0	CI	WZW
Dislocations + RMF	C_0	AIII	WZW
Ripples, RMF	Λ_z, C_0	$2 \times$ AIII	WZW
Charged impurities	Λ_z, $T_\perp$	$2 \times$ AII	$\theta = \pi$
Random Dirac mass: $\sigma_3\tau_0$, $\sigma_3\tau_3$	$\Lambda_z, CT_\perp$	$2 \times$ D	$\theta = \pi$
Charge impurities + (RMF, ripples)	Λ_z	$2 \times$ A	$\theta = \pi$

For our simulations we introduce vacancies by cutting all links to a "vacant" site. Vacancies are quite common in real experiments and can be realized (approximately) not only by making a hole in the lattice, but also approximated by breaking the sp^2- hybridization at one atomic site, for example by adding a hydrogen to the carbon atom. Vacancies are in the BDI symmetry class, they preserve chiral and time reversal symmetry. The preservation of the chiral symmetry means that for the missing mode due to the vacancy, there must be a symmetric mode appearing, as the modes can only leave the Hilbert space pairwise. The missing mode from the vacancy can also be thought of as being a mode with zero energy sitting on the atomic cite of the vacancy, thus somewhere else needs to appear another zero mode, an extra zero-energy electronic state exactly at the boundary of valence and conduction band. At least this holds when all vacancies are in one sub-lattice. In this case each vacancy introduces an additional mode at zero energy. When introducing vacancies in both sub-lattices this is not the case anymore. It

has been shown that the number of zero modes is equivalent to the difference of vacancies in both sub-lattices $N_B - N_A$, using a theorem in linear algebra called *index theorem* [20]. In reference [19, 20] is explained why the induced zero modes of the vacancies of the same sub-lattice do not mix. The wave-function of zero modes look like this,

$$\Psi(x, y) \propto \frac{e^{i\boldsymbol{K'r}}}{x + iy} \frac{e^{i\boldsymbol{Kr}}}{x - iy} \tag{6.10}$$

it is localized around the vacancy and decays around it with r^{-1}. Those modes sits always on the sub-lattice with less defects. In addition to the zero modes, vacancies also introduce structure into the DoS away from $E = 0$. This sub gap-structure is what we would like to investigate closer in this thesis.

6.1.3 Transport properties

Graphene has some special transport properties due to the relativistic behavior of charge carriers. The usual setup to measure the conductivity is to place a sample between two doped leads, the graphene flake can be pinned at the Dirac point or doped to change the Fermi energy. Experiments have shown a minimal conductance for clean graphene of a quantum unit $4e^2/h$ in the limit of zero charge carriers, Fig. 6.3.

With a view on the Einstein relation,

$$\sigma = e^2 D(E) \mathcal{D} \tag{6.11}$$

it is somewhat surprising, perhaps, that graphene's conductivity takes a finite value to begin with. Indeed, in clean systems $D(E)$ is of the order 1/Volume and hence the finite value of σ could be understood as a consequence of

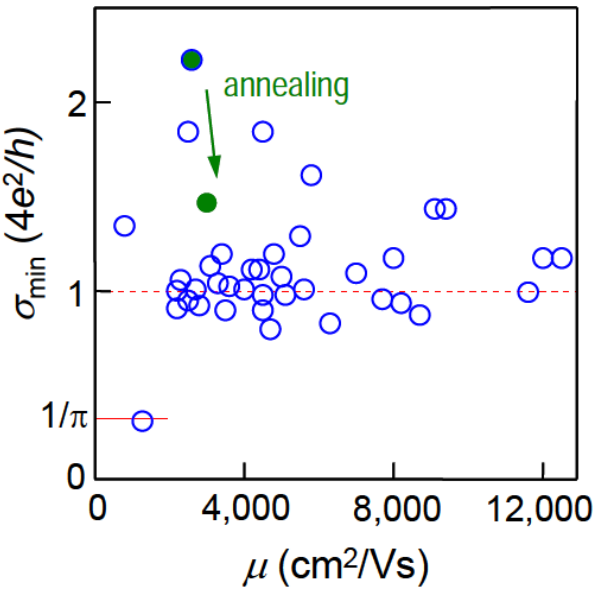


Figure 6.3: Minimal conductivity measured in experiments, the green dot marks the changed conductivity after heating up the sample to reduce macroscopic inhomogeneities. The results are independent of the carrier mobility and show approximately a conductivity of $4e^2/h$. Figure taken from [8]

$\mathcal{D} \sim$ Volume, divergent.

In the presence of disorder, $D(E)$ can be vanishing or diverging dependent on the distribution of vacancies over the sub lattice. In the compensated case, as we shall see, $D(E) \sim 1/E$ and again σ can stay finite only, if $\mathcal{D}$ effectively compensates the singularity by effectively $D(E) \sim E$, i.e. vanishing. Thus there should be a subtle interplay between $D(E)$ and the diffusion process. We will focus on detailed behavior of $D(E)$ in the following and relegate the analysis of diffusion dynamics to a forthcoming project.

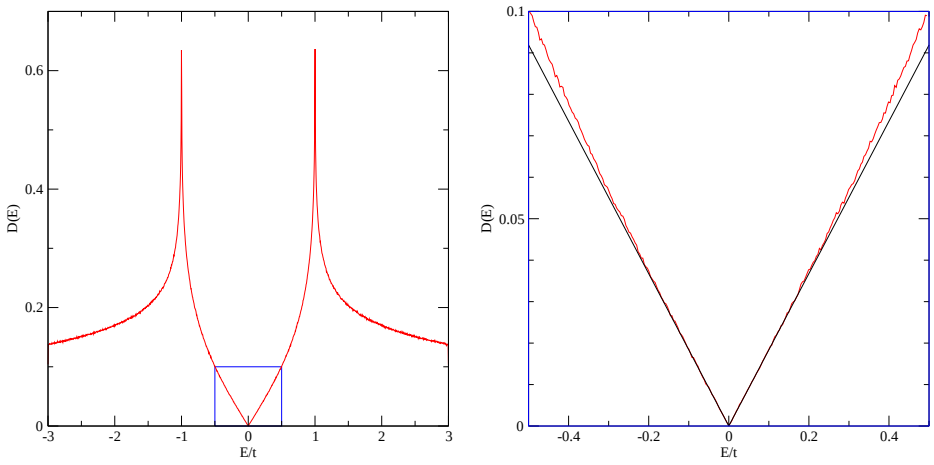


Figure 6.4: DoS of a clean graphene lattice with 8192×8192 sites. The energy resolution is about $3.8 \cdot 10^{-3}$ in units of t. Left: the full DoS. The important features are the van Hove singularities at $E = \pm t$ and the signature of the Dirac point at zero energy where the DoS goes linear in $|E|$. Right: a close up on the Dirac point. The slope of the DoS around the Dirac point is related to the Fermi velocity, E.g 6.16. The linear function in black is the analytical low energy asymptotic which has the slope $\frac{1}{\sqrt{3}\pi}$ as seen in the text.

6.2 Density of states, numerical results

The observables we would like to to compute with our numerics have been introduced in Sec. 4, but some aspects are specific to graphene and thus will also be covered here. The most notable features of the Density of states of clean graphene can be seen in Fig. 6.4. At zero energy the DoS vanishes. This so called Dirac point is induced by the Dirac cone of the electronic band structure. The slope of the DoS around the Dirac point is linear $D(E) \propto |E|$ which reflects the relativistic behavior of particles. The DoS is symmetric around the Dirac point, particle hole symmetry is conserved. This shows again that all eigenvalues come in pairs and result from the chiral nature of

the particles. We present our results for the clean case. The DoS of clean graphene is well understood. We use it to further validate the results of the simulation. We looked at the slope around the Dirac point, which is related to the Fermi velocity. Most aspects related to the implementation and the numerical method like the influence of the total simulation time or the system size are covered in the appendix. We confirm the relation of simulation time to energy resolution, and trace back the origin of our numerical artifacts to the time step and Krylov dimension. In particular we checked the norm of the DoS for the clean and the disordered case (this is important to make sure the normalization factors of the Fourier transformation are correct).

In this section after dealing with the clean case we introduce defects, in particular vacancies, and then present our results for various disorder configurations. We summarize our findings in a "phase diagram" with the total vacancy concentration on one axis and the compensation on the other (the compensation of the lattice is related to the difference in concentrations between the sub-lattices A and B of the system). The phase diagram shows the dependency of the exponent of the power law behavior of the slope around the Dirac point for different disorder configurations.

6.2.1 Clean DoS

The density of states of the hexagonal lattice has quite a few remarkable features. Around zero energy the DoS goes linear to zero from both sides. This is the Dirac singularity, the signature of the Dirac cone. At zero energy $D(E)$ is exactly zero, confirming the semi-metallic behavior of graphene. The band we get from the tight binding model has a spectrum which goes from $-3t$ to $3t$. Deriving an analytical expression for the DoS of clean graphene has already been done in 1953 by Hobson and Nierenberg, we present the

formula from [5, p. 6].

$$D(E) = \frac{4|E|}{\pi^2 t\sqrt{Z_0}} \boldsymbol{F}\left(\frac{\pi}{2}, \sqrt{\frac{Z_1}{Z_0}}\right) \tag{6.12}$$

with the complete elliptical integral of first kind $\boldsymbol{F}(\pi/2, x)$ and Z_0, Z_1 :

$$Z_0 = \begin{cases} \left(1 + \left|\frac{E}{t}\right|\right)^2 - \frac{\left(\left(\frac{E}{t}\right)^2 - 1\right)^2}{4} & ; \quad -t \leq E \leq t \\ \\ 4\left|\frac{E}{t}\right| & ; \quad -3t \leq E \leq -t \vee t \leq E \leq 3t \end{cases} \tag{6.13}$$

$$Z_1 = \begin{cases} 4\left|\frac{E}{t}\right| & ; \quad -t \leq E \leq t \\ \\ \left(1 + \left|\frac{E}{t}\right|\right)^2 - \frac{\left(\left(\frac{E}{t}\right)^2 - 1\right)^2}{4} & ; \quad -3t \leq E \leq -t \vee t \leq E \leq 3t \end{cases} \tag{6.14}$$

$$\boldsymbol{F}(\pi/2, x) = \int_0^{\pi/2} \frac{d\Theta}{\sqrt{1 - k^2 sin^2\Theta}} \tag{6.15}$$

The elliptical integral is important to understand the structure of the DoS. It goes to infinity for $x = t$ and to $\pi/2$ for $x = 0$. We can now connect the linear dispersion $E(\boldsymbol{k}) \approx v_F \cdot |\boldsymbol{k}| + \mathcal{O}(k^2)$ with $|E| < t$ to the linear slope in the DoS around $E = 0$. With only linear terms of $|E|$ we have $Z_0 = 3/4$ and $Z_1 = 0$, the elliptical integral becomes $\boldsymbol{F}(\pi/2, 0) = \pi/2$. Summarizing we obtain the following expression for the DoS of graphene around the Dirac

point,

$$\begin{aligned} D(E) &= \frac{4|E|}{\pi^2 t\sqrt{3/4}} \boldsymbol{F}\left(\frac{\pi}{2}, 0\right) \\ &= \frac{4|E|}{\pi t\sqrt{3}} \\ &= \frac{2A_c|E|}{\pi v_F^2} \end{aligned} \tag{6.16}$$

with $A_c = 3\sqrt{(}3)a^2/2$ the area of the unit cell of the graphene lattice and the Fermi velocity $v_F = 3t/2$. For $|E| = t$ we have $Z_0 = Z_1 = 4$ the elliptical integral $\boldsymbol{F}(\pi/2, 1)$ becomes infinite. There we expect two van Hove singularities in the DoS. They are indeed present as can be seen in Fig. 6.4. The slope around the Dirac point becomes then $\frac{t}{\sqrt{3}\pi} \approx 0.184t$. We plot the low energy asymptotic in Fig. 6.4. The analytical value for the slope describes the data well.

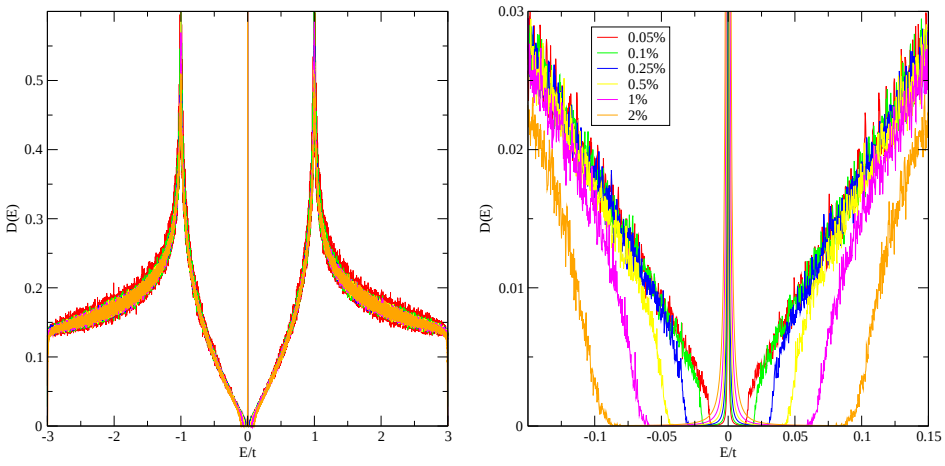


Figure 6.5: DoS of graphene with the vacancies in only one sub lattice. The system size is 2048×2048 with en energy resolution of $2.4e^{-3}$. On the left we show a close up on the Dirac point.

6.2.2 DoS with vacancies in one sub lattice

Introducing vacancies The algorithm which distributes the vacancies places them at random atomic sites up to a given percentage in the first and up to another percentage in the second sub-lattice. Our method avoids to place two vacancies on the same site. As we will see later, it is important to know exactly if the sub-lattices are compensated or not, this means if they have the same number of defects.

When introducing vacancies we distinguish between different special cases. The first two extreme cases can be seen in Fig. 6.5. The defects can be equally distributed in both sub-lattices, this is the "compensated" case. Alternatively, all vacancies can be put only in one sub-lattice, this is the "fully uncompensated" case. Everything in between is also possible and indeed very interesting.

Pseudo-gap We studied how the DoS evolves when augmenting the impurity concentration in one sub-lattice only. We observe how the delta peak forms at the Dirac point. A pseudo gap opens around it with gap width E_g. This has already been discussed in Ref. [4, 20, 24] and is confirmed here.

At zero energy, in the middle of the gap, we see a sharp peak rising, $D(E)$ ideally $N_A \cdot \delta(E)$. The weight under the delta peak corresponds to the concentration of vacancies, which means it corresponds to the number of zero modes per area as seen on the left panel of Fig. 6.6. This confirms what has been discussed above. Every vacancy introduces a zero mode. More generally speaking the weight under the peak equals the difference of vacancies, $N_{ZM} = N_B - N_A$, in both sub-lattices. The numerical data indeed shows $N_{ZM} = \int_{-\epsilon}^{\epsilon} dE\ D(E)$, as presented in the appendix.

We also investigated the method induced broadening of the peak which

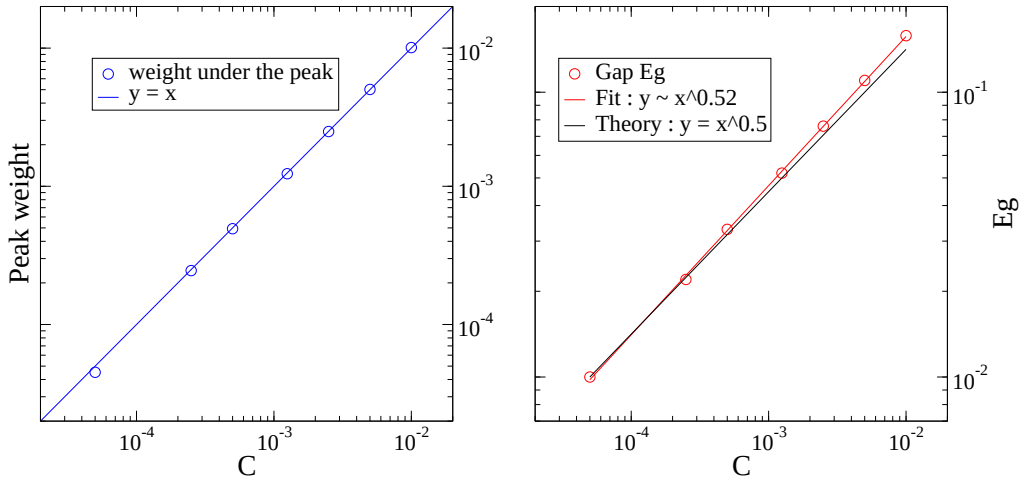


Figure 6.6: On the left panel we show that the weight under the delta peak corresponds to the disorder concentration C, with vacancies in one sub lattice only. We obtained these results by integrating over the whole gap from $-E_g$ to E_g. Right we show how the gap opens up with the increasing concentration of vacancies, we plot the width of the gap over the corresponding vacancy concentration.

is $D(E) \propto |E|^{-2}$. We investigated all simulation parameters to find out the relevant ones. It turned out that the only relevant parameter is the time step used for the time evolution. Reducing the time step also reduces the broadening of the peak, the full discussion regarding the time step and other simulation parameter can be found in the appendix.

The width of the gap gets larger with increasing disorder concentration. The presence of vacancies induce a new energy scale l, the average distance between vacancies.

$$\epsilon \propto \frac{v_F}{l} \propto v_F \sqrt{n_{imp}} \tag{6.17}$$

One expects the gap to open up proportional to the square-root of the concentration of defects [20]. This is indeed the case as can be seen on the right

panel of Fig. 6.6. The numerics fits quite well to the theory. One first inter-

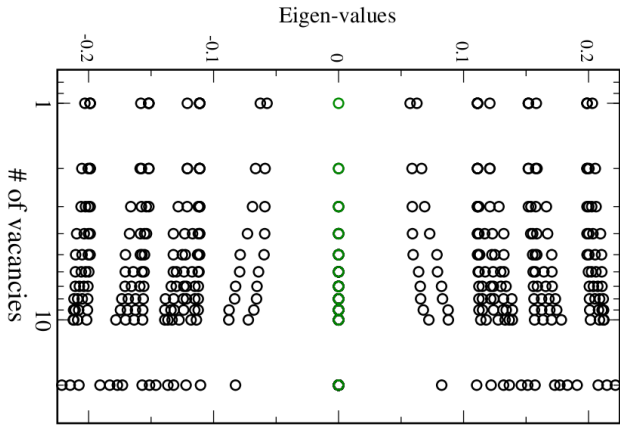


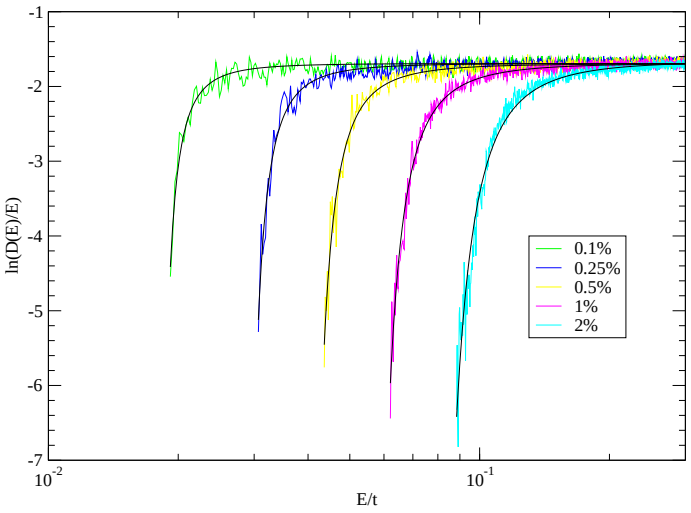
Figure 6.7: The flow of the eigenvalues close to the Dirac point at zero energy when increasing the number of vacancies. Instead of going into the peak with increasing disorder they go away. The zero modes must come from somewhere else. Adapted from the work of Soumya Bera [4, p. 67].

pretation of the formation of the pseudo-gap, was that the eigenvalues close to the Fermi energy flow into the peak. As shown in Fig. 6.7 it appears that this is not the case. Instead of flowing into the central peak they move away with increasing disorder.

The behavior at the gap edge E_g is,

$$D(E) = A_0 \cdot D_0(E) \cdot e^{-\left(\frac{\Delta_g}{E-E_g}\right)^2} \quad , |E| > E_g \tag{6.18}$$

with $D_0(E) \propto E$ the DoS for the clean system near the Dirac point, A_0 an amplitude, Δ_g a length scale and E_g the energy for which $D(E) \to 0$ for $E \to E_g, E > E_g$, equivalent to the gap width.



Disorder	$A_0 \cdot D_0/E$	E_g	Δ_g
0.1%	−1.69	0.0171	0.0034
0.25%	−1.69	0.0270	0.0068
0.5%	−1.70	0.0374	0.0121
1%	−1.69	0.0518	0.0213
2%	−1.67	0.0704	0.0393

Figure 6.8: We plot the gap for different concentrations in one sub lattice (100% disorder in one sub lattice means 50% disorder for the whole lattice). We fit the edge behavior with Eq. (6.18). The fitted parameter are presented in the table.

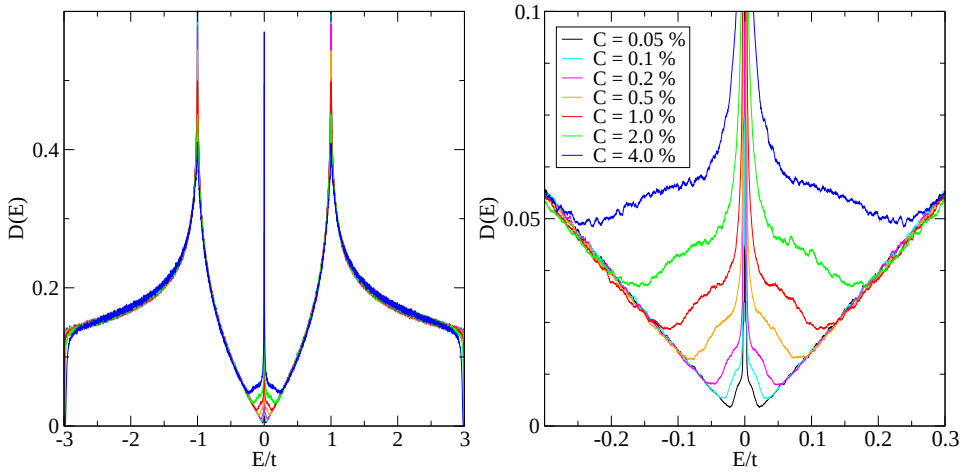


Figure 6.9: DoS of graphene with the same vacancy concentration in both sub lattices. The system size is 2048×2048 with en energy resolution of 10^{-4}. On the left we show a close up on the Dirac point.

6.2.3 Vacancies equally distributed in both sub lattices

In the previous section we put vacancies only in one sub lattice. Here, we have the same concentration of vacancies in both sub lattices. The most important difference is the absence of the gap. Another intriguing observation is the presence of the central peak, now with an intrinsic broadening that is perfectly physical. In the previous case the peak is proportional to the number of zero modes which themselves came from the uncompensation between the sub lattices. When putting the same number of defects on both sub lattices, the delta-peak vanishes because all zero modes hybridize with each other. But as can be seen in Fig. 6.9, another singular structure near zero energy reappears, which originates from the mixing of zero modes in both sub lattices. An analytical approach with strong-coupling effects [14] propose an equation for the low energy spectrum of the DoS in the presence

of disorder of the the symmetry class BDI:

$$D(E) \propto E^{-1} e^{-\frac{1}{2}(3 \cdot g_m^{-1} |\ln(E/\Delta)|)^{2/3}} \tag{6.19}$$

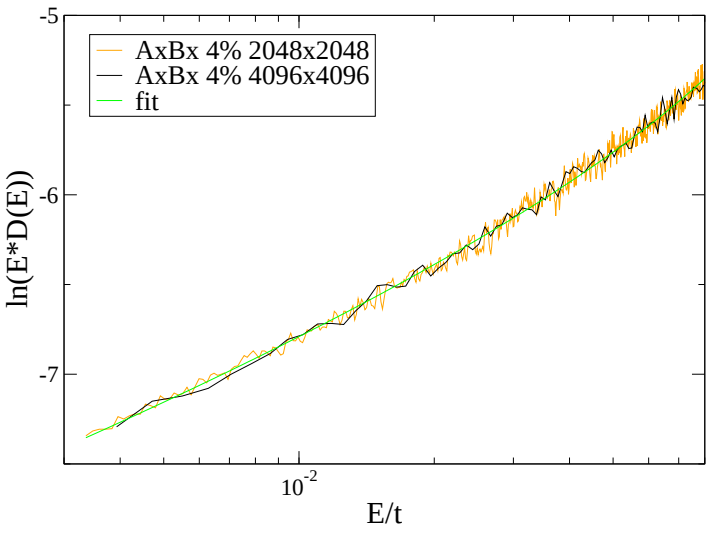
with g_m the random chiral mass. We fit the equation to our data for different system sizes and vacancy concentrations. We transform the equation for the fit,

$$\ln(E \cdot D(E)) = A_0 - \frac{1}{2}(\frac{3}{g_m} |\ln(E/\Delta)|)^{2/3} \tag{6.20}$$

where A_0 fixes the amplitude. We tried the free fit (the exponent 2/3 replaced by a free parameter) but the fit is not stable.

As seen from Fig. 6.10a, the numerical data fit the sublinear dependency of the exponent on $\ln E/t$ well. Note, however, that according to the theoretical formula (6.20), the traces should exhibit a shift along the E-axis with increasing the system size, since Δ denotes the single particle level spacing, $\Delta \sim 1/\text{Area}$. This is not what we observe. Instead, with us the DoS in the reported energy range is not sensitive to the system size. Hence, application of (6.20) for fitting our data is to be taken with a grain of salt. Conceivably, we are not yet in the low energy regime, where the only relevant length scale left is Δ.

We then show how the effective disorder parameter g_m goes to zero with the disorder concentration in Fig. 6.11a.



System	Disorder	A_0	g_m	Δ
2048^2	4%	-4.74	0.90	0.12
4096^2	4%	-4.43	0.81	0.16

Figure 6.10: We plot $\ln(E \cdot D(E))$ over E for two different system sizes and compensated disorder, $\Delta t = 0.05$ for the small system and 0.01 otherwise. Analytical results are fitted with a non linear fit (6.20).

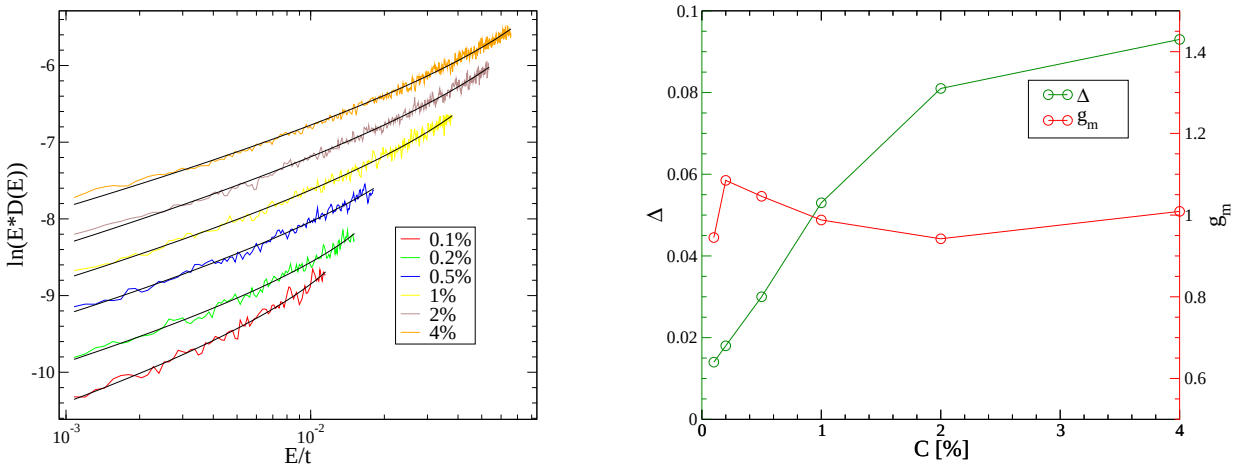
6.2.4 Disorder in both sub lattices

Now we slowly raise the defect concentration in one sub-lattice, Fig. 6.12. The gap begins to close, filling up from the edge. While the qualitative behavior has been reported by earlier authors already, the detailed sub gap-structure of the DoS has not been resolved before – to the best of our knowledge.

We introduce the compensation parameter η,

$$\eta = \frac{c_\mathrm{A} - c_\mathrm{B}}{c_\mathrm{A} + c_\mathrm{B}} \quad c = \frac{c_\mathrm{A} + c_\mathrm{B}}{2} \tag{6.21}$$

$\eta \in [0, 1]$ parametrizes the amount of uncompensation between both sub-



System	Disorder	A_0	g_m	Δ
2048^2	0.1%	−8.334	0.945	0.014
2048^2	0.2%	−7.858	1.085	0.018
2048^2	0.5%	−6.960	1.046	0.030
2048^2	1%	−6.151	0.988	0.053
2048^2	2%	−5.418	0.942	0.081
2048^2	4%	−5.011	1.009	0.093

Figure 6.11: We plot $\ln(E \cdot D(E))$ over E like in Fig. 6.10a, for compensated disorder with different vacancy concentrations. On the other plots we show the fitting parameters, the level spacing Δ and the coupling constant g_m.

lattices A and B, c is the total vacancy concentration in the graphene flake. The evolution with η gets quite complicated now. We will indicate some characteristic energy scales, represented also on Fig. 6.13. We still have a peak near $E = 0$. The energy of the full gap is E_g (fig 6.8a). In between we distinguish three different regimes.

Low energy regime: The low energy regime $D(E) \propto |E|^2$ is only visible for very uncompensated lattices, $\eta << 1$, as shown in Fig. 6.13. The $|E|^2$ regime terminates at an upper scale E_t where it gives way to a transient regime with an intermediate effective power law. E_t is indicated in Fig. 6.13

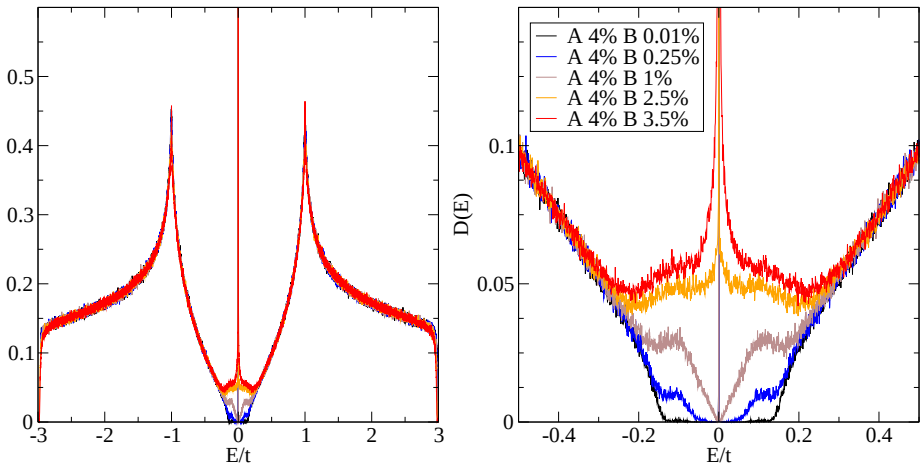


Figure 6.12: DoS of graphene with disorder, filling up with vacancies from the uncompensated to the compensated case. The system size is 4096×4096 with en energy resolution of $6 \cdot 10^{-4}$. On the left we show a close up on the Dirac point.

right by a small circle.

Power law regime: The next regime shows an intermediate power law behavior $|E|^{1/z}$ as we show in Fig. 6.13, right. We fit the exponent $1/z$ for different disorder mismatch η and present the results in a phase diagram 6.16 below. The axes are η and the total disorder concentration c, the amplitude is the color coded exponent $1/z$. We notice that the exponent changes sign around $50\% - 60\%$ compensation.

Our data do not allow to clearly determine where this intermediate power law starts for all η. In other words, it is difficult to nail own $E_t(\eta, c)$. An attempt has been made Fig. 6.15b, right. It is not clear, whether or not the $|E|^2$ behavior at lowest energies remains as η becomes smaller than $\eta \approx 0.72$.

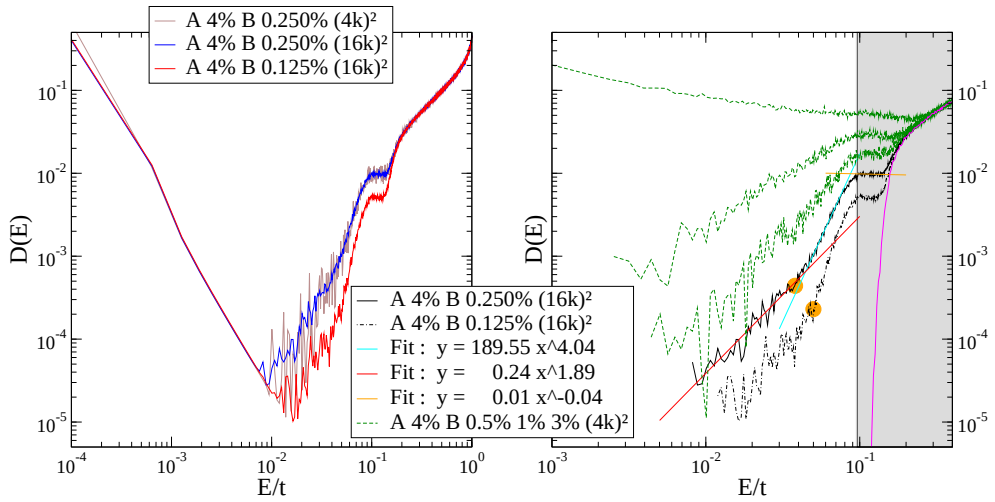


Figure 6.13: Left: the behavior around the delta peak. The raw data for 4096×4096displays a behavior at lower energies (strong power law increase) that reflects the δ-peak broadening of our method. This is an artifact that must be removed for interpreting physical data. We did two big systems (blue and red) with 16384×16384 sites and were able to resolve much better the behavior around the delta peak. Right: we propose a schematic description, for a low vacancy concentration we observe two power laws which is not the case for bigger concentrations. It seams that for a given energy the power laws go into a plateau (Grey area).

Plateau regime: The third regime is a local maximum, a "plateau". We show how the height of the plateau depends on the compensation η in Fig. 6.15a. As can be seen in Fig. 6.13 the point E_p where the DoS changes into the plateau regime does not strongly depend on the amount of uncompensation. This point also coincides with the band gap E_g as shown in Fig. 6.14. After the plateau the DoS resumes the clean behavior with the van Hove singularity at $|E| = t$.

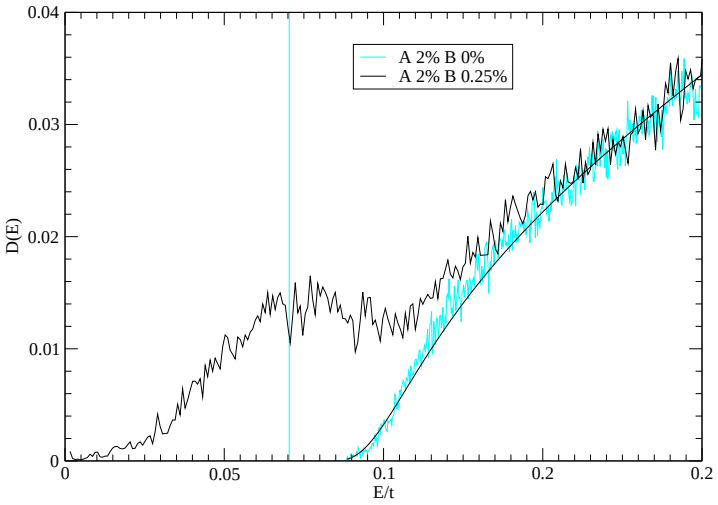


Figure 6.14: We plot the edge of the gap for the uncompensated case with 2% vacancies in one sublattice, the vertical line corresponds to the gap width $E_g = 0.0704$ from tab. 6.8b. The energy scale E_p where the plateau begins coincide well with E_g.

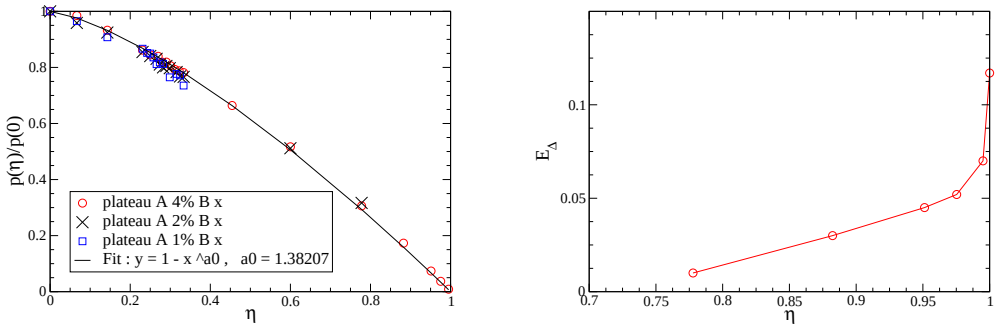


Figure 6.15: Left: Height of the "plateau" (seen in the data Fig. 6.13, right near $E \approx 0.1t$) over the compensation parameter η. The height is normalized to the height of the compensated lattice $p_{4\%}(0) = 0.0566, p_{2\%}(0) = 0.0436, p_{1\%}(0) = 0.0325$. As shown on the right, we expect the regime with the asymptotic behavior of E^2 to vanish with compensation $\eta < 0.72$, this means that in the sub lattice B are at least 28% of the amount of vacancies of sub lattice A. This regime is important for the lower energy physics.

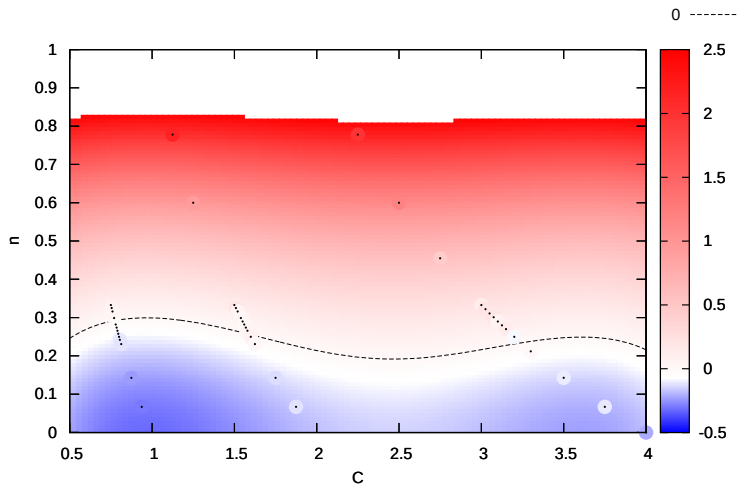


Figure 6.16: The DoS follows a power law, we plot the exponent $1/z$ in a phase diagram over the total concentration of vacancies and the compensation parameter $\eta = \frac{c_A - c_B}{c_A + c_B}$ and the total concentration $c = \frac{c_A + c_B}{2}$. The dotted line is the interpolated exponent value corresponding to a slope of zero. The colored disks with the black dots are the data the surface is fitted on.

7 Outlook

The application of the Krylov method to wave function propagation as it has been implemented in this work has been showing the excellent performance that we have been hoping for: To compute the density of states (Dos) for one disorder configuration on a 4096×4096 lattice, takes less than 24 hours using the 64 cores per job on a local cluster in Karlsruhe, the HC3. However, our system size is not limited to this. Namely, due to the good scalability of our code, we achieved on another cluster, the JUROPA cluster in Jülich, considerably bigger system sizes, 16384×16384 using 4096 cores in moderate 12h.

Due to the fact that (1) the Hamiltonian is never explicitly represented, and (2) the wave package storage is distributed over many nodes, calculations are not limited by computer storage. Instead, the limiting factor is computer time: Large system sizes need more computer time in order for the time evolution to reach the observation time, typically $t_{\text{obs}} \sim 10^6$ time steps, needed for a good energy resolution. The computer time in turn is limited by the "wall time" defined the by the Rechenzentren, e.g., SCC (Karlsruhe) has a wall time of 72h for 64 nodes. Obviously, a future step in code development will be to install a "restart" for the parallel jobs, so that longer observation times much longer than the wall time are possible. A "restart" option at JUROPA together with the necessary total computer time budget opens up possibilities for the treatment of system sizes of the order of 50.000×50.000, we believe. This corresponds to a graphene flake with a lateral extension of 6 μm, which is a scale approaching almost macroscopic dimensions.

In this thesis we have been focusing on the density of states in graphene with vacancy disorder as the observable of the first application of our code. We have been able to resolve a sub gap structure in the DoS near zero energy

and find regimes, that (to the best of our knowledge) have not been discussed in the literature so far.

Here, we would like to emphasize that wave function propagation is also a natural tool to inquire into the quantum dynamics. Therefore, first attempts have been made already within this work to study the diffusion dynamics of the wave package in graphene with vacancies.

This work on diffusion dynamics is to be continued, however, in future endeavors. It has a particular charm in the sense, that it has been shown [17, 18] that the conductivity of the graphene sheet with vacancies is finite, while our work as well as others shows that the DoS of states is singular. Hence, according to the Einstein relation, the diffusion constant must compensate the singularity of the DoS in a sense, and it will be interesting to see, how this works in detail.

Finally, we also note that anomalous structures, like non-analyticities in time (energy), usually also translate into very interesting spatial properties. Indeed, our method can be used to probe the diffusion constant $D(t, \mathbf{r})$ and thus perhaps discover intermediate length scales which signalize anomalous fluctuation properties that the wave function statistics usually exhibits in two-dimensional systems that do not undergo a localization transition.

A Appendix: validation

Simulation parameters: In our simulation we have three parameters which affect the quality of the data: the time step Δt, the total observation time t_{obs} and the system size $L \times L$. Δt describes the coarse graining in time. Thus, it controls the integration stability of the system and, in particular, also the broadening of sharp spectral features, like the delta peak in graphene's DoS near zero energy. This broadening is a simulational artifact without physical meaning. It has the shape $\sim \Delta t^3/E^2$ as seen from Fig. A.1, right. It effects vanishes in the continuum limit $\Delta t \to 0$, Fig. A.1, left. Typical values for Δt in our simulations have been $0.01 < \Delta t < 0.05$ in units of $1/t$.

The total simulation time is inversely proportional to the energy resolution due to the Fourier transformation. For too large simulation times the discrete nature of the spectrum of the finite size system becomes visible and sharp peaks appear. This trend manifests itself as "noise" on the DoS traces. Apart from this, t_{obs} does not have any obvious impact on the behavior of the DoS as seen in Fig. A.2. Typically, the observation time used in our simulation was $t_{\text{obs}}/\Delta t = 10^6$.

The system size, L, is important when we would like to augment the energy resolution. A bigger system means a smaller mean level spacing. This in turn implies, that we can resolve finer spectral structures and thus less "noise" for the same observation time. Apart from the energy resolution, the DoS does not change in a systematic way with L, as far as we can see on Fig. A.3, left.

Norm of the DoS: The area under the DoS should always be unity. We have checked unitarity of our time evolution with and without defects. Fig. A.4 on page 84 shows that the area is in both cases one.

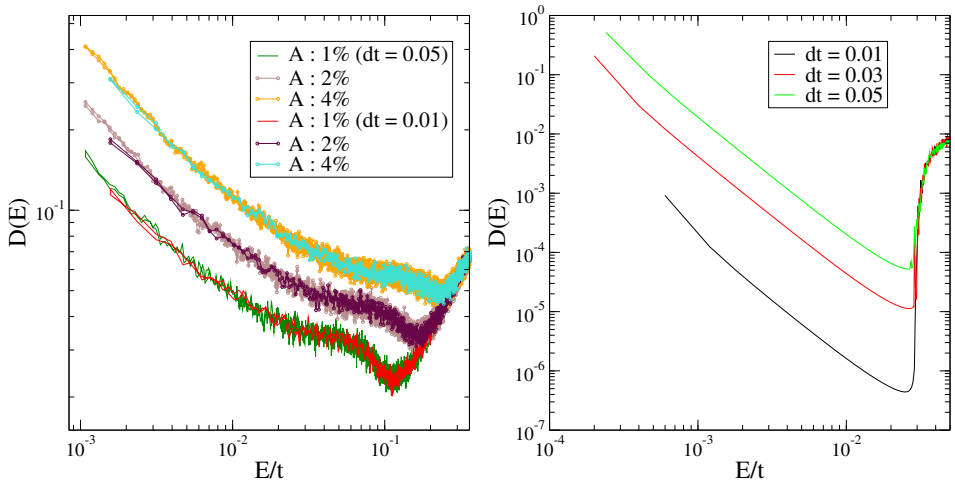


Figure A.1: We check how the width of the time step influence the data. Left we have two kind of data plotted onto each other. A system size 2048×2048 with time step $\Delta t = 0.05$ and 4096×4096 with a time step $\Delta t = 0.01$. The energy resolution for the smaller system is much higher because the total simulation time is longer due to the bigger time step. Up to the last point the data coincides well, the time step does not matter much for the compensated case. Right we show the data for the fully uncompensated case. The system size is 2048×2048 with 0.25% vacancies in sub lattice A and 0% in B. The delta peak is clearly an artifact of our numerics and seams to vanish for smaller time steps.

Broadening of the delta peak at zero energy: As already indicated above, the delta peak acquires a power-law (Lorenzian type) broadening of $\sim E^{-2}$. This fact is emphasized once more in Fig. A.5, where the power law is displayed for different disorder concentrations. The amplitude reproduces correctly the number of zero modes, given by the sub lattice mismatch. It does not change with disorder configuration or any other parameter of the simulation, but the width of the delta peak does change with the total disorder concentration. The width does also change with the length of the time-step, with smaller time-steps the width of the peak gets smaller, the peak as such is an artifact of our numerics, we expect to get a real delta-peak for

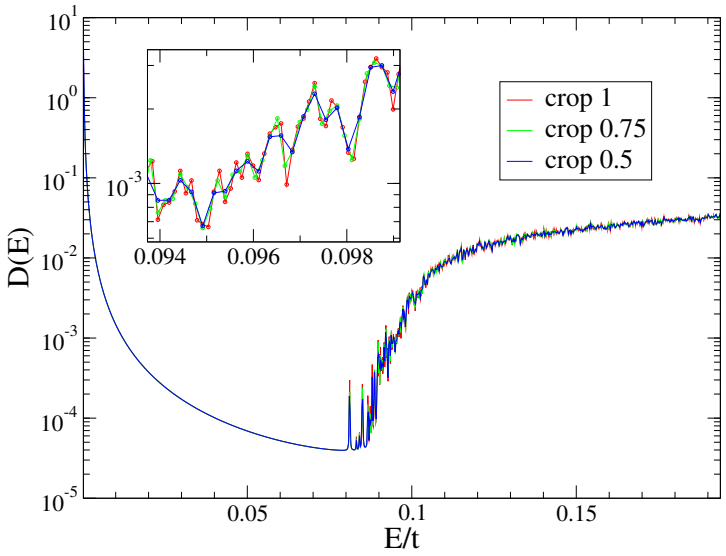


Figure A.2: We check for the uncompensated case how the total simulation time t_{obs} influence the data. The system size is 2048×2048 with 2% vacancies in sub lattice A and 0% in B. We plot the data for three different t_{obs}, we compute the correlation function for a long t_{obs} and than Fourier transform it to obtain the DoS. We do this three times, first with the full correlation function, then we crop it and use 75% and 50% which corresponds to $0.75t_{obs}$ and $0.5t_{obs}$. We conclude that the total observation time does not have any impact on the DoS of graphene.

infinitely small time-steps.

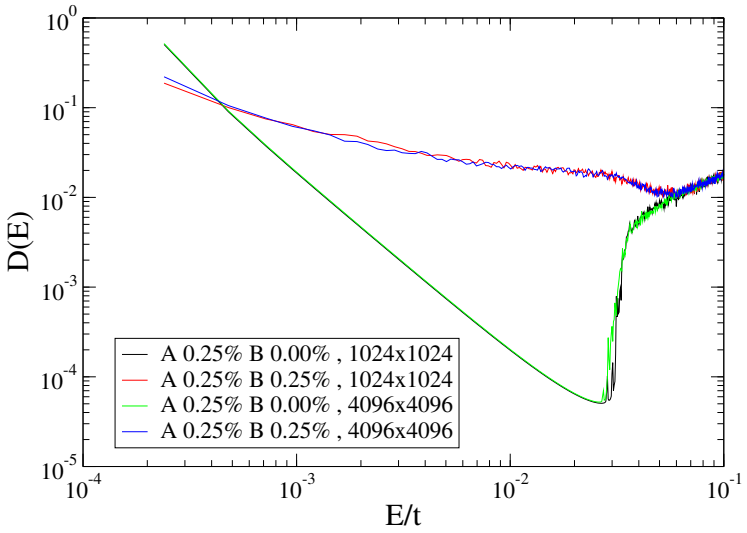


Figure A.3: We check for how the system size $L \times L$ influence the data. We compare two system sizes 1024×1024 and 4096×4096. Data for different system sizes coincide very well. In particular, even the smallest system size is already big enough for the DoS to be self averaging.

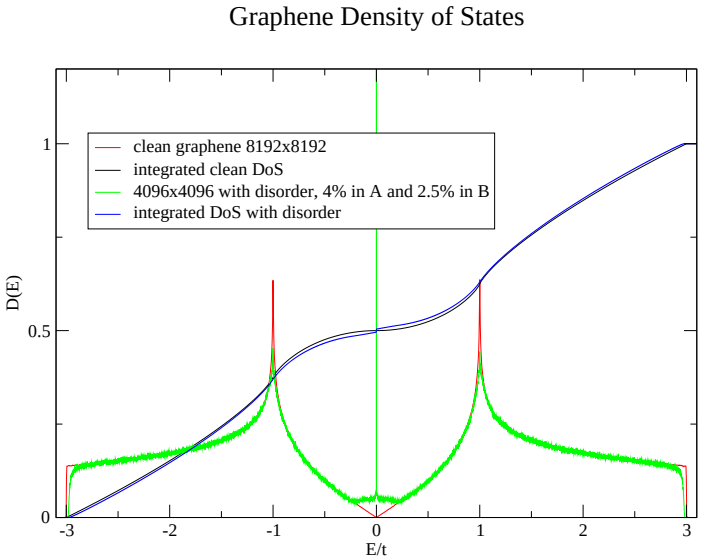


Figure A.4: We check if the DoS is properly normalized. The area under the DoS for clean and disordered graphene is in both cases one.

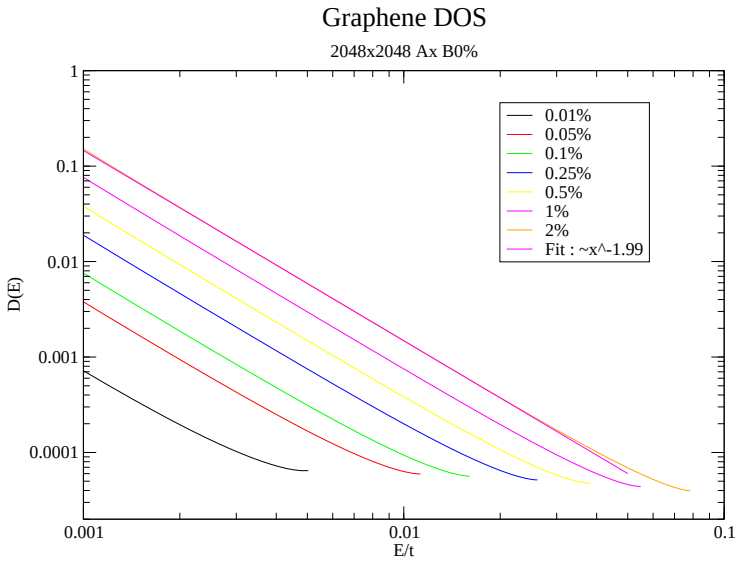


Figure A.5: Peak of the DoS for different disorder concentrations, fully uncompensated case. The power-law dependency E^{-2} of the numerical peak broadening is indicated. ($\Delta t = 0.05$ in this simulation.)

B Appendix: implementation

B.1 Appendix: Krylov method

The simulation can propagate systems without any restriction on the geometry. Below our implementation of the tight binding Hamilton routine for our application to graphene.

```
void HPsi_mpi(state& src, state& res) {
 src.apply_mask(s->defects_mask);
 getBoundsFrom(src);
 gr->getBounds(s->k, s->northbound, s->southbound, s->westbound, s->eastbound);

 for (int i=0;i<k2;i++) {
  res[i] = 0;
  int x = i%s->k;
  int y = i/s->k;
  int g = x + y;

  if (x != 0) res[i] += src[i-1]; //Innerhalb einer Zeile--------------------
  if (x != s->k-1) res[i] += src[i+1];

  //zwischen den Zeilen-----------------------------------
  if (g%2 == 0) if (i < k2-s->k) res[i] += src[i+s->k];

  if (g%2 == 1) if (i >= s->k) res[i] += src[i-s->k];

  if (x == 0) res[i] += s->westbound[y]; //periodizitaet horizontal-------
  if (x == s->k-1) res[i] += s->eastbound[y];

  //periodizitaet vertikal-------
  if (g%2 == 0) if (i >= k2-s->k) res[i] += s->southbound[x];
  if (g%2 == 1) if (i < s->k) res[i] += s->northbound[x];
 }
 res.apply_mask(s->defects_mask);
}
```

At the beginning and at the end the defects mask is applied to the states.

It is an array filled with one and zero, zero for vacancies. When the mask is applied to the state, every element from the mask is multiplied to the corresponding element from the state. The vacancies take effect in the consequence of the above, by artificially maintaining the amplitude on those sites at zero we prevent them from contributing to the hoping.

Parallelization: The lattice is split in multiple patches like a chessboard. The bounds have to be exchanged for every patch of the graphene lattice. If there is no parallelization the single patch uses his own edges to satisfy the periodic boundary conditions. At last each element of the new state is computed from 3 terms (the 3 next neighbors). And now the whole routine:

```
double computeBasis() {
 normC[0]=1; //norm of the current state

 for (int i=0;i<s->m-1;i++) HPsi_mpi(s->krylov_basis[i], s->krylov_basis[i+1]);

 for (int i=0;i<s->m;i++) {//vor gram schmidt!
  o_hh_o[i] = s->krylov_basis[i].mult(s->krylov_basis[0]);
  o_hh_o[i+s->m-1] = s->krylov_basis[i].mult(s->krylov_basis[s->m-1]);
 }

 //gram schmidt: b(i) = H i> - |j><j Hi 0> = w(i) - ...
 cplx c; int i1,i2,i3;
 for (i1=1; i1<s->m; i1++) {// calc b(i)
  for (i2=0; i2<i1; i2++) {
   c = khio(i2, i1);
   for (i3=0; i3<k2; i3++)
    s->krylov_basis[i1][i3] -= s->krylov_basis[i2][i3] * c;//|i> -= c|j>
  }

  normC[i1] = s->krylov_basis[i1].normalize();
 }

 for (int i=0; i<=s->m; i++) {
  for (int j=0; j<s->m; j++) {//k
   k_h_o[i*s->m+j] = khio(j, i);
  }
 }

 return 0;
}
```

B.2 Appendix: MPI

Nearly all MPI calls are wrapped in the "mpi_grid" class.

MPI Initialization

```
MPI_Init(&opt->argc, &opt->argv);
MPI_Comm_size(MPI_COMM_WORLD, &process_N);
MPI_Comm_rank(MPI_COMM_WORLD, &process_i);
MPI_Get_processor_name(processor_name, &name_len);
```

MPI_Init must be called before any other MPI calls, the other three calls get the values for the total number of processes, the id of the current process, and the name of the current node.

MPI Finalization

```
MPI_Finalize();
```

This must be the last MPI call.

Barrier

```
MPI_Barrier(MPI_COMM_WORLD);
```

This call is an effective barrier, its purpose is to synchronize all processes. This function is used in different places in the application, not just in the MPI wrapper.

Communication

```
cplx gatherSum(cplx& c) {
 cplx rbuf[process_N];
 MPI_Gather(&c, 1, type, rbuf, 1,type,0,MPI_COMM_WORLD);

 cplx c_sum = 0;
 if (process_i == 0) for (int i=0;i<process_N;i++) c_sum += rbuf[i];

 MPI_Bcast(&c_sum, 1, type, 0, MPI_COMM_WORLD);
 return c_sum;
}
```

This is one of the most important functions. Its purpose is to gather one complex number from every process at a certain point, add all together and distribute it again to everyone. This is for example used when computing the norm of a state, or for the correlation function.

Implementation

```
int tag = 0;
MPI_Status status;
MPI_Request request[4];

tag = 1;
MPI_Isend (westbound+k, k, type, east, tag, MPI_COMM_WORLD, &request[0]);
MPI_Recv (eastbound, k, type, west, tag, MPI_COMM_WORLD, &status);

tag = 2;
MPI_Isend (eastbound+k, k, type, west, tag, MPI_COMM_WORLD, &request[1]);
MPI_Recv (westbound, k, type, east, tag, MPI_COMM_WORLD, &status);

tag = 3;
MPI_Isend (northbound+k, k, type, south, tag, MPI_COMM_WORLD, &request[2]);
MPI_Recv (southbound, k, type, north, tag, MPI_COMM_WORLD, &status);

tag = 4;
MPI_Isend (southbound+k, k, type, north, tag, MPI_COMM_WORLD, &request[3]);
MPI_Recv (northbound, k, type, south, tag, MPI_COMM_WORLD, &status);

//check the request objects to free internal memory,
//else there is a big memory leak!!!
MPI_Waitall(4, request, MPI_STATUSES_IGNORE);
```

The MPI calls used here are MPI_Isend and MPI_Recv. MPI_Isend sends an array of complex numbers to a specific node, this call is non blocking! This means the sending process will not wait for the send operation to complete. The MPI_Recv call receives the array of data, this call is blocking! It waits until it has terminated. The idea of all this is quite simple. All processes send their surface data to the 'west' and wait for the data coming from the 'east'. When this is done they send their data to the 'east' and wait for data from the 'west', and so on.. The boundary conditions enters during the initialization in the list of neighbors which depends on the index of the

process.

B.3 Appendix: diffusion

First the distances(radius) of every site to the central site are computed and stored

```
void fillRmap() {//all possible R values
 for (int i=-rmax; i<=rmax; i++) {
  for (int j=-rmax; j<=rmax; j++) {
   int r = getIntegerDistance(i+x0,j+y0);
   if(find(Rmap.begin(), Rmap.end(), r) == Rmap.end())
    Rmap.push_back(r);
  }
 }

 sort(Rmap.begin(), Rmap.end());
}
```

The next method goes through all the sites again and match every site with its radius.

```
void fillPmap() {
 fillRmap();

 Pmap.resize(Rmap.size(), vector<int*>());

 for (int i=-rmax; i<=rmax; i++) {
  for (int j=-rmax; j<=rmax; j++) {
   int r = getIntegerDistance(i+x0,j+y0);
   int id = int(find(Rmap.begin(), Rmap.end(), r) - Rmap.begin());
   int* pos = new int[2];
   pos[0] = i+x0; pos[1] = j+y0;
   Pmap[id].push_back(pos);
} } }
```

This is used every time step to sum over the sites with the same radius. That way we already get some averaging.

```
void process(int t) {
 for (int i=0; i<Pmap.size(); i++) {
  for (int j=0; j<Pmap[i].size(); j++) {
   int id = Pmap[i][j][0] + Pmap[i][j][1]*s->k;
   cplx c = s->krylov_basis[0][id];
   prt[t + i*s->T] += norm(c)/Pmap[i].size();
} } }
```

The data is stored in a separate object P_rt.

The data is stored in binary format, the following routine converts it in ASCII format and can average over different files.

```
void average(string path) {
 cout << "\naverage over diffusion data\n";
 vector<string> paths = getPaths(path);
 int N = paths.size();
 if (N == 0) {
  cout << "\nNo files found\n";
  return;
 }

 prt.load(paths[0]);
 prt.setRads(Rmap);

 P_rt tmp;
 for (int i=1; i<N; i++) {
  tmp.load(paths[i]);
  for (int j=0; j<prt.size();j++) prt[j] += tmp[j];
 }

 //generate path string
 string ptmp = "avg_";
 string dir = paths[0];
 size_t pos_slash = dir.find_last_of('/');
 dir = dir.substr(pos_slash+1, 1000);
 ptmp.append(dir);

 //prt.save(ptmp);
 if (opt->order == 't') prt.sort(true);
 if (opt->order == 'r') prt.sort(false);
 prt.writeAscii(ptmp);
}
```

C Appendix: Displacement shader

To accelerate the 3D visualization, we used following shader.

```
// vertex shader program (Phong)
string dcShader::vs_program = //color shader ist jetzt hier dabei!!
"uniform sampler2D displacementMap;\n"
"uniform float dtc;\n"//texel size
"uniform float Dtc;\n"//texture size

"vec4 c_p0  = vec4(0.0, 0.0, 0.5, 1.0);\n"
"vec4 c_p1  = vec4(1.0,  1.0,  0.0,  1.0);\n"
"vec4 c_p2  = vec4(0.0,  1.0,  0.0,  1.0);\n"
"vec4 c_p3  = vec4(1.0,  0.0,  0.0,  1.0);\n"
"varying vec3 ViewDirection;\n"
"varying vec3 fvObjectPosition;\n"
"varying vec3 Normal;\n"
"float s;\n"//scale

//for debugging
"varying vec2 tc;\n"



//------------------------------------------------------------texture2D_bilinear
"vec4 texture2D_bilinear( sampler2D tex, vec2 uv ) {\n"
" vec2 f = fract( uv.xy * Dtc );\n"
" vec4 t00 = texture2D( tex, uv );\n"
" vec4 t10 = texture2D( tex, uv + vec2( dtc, 0.0 ));\n"
" vec4 tA = mix( t00, t10, f.x );\n"
" vec4 t01 = texture2D( tex, uv + vec2( 0.0, dtc ) );\n"
" vec4 t11 = texture2D( tex, uv + vec2( dtc, dtc ) );\n"
" vec4 tB = mix( t01, t11, f.x );\n"
" return mix( tA, tB, f.y );\n"
"}\n"
```

```
//-----------------------------------------------------DISPLACEMENT FKT
"vec4 getDisplacementFromMap( vec2 tc ) {\n"
" vec4 v;\n"
" vec4 c;\n"
" float df;\n"//displacement
//" c = texture2D_bilinear(displacementMap, tc * dtc);\n"
" c = texture2D(displacementMap, tc * dtc);\n"
" s = 1000.0*c[3];\n"
" df = c.x*s + c.y*s*0.00392156862745 + c.z*s*1.53787004998e-05;\n"
" v = vec4(0.0, df, 0.0, 0.0);\n"
" return v;\n"
"}\n"



//-------------------------------------------------------------------MAIN--VP
"void main( void ) {\n"
//displacement shader-----------------

" tc = gl_MultiTexCoord0.xy;\n"
" vec4 disp0 = getDisplacementFromMap(tc);\n"
" vec4 new_v_pos = disp0 + gl_Vertex;\n"

" gl_Position = gl_ModelViewProjectionMatrix * new_v_pos;\n"
```

```
//NORMALS----------------------
" vec4 dispx1 = vec4(0.0, 0.0, 0.0, 0.0);\n"
" vec4 dispx2 = dispx1;\n"
" vec4 dispy1 = dispx1;\n"
" vec4 dispy2 = dispx1;\n"

" dispx1 = getDisplacementFromMap(tc + vec2(1.0, 0.0));\n"
" dispx2 = getDisplacementFromMap(tc + vec2(-1.0, 0.0));\n"
" dispy1 = getDisplacementFromMap(tc + vec2(0.0, 1.0));\n"
" dispy2 = getDisplacementFromMap(tc + vec2(0.0, -1.0));\n"

" vec3 tanx1 = vec3(dispx1 - disp0) + vec3(1.0, 0.0, 0.0);\n"
" vec3 tanx2 = vec3(dispx2 - disp0) + vec3(-1.0, 0.0, 0.0);\n"
" vec3 tany1 = vec3(dispy1 - disp0) + vec3(0.0, 0.0, 1.0);\n"
" vec3 tany2 = vec3(dispy2 - disp0) + vec3(0.0, 0.0, -1.0);\n"
" vec3 normal1 = cross(tany1, tanx1);\n"
" vec3 normal2 = cross(tany2, tanx2);\n"
" vec3 normal3 = cross(tanx2, tany1);\n"
" vec3 normal4 = cross(tanx1, tany2);\n"
" vec3 normal = normal1*0.5 + normal2*0.5 + normal3*0.5 + normal4*0.5;\n"
//" vec3 normal = gl_Normal;\n"



// PHONG-----------------
" fvObjectPosition = vec3(gl_ModelViewMatrix * new_v_pos);\n"
" ViewDirection = - fvObjectPosition.xyz;\n"
" Normal = gl_NormalMatrix * normal;\n"
//" Normal = gl_NormalMatrix * gl_Normal;\n"

//color----------------------
//define the base color depending on vertex height using bezier
" float t = 32.0*new_v_pos.y/s;\n"
//" gl_FrontColor = t*t*t*(-c_p0 + 3.0*c_p1 - 3.0*c_p2 + c_p3)
                + t*t*3.0*(c_p0 - 2.0*c_p1 + c_p2) + t*3.0*(c_p1 - c_p0) + c_p0;\n"
" gl_FrontColor = c_p0 + t*c_p1 - t*c_p0;\n"

"}\n";
```

```
//-------------------------------------------------------------MAIN--FP
// fragment shader program for bump mapping in surface local coordinates (Phong)
string dcShader::fs_program =
"uniform sampler2D displacementMap;\n"
"vec4 fvAmbient  = vec4(0.36, 0.36, 0.36, 1.0);\n"
"vec4 fvSpecular = vec4(0.7,  0.7,  0.7,  1.0);\n"
"vec4 fvDiffuse  = vec4(0.5,  0.5,  0.5,  1.0);\n"
"float fSpecularPower = 25.0;\n"
"\n"
"uniform sampler2D baseMap;\n"
"uniform int useTexture;\n"
"\n"
"varying vec2 tc;\n"
"varying vec3 ViewDirection;\n"
"varying vec3 fvObjectPosition;\n"
"varying vec3 Normal;\n"
"\n"
```

```
"void main( void )\n"
"{\n"
" vec3  fvLightDirection = normalize( gl_LightSource[0].position.xyz
                        - fvObjectPosition.xyz);\n"
" vec3  fvNormal         = normalize( Normal );\n"
" float fNDotL           = dot( fvNormal, fvLightDirection ); \n"
" \n"
" vec3  fvReflection     = normalize( ( ( 2.0 * fvNormal ) * fNDotL )
                        - fvLightDirection ); \n"
" vec3  fvViewDirection  = normalize( ViewDirection );\n"
" float fRDotV           = max( 0.0, dot( fvReflection, fvViewDirection ) );\n"
" \n"
" vec4  fvBaseColor      = gl_Color;\n"
" \n"
" vec4  fvTotalAmbient   = fvAmbient * fvBaseColor; \n"
" vec4  fvTotalDiffuse   = fvDiffuse * fNDotL * fvBaseColor; \n"
" vec4  fvTotalSpecular  = fvSpecular * ( pow( fRDotV, fSpecularPower ) );\n"
" \n"
" gl_FragColor = ( fvTotalAmbient + fvTotalDiffuse + fvTotalSpecular );\n"
//" gl_FragColor = gl_Color;\n"
//" gl_FragColor = texture2D(displacementMap, tc);\n"
"}\n";
```

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[3] C. W. J. Beenakker. Colloquium: Andreev reflection and Klein tunneling in graphene. *Reviews of Modern Physics*, 80:1337–1354, Oct. 2008.

[4] M. S. Bera. *Graphene: Elastic properties, signatures of criticality induced by zero modes and multifractality near a quantum Hall transition*. PhD thesis, Karlsruhe Institute of Technology, June 2011.

[5] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Rev. Mod. Phys.*, 81:109–162, Jan 2009.

[6] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[7] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation.

In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[8] A. Geim and K. Novoselov. The rise of graphene. *Nat Mater*, 6(3):183–91, 2007.

[9] R. Haydock, V. Heine, and M. J. Kelly. Electronic structure based on the local atomic environment for tight-binding bands. *Journal of Physics C: Solid State Physics*, 5(20):2845, 1972.

[10] M. R. Hermann and J. A. Fleck. Split-operator spectral method for solving the time-dependent schrödinger equation in spherical coordinates. *Phys. Rev. A*, 38:6000–6012, Dec 1988.

[11] W.-M. Huang, J.-M. Tang, and H.-H. Lin. Power-law singularity in the local density of states due to the point defect in graphene. *Phys. Rev. B*, 80(12):121404, Sep 2009.

[12] B. Huckestein and R. Klesse. Wave-packet dynamics at the mobility edge in two- and three-dimensional systems. *Phys. Rev. B*, 59:9714–9717, Apr 1999.

[13] A. S. M.D Feit, J.A Fleck Jr. Solution of the schrdinger equation by a spectral method. *J. Comput. Phys.*, 47:412–433, 1982.

[14] A. D. Mirlin, F. Evers, I. V. Gornyi, and P. M. Ostrovsky. Anderson Transitions:. Criticality, Symmetries and Topologies. *International Journal of Modern Physics B*, 24:1577–1620, 2010.

[15] C. Moler and C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[16] P. M. Ostrovsky, I. V. Gornyi, and A. D. Mirlin. Electron transport in disordered graphene. *Phys. Rev. B*, 74:235443, Dec 2006.

[17] P. M. Ostrovsky, I. V. Gornyi, and A. D. Mirlin. Quantum criticality and minimal conductivity in graphene with long-range disorder. *Phys. Rev. Lett.*, 98:256801, Jun 2007.

[18] P. M. Ostrovsky, M. Titov, S. Bera, I. V. Gornyi, and A. D. Mirlin. Diffusion and criticality in undoped graphene with resonant scatterers. *Phys. Rev. Lett.*, 105:266803, Dec 2010.

[19] V. M. Pereira, F. Guinea, J. M. B. Lopes dos Santos, N. M. R. Peres, and A. H. Castro Neto. Disorder induced localized states in graphene. *Phys. Rev. Lett.*, 96(3):036801, Jan 2006.

[20] V. M. Pereira, J. M. B. Lopes dos Santos, and A. H. Castro Neto. Modeling disorder in graphene. *Phys. Rev. B*, 77(11):115109, Mar 2008.

[21] J. Schindler. Numerische wellenfunktionspropagationsmethoden in ungeordneten medien. Diploma thesis, Karlsruhe Institute of Technology, 2012.

[22] L. Schweitzer. Narrow depression in the density of states at the dirac point in disordered graphene. *Phys. Rev. B*, 80(24):245430, Dec 2009.

[23] S. Wu, L. Jing, Q. Li, Q. W. Shi, J. Chen, H. Su, X. Wang, and J. Yang. Average density of states in disordered graphene systems. *Phys. Rev. B*, 77(19):195411, May 2008.

[24] S. Yuan, H. De Raedt, and M. I. Katsnelson. Modeling electronic structure and transport properties of graphene with resonant scattering centers. *Phys. Rev. B*, 82:115448, Sep 2010.