Technische Universiteit Delft
Faculteit Technische Natuurwetenschappen
Faculteit Elektrotechniek, Wiskunde en Informatica

The Kernel Polynomial Method applied to tight binding systems with time-dependence

Verslag als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
Technische Natuurkunde
en
Technische Wiskunde

door

Roeland ter Hoeven

Delft, Nederland
Augustus 2016

Copyright © 2016 door Roeland ter Hoeven. Alle rechten voorbehouden.

BSc verslag TECHNISCHE NATUURKUNDE & TECHNISCHE WISKUNDE

”The Kernel Polynomial Method applied to tight binding systems with time-dependence”

Roeland ter Hoeven

Technische Universiteit Delft

Begeleiders

Dr. A.R. Akhmerov
Dr. N.V. Budko
Dr. D.C. Sticlet

Overige commissieleden

Prof. dr. ir. C. Vuik
Prof.dr. Y.M. Blanter
Dr. J.A.M. de Groot

Augustus, 2016 Delft

Abstract

The Kernel Polynomial Method (KPM) is a method to approximate any function as a polynomial of finite order. In this research the KPM will be used to approximate the density of states and expectation values of physical observables in a tight binding system.

To be able to use the KPM a tight binding Hamiltonian is obtained by using the python package Kwant. A python implementation of KPM combined with a time propagator then results in time-dependent expectation values of any operator.

This approach benefits from large system sizes, since KPM uses a statistical approximation that increases in accuracy with larger size. The statistical approximation always converges, but there is limited guarantee as to how fast it converges.

In further research the KPM could be compared to other available approaches. Another possibility is to use the KPM to get a general overview of the system and combine it with more exact approaches to find specific results.

Contents

1 Introduction

Graphene is a material that has been known for years but could not be produced effectively. In 2010 the Nobel Prize in physics was won for the study of the properties of graphene. This work made it possible to produce graphene and since then a lot of research is being done on the possible applications of graphene.

One of the ways to study solids is to model the solid as a tight binding system. In this model electrons are isolated around the atom they belong to and have little interaction with the surrounding atoms. The Python package Kwant will be used to define a tight binding system, and extract a tight binding Hamiltonian.

When tight binding systems are exposed to light there will be time-dependence entering the system. The expectation values of physical properties like energy or current will therefore become time-dependent. Calculating these expectation values for larger systems exactly will in most cases not be possible. For further physical background on the interaction of tight binding systems like graphene with light, see [1].

In this project the Kernel Polynomial Method (KPM) is examined as a method to approximate these expectation values quickly. The Kernel Polynomial Method is a mathematical method to expand any function as a finite order polynomial. This project is based on a previous publication about the KPM [2]. The first part of this report is similar to the start of that paper, but some additional proofs and explanations are added. After that different physical examples are examined as well as time dependence.

The goal of this project is to use the tight binding Hamiltonian from Kwant to expand physical quantities using KPM. These physical quantities include (time-dependent) expectation values of operators and the density of states.

2 Expectation values and density of states

This section will show the applications of the Kernel Polynomial Method (KPM) used in this project. The next sections will derive the KPM and show that the physical quantities can indeed by approximated by this method. In all of the applications a finite system is required, this means that the Hamiltonian H of the system is of finite dimension D^2. For more information about tight binding Hamiltonians, see an introduction to solid state physics [3]. The eigenvectors $|k\rangle$ of H represent the eigenstates of the system and E_k the eigenvalues. In that case the density of states is given by:

$$\rho(E) = \frac{1}{D} \sum_{k=0}^{D-1} \delta(E - E_k) \tag{1}$$

The expectation value of an operator A in a state $|\psi\rangle$ is defined as $\langle A_\psi \rangle = \langle \psi | A | \psi \rangle$. For an introduction to quantum mechanical expectation values see [4]. The total expectation value of A can be obtained by integrating over all the occupied states of a system.

$$\langle A \rangle = \int_{E_{min}}^{E_{max}} a(E) \rho(E) n(E) \quad \mathrm{d}E \tag{2}$$

Where $a(E)$ is the expectation value of A in a state with energy E, $\rho(E)$ is the density of states and $n(E)$ is the average number of particles in a state with energy E. The integration boundaries E_{min} andE_{max} are finite for finite systems. In this case $\rho(E)n(E)$ can be seen as a weight function corresponding to the amount of occupied states with energy E.

The systems that will be investigated in this project are semiconductors or metals. Graphene is a semiconductor without band gap, which is also called a semi metal. In those cases $n(E)$ is given by the Fermi-Dirac distribution, since the particles are electrons.

$$n(E) = \frac{1}{1 + \exp(\frac{E-\mu}{k_B T})} \tag{3}$$

Where μ is the chemical potential of the system, T is the temperature and $k_B T$ the Boltzmann constant. Further information about this can be found in an introduction to statistical physics, like [5].

Note that the function $a(E)\rho(E)$ is always zero except for the values of E corresponding to the eigenvalues E_k, since $\rho(E)$ is given by delta peaks. The function $a(E)$ is given for the eigenvalues E_k as:

$$a(E_k) = \langle k | A | k \rangle \,, \tag{4}$$

since $|k\rangle$ is the state of the system corresponding to the energy E_k. Combining these results gives:

$$\rho_A(E) \equiv a(E)\rho(E) = \frac{1}{D} \sum_{k=0}^{D-1} \langle k | A | k \rangle \, \delta(E - E_k) \tag{5}$$

Here the notation $\rho_A(E)$ has been introduced for $a(E)\rho(E)$.

If the Hamiltonian is time-dependent $H = H(t)$ the expectation value of A changes over time. This is also the case when the operator A and the Hamiltonian do not commute. The time-dependent expectation value of an operator A in a state $|\psi(t)\rangle$ is then given by:

$$\langle A_{\psi(t)}(t)\rangle = \langle\psi(t)|A|\psi(t)\rangle \tag{6}$$

The total expectation value of A at a time t can then be calculated by integrating.

$$\langle A(t)\rangle = \int_{E_{min}}^{E_{max}} a(E,t)\rho(E)n(E) \quad \mathrm{d}E \tag{7}$$

Where $a(E,t)$ is the expectation value of A in a state with energy E at a time t. The relation between an initial state $|\psi(0)\rangle$ and the state at time t, $|\psi(t)\rangle$ is now derived. In general for a time-dependent Hamiltonian $H(t)$ the state $|\psi(t)\rangle$ is given by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)|\psi(t)\rangle \tag{8}$$

In general this equation is hard to solve, especially for large systems. However in the case that $H(t) = H$, meaning that there is no time-dependence, the solution simplifies. If for a state the initial wave function is given by $|\psi(0)\rangle$, then the wave function at time t is given by

$$|\psi(t)\rangle = \exp(-iHt/\hbar)|\psi(0)\rangle \tag{9}$$

Here since H is a matrix, this exponential is in fact the matrix exponential defined for a square matrix A as:

$$\exp(A) \equiv \sum_{n=0}^{\infty}\frac{1}{n!}A^n \tag{10}$$

This matrix exponential is hard to calculate for any matrix that is not fully diagonalized. When implementing the matrix exponential for numerical results a suitable approximation must be found. This will be discussed in a later section about the implementation of the time propagator.

However when the Hamiltonian is time-dependent (9) does not hold. In that case multiple small time steps dt can be taken to calculate the state at any time. The idea behind this is that $H(t) \approx H(t+dt)$ so that $H(t)$ is basically time independent at the small interval $[t, t+dt]$. Using (9) this means that:

$$|\psi(dt)\rangle = \exp(-iH(0)dt)|\psi(0)\rangle \tag{11}$$

Repeating this operation k times gives $|\psi(kdt)\rangle$:

$$|\psi(kdt)\rangle = \prod_{n=0}^{k-1}\exp(-iH(ndt)dt)|\psi(0)\rangle \tag{12}$$

The following notation will be used to refer to the time propagator:

$$U(q \cdot dt, p \cdot dt) \equiv \prod_{j=p+1}^{q} \exp(-iH(jdt)dt) \tag{13}$$

Using this notation (14) can be written as:

$$|\psi(kdt)\rangle = U(kdt, 0)\, |\psi(0)\rangle \tag{14}$$

The states $|k\rangle$ are now the eigenstates of the initial Hamiltonian $H(0)$ and E_k are the corresponding eigenvalues. The function $a(E_k, t)$ is then given by:

$$a(E_k, t) = \langle k(t)|A|k(t)\rangle \tag{15}$$

Where $|k(t)\rangle$ is notation for the time propagated eigenstates of $H(0)$, meaning $|k(t)\rangle = U(t, 0)\, |k(0)\rangle$.

Combining these results gives:

$$\rho_A(E, t) \equiv a(E, t)\rho(E) = \frac{1}{D} \sum_{k=0}^{D-1} \langle k(t)|A(t)|k(t)\rangle\, \delta(E - E_k) \tag{16}$$

Where $\rho_A(E, t)$ is introduced similar to the time independent case. Note that the operator itself can also be time-dependent so $A = A(t)$. In practice this function $\rho_A(E, t)$ will not be a continuous function of time but discrete. The propagator is defined for a number of time steps and at each time step the function can be calculated.

The Kernel Polynomial Method can be used to approximate the density of states (1), the expectation value of an operator (2) and the time-dependent expectation value (7). In all of the above mentioned formulas the eigenstates $|k\rangle$ of the Hamiltonian are required. The KPM approximation of these quantities does not need any of the eigenstates but instead uses a statistical approach; this will be derived in the coming sections.

3 The Kernel Polynomial Method

3.1 Chebyshev expansion and rescaling

The Kernel Polynomial Method is based on Chebyshev polynomials. In the appendix about Chebyshev polynomials and Chebyshev expansion all the needed relations are proven. The main result of this appendix is that a function $f(x)$ can be written as:

$$f(x) = \frac{1}{\pi\sqrt{1-x^2}}\left(\mu_0 + 2\sum_{n=1}^{\infty}\mu_n T_n(x)\right) \tag{17}$$

Where $T_n(x)$ is the Chebyshev polynomial of order n defined as:
$T_n(x) = \cos(n\arccos(x))$ and μ_n are the expansion coefficients given by

$$\mu_n = \int_{-1}^{1} f(x)T_n(x)\ \mathrm{d}x \tag{18}$$

The main challenge is to calculate moments corresponding to the physical quantities discussed in the last section in an efficient way. More specifically the functions $\rho(E)$, $\rho_A(E)$ and $\rho_A(E,t)$ are the functions that will be expanded. The step from these last two functions to the (time-dependent) expectation values is not the hard part and will be shown later on.

Note that the expanded function is only used on the interval $[-1,1]$, whereas in the last section the function was defined on a finite interval $[E_{min}, E_{max}]$ where E_{min} is the smallest eigenvalue of the Hamiltonian and E_{max} the largest eigenvalue. To be able to use this expansion a rescaling of the eigenvalues is required; the rescaled eigenvalues $\tilde{E}$ will have to be in the range $-1 < \tilde{E} < 1$. Here strict inequality is used to avoid stability problems. To achieve this, the following scaling will be done:

$$\tilde{E} = \frac{E-b}{a}, \tag{19}$$

$$\tilde{H} = \frac{1}{a}(H - bI), \tag{20}$$

with the scaling factors a and b:

$$a = \frac{E_{max} - E_{min}}{2-\epsilon}, \qquad b = \frac{E_{max} + E_{min}}{2} \tag{21}$$

and $\epsilon > 0$ to achieve strict inequality. This rescaling can be proven using that:

$$\frac{E_{max} - E_{min}}{2-\epsilon} < \frac{E_{max} - E_{min}}{2}, \tag{22}$$

Which can be done as follows:

$$\begin{aligned}
E_{min} &\leq E \leq E_{max} \\
\frac{-E_{max}+E_{min}}{2} + \frac{E_{max}+E_{min}}{2} &\leq E \leq \frac{E_{max}-E_{min}}{2} + \frac{E_{max}+E_{min}}{2} \\
\frac{-E_{max}+E_{min}}{2-\epsilon} + \frac{E_{max}+E_{min}}{2} &< E < \frac{E_{max}-E_{min}}{2-\epsilon} + \frac{E_{max}+E_{min}}{2} \\
-a+b &< E < a+b \\
-a &< E-b < a \\
-1 &< \frac{E-b}{a} < 1 \\
-1 &< \tilde{E} < 1
\end{aligned} \tag{23}$$

Now the rescaling of $\tilde{H}$ follows from rescaling of the eigenvalues using some linear algebra. To do this $|k\rangle$ is an eigenvector of H, it then follows from $H|k\rangle = E_k|k\rangle$ that:

$$\tilde{E}_k|k\rangle = \frac{E_k-b}{a}|k\rangle = \frac{E_k|k\rangle}{a} - \frac{b}{a}|k\rangle = \frac{1}{a}H|k\rangle - \frac{b}{a}|k\rangle = \frac{1}{a}(H-bI)|k\rangle = \tilde{H}|k\rangle \tag{24}$$

This also shows that the eigenvectors $|k\rangle$ of H are also the eigenvectors of $\tilde{H}$.

3.2 Calculation of moments

When applying the Kernel Polynomial Method, the moments μ_n will have to be calculated. Two types of moments will be discussed, the first can be used to expand the density of states. The second type is about expectation values of an operator and will be discussed later in this section. The first type is:

$$\mu_n = \langle\beta|T_n(\tilde{H})|\alpha\rangle \tag{25}$$

Where $|\alpha\rangle$ and $|\beta\rangle$ are states of the system described by $\tilde{H}$. The moments can then be calculated using a recursion relation for Chebyshev polynomials derived in the appendix about Chebyshev polynomials

$$T_0(x) = 1 \qquad T_1(x) = x \tag{26}$$

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x) \tag{27}$$

In that case $|\alpha_n\rangle = T_n(\tilde{H})|\alpha\rangle$ gives the following recursion relation for $|\alpha_n\rangle$:

$$\begin{aligned}
|\alpha_0\rangle &= |\alpha\rangle \\
|\alpha_1\rangle &= \tilde{H}|\alpha_0\rangle \\
|\alpha_n\rangle &= 2\tilde{H}|\alpha_{n-1}\rangle - |\alpha_{n-2}\rangle
\end{aligned} \tag{28}$$

Then the moments are given by:

$$\mu_n = \langle\beta|\alpha_n\rangle \tag{29}$$

When $|\alpha\rangle = |\beta\rangle$ this can be done more efficiently using another derived relation in the appendix:

$$2T_m(x)T_n(x) = T_{m+n}(x) + T_{m-n}(x) \tag{30}$$

Now it is possible to calculate two moments μ_n for each $|\alpha_n\rangle$. The following relations then hold:

$$\mu_{2n} = 2\langle\alpha_n|\alpha_n\rangle - \mu_0 \tag{31}$$
$$\mu_{2n+1} = 2\langle\alpha_{n+1}|\alpha_n\rangle - \mu_1 \tag{32}$$

This can be shown using $|\alpha_n\rangle = T_n(\tilde{H})|\alpha\rangle$, and also using the fact that the Hamiltonian is a Hermitian matrix. This means that:

$$\begin{aligned} T_n(\tilde{H})^\dagger &= (\sum_{k=0}^{n} t_k\tilde{H}^k)^\dagger \\ &= \sum_{k=0}^{n} t_k(\tilde{H}^\dagger)^k \\ &= \sum_{k=0}^{n} t_k(\tilde{H})^k \\ &= T_n(\tilde{H}) \end{aligned} \tag{33}$$

This gives for the even terms:

$$\begin{aligned} 2\langle\alpha_n|\alpha_n\rangle - \mu_0 &= 2\langle\alpha| T_n(\tilde{H})^\dagger T_n(\tilde{H}))|\alpha\rangle - \mu_0 \\ &= 2\langle\alpha| T_n(\tilde{H})T_n(\tilde{H})|\alpha\rangle - \mu_0 \\ &= \langle\alpha|(T_0(\tilde{H}) + T_{2n}(\tilde{H}))|\alpha\rangle - \mu_0 \\ &= \langle\alpha|\alpha_0\rangle + \langle\alpha|\alpha_{2n}\rangle - \mu_0 \\ &= \mu_0 + \mu_{2n} - \mu_0 \\ &= \mu_{2n} \end{aligned} \tag{34}$$

And for the odd terms:

$$\begin{aligned} 2\langle\alpha_{n+1}|\alpha_n\rangle - \mu_1 &= 2\langle\alpha| T_{n+1}(\tilde{H})^\dagger T_n(\tilde{H}))|\alpha\rangle - \mu_1 \\ &= 2\langle\alpha| T_{n+1}(\tilde{H})T_n(\tilde{H}))|\alpha\rangle - \mu_1 \\ &= \langle\alpha|(T_1(\tilde{H}) + T_{2n+1}(\tilde{H}))|\alpha\rangle - \mu_1 \\ &= \langle\alpha|\alpha_1\rangle + \langle\alpha|\alpha_{2n+1}\rangle - \mu_1 \\ &= \mu_1 + \mu_{2n+1} - \mu_1 \\ &= \mu_{2n+1} \end{aligned} \tag{35}$$

The density of states of $\tilde{H}$ is given by:

$$\tilde{\rho}(\tilde{E}) = \frac{1}{D}\sum_{k=0}^{D-1}\delta(\tilde{E} - \tilde{E}_k) \tag{36}$$

Before looking at the moments for this function, it must first be shown that the eigenvalues of $T_n(\tilde{H})$ are $T_n(\tilde{E})$. This actually holds for all polynomials and is not a specific property of the Chebyshev polynomials, for the proof see the appendix.

Now by applying (18) the moments can be calculated. It will be used that the trace over a matrix equals the sum of its eigenvalues. Since $T_n(\tilde{E})$ is a scalar it follows that $T_n(\tilde{E_k}) = \langle k|k\rangle\, T_n(\tilde{E_k}) = \langle k|T_n(\tilde{E_k})|k\rangle$. Now for μ_n:

$$\begin{aligned}
\mu_n &= \int_{-1}^{1} \tilde{\rho}(\tilde{E})T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D}\sum_{k=0}^{D-1}\int_{-1}^{1} \delta(\tilde{E}-\tilde{E_k})T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D}\sum_{k=0}^{D-1} T_n(\tilde{E_k}) \\
&= \frac{1}{D}\sum_{k=0}^{D-1} \langle k|T_n(\tilde{E_k})|k\rangle \\
&= \frac{1}{D}\sum_{k=0}^{D-1} \langle k|T_n(\tilde{H})|k\rangle \\
&= \frac{1}{D}\mathrm{Tr}(T_n(\tilde{H}))
\end{aligned} \tag{37}$$

The next section is about a statistical approach to calculate moments of this trace form efficiently.

The second type of moments are used to calculate the expectation value of an operator A. This type can also be used in the time-dependent case, as will be shown.

Remember that $\rho_A(E)$ is the function that will be expanded, the rescaled version of this function is:

$$\tilde{\rho_A}(\tilde{E}) = \frac{1}{D}\sum_{k=0}^{D-1} \langle k|A|k\rangle\, \delta(\tilde{E}-\tilde{E_k}) \tag{38}$$

As seen earlier in this section, the trace of $T_n(\tilde{H})$ can be written using its normalized eigenvectors $|k\rangle$:

$$Tr(T_n(\tilde{H})) = \sum_{k=0}^{D-1} \langle k|T_n(\tilde{H})|k\rangle \tag{39}$$

However now the trace of $AT_n(\tilde{H})$ is wanted and $|k\rangle$ are in general not the eigenvectors of this matrix. It can be shown however that for any orthonomal set of vectors $|u\rangle$, the trace of an operator O is given by $\sum_u \langle u|O|u\rangle$. See the appendix for the proof.

Now we can use (18) for the function $\tilde{\rho_A}(\tilde{E})$.

$$\begin{aligned}
\mu_n &= \int_{-1}^{1} \tilde{\rho_A}(\tilde{E}) T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \int_{-1}^{1} \frac{1}{D} \sum_{k=0}^{D-1} \langle k|A|k\rangle\, \delta(\tilde{E} - \tilde{E_k}) T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D} \sum_{k=0}^{D-1} \langle k|A|k\rangle \int_{-1}^{1} \delta(\tilde{E} - \tilde{E_k}) T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D} \sum_{k=0}^{D-1} \langle k|A|k\rangle\, T_n(\tilde{E_k}) \qquad (40) \\
&= \frac{1}{D} \sum_{k=0}^{D-1} \langle k|AT_n(\tilde{E_k})|k\rangle \\
&= \frac{1}{D} \sum_{k=0}^{D-1} \langle k|AT_n(\tilde{H})|k\rangle \\
&= \frac{1}{D} Tr(AT_n(\tilde{H}))
\end{aligned}$$

Again, the next section will be about a statistical approach to calculate this trace.

The last type of moments are in the time-dependent case, for the function $\rho_A(E, t)$. For the rest of this section discrete time will be used, since the propagator gives time steps, meaning $t_m = m \cdot dt$.

$$\rho_A(E, m \cdot dt) = \frac{1}{D} \sum_{k=0}^{D-1} \langle k_{t=m\cdot dt}|A_{t=m\cdot dt}|k_{t=m\cdot dt}\rangle\, \delta(\lambda - \lambda_k) \qquad (41)$$

Where again $|k_{t=m\cdot dt}\rangle = U(m \cdot dt, 0)\,|k\rangle$ and $A_{t=m\cdot dt}$ is the possibly time-dependent operator.

Firstly the moments at $t = 0$ can be calculated recursively as done in the time independent case. The only difference is an extra index to the states $|\alpha_n\rangle$ representing the time:

$$|\alpha_{n,t=0}\rangle = T_n(\tilde{H}_{t=0})\,|\alpha_{t=0}\rangle \qquad (42)$$

From these states $|\alpha_{n,t=0}\rangle$ the moments for $t = 0$ can be calculated. This goes exactly the same as in the stationary case,

$$\mu_{n,t=0} = \frac{1}{D} Tr(A_{t=0} T_n(\tilde{H_{t=0}})) \qquad (43)$$

So far everything is the same as in the stationary case. Now a time step dt can be taken, which means that the states $|\alpha_n\rangle$ are propagated one step in time:

$$|\alpha_{n,t=dt}\rangle = U(dt,0)\,|\alpha_{n,t=0}\rangle \tag{44}$$

And in general for any amount of time steps:

$$|\alpha_{n,t=m\cdot dt}\rangle = U(m\cdot dt,0)\,|\alpha_{n,t=0}\rangle \tag{45}$$

Note that $|\alpha_{n,t=(m+1)\cdot dt}\rangle$ can be calculated by just propagating $|\alpha_{n,t=m\cdot dt}\rangle$ once. This means that for each time step the states at the previous time can be used. Now the time-dependent moments $\mu_{n,t=m\cdot dt}$ can be derived to a trace form:

$$\begin{aligned}
\mu_{n,t=m\cdot dt} &= \int_{-1}^{1} \tilde{\rho}_A(\tilde{E},t=m\cdot dt)T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \int_{-1}^{1} \frac{1}{D}\sum_{k=0}^{D-1} \langle k_{t=m\cdot dt}|A_{t=m\cdot dt}|k_{t=m\cdot dt}\rangle\,\delta(\tilde{E}-\tilde{E}_k)T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D}\sum_{k=0}^{D-1} \langle k|U(0,m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt,0)|k\rangle \int_{-1}^{1} \delta(\tilde{E}-\tilde{E}_k)T_n(\tilde{E})\ \mathrm{d}\tilde{E} \\
&= \frac{1}{D}\sum_{k=0}^{D-1} \langle k|U(0,m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt,0)T_n(\tilde{E}_k)|k\rangle \\
&= \frac{1}{D}\sum_{k=0}^{D-1} \langle k|U(0,m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt,0)T_n(\tilde{H}_{t=0})|k\rangle \\
&= \frac{1}{D}Tr(U(0,m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt,0)T_n(\tilde{H}_{t=0}))
\end{aligned} \tag{46}$$

In this case $U(0,m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt,0)$ can be seen as a time-dependent operator. After the section about the statistical approximation there will be a section about how to calculate these time-dependent moments efficiently.

3.3 Statistical approximation

The following useful approximation will be examined in this section.

$$\mu_n = Tr(AT_n(\tilde{H})) \approx \frac{1}{R}\sum_{r=1}^{R} \langle r|AT_n(\tilde{H})|r\rangle \tag{47}$$

Here $|r\rangle$ is a randomly chosen state, meaning the trace is approximated using only a random set of states. Also $R \ll D$ resulting in a clear advantage over calculating the entire trace:

$$\frac{1}{D}\sum_{k=1}^{D} \langle k|AT_n(\tilde{H})|k\rangle \tag{48}$$

A random vector can be obtained by taking a set of random variables: $\xi_{ri} \in \mathbb{C}$ and a basis $\{|i\rangle\}$ of dimension D. From which $|r\rangle$ can be constructed:

$$|r\rangle = \sum_{i=0}^{D-1} \xi_{ri} \, |i\rangle \tag{49}$$

The double index i, r of ξ_{ri} simply means that a different ξ is picked for all vectors $|i\rangle$ of the basis and all random vectors $|r\rangle$. The ξ_{ri} can be picked from a distribution which has the following statistical properties, when $\xi_{ri} \in \mathbb{C}$

$$\begin{aligned}
&1) \qquad \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \xi_{ri} \equiv \ll \xi_{ri} \gg = 0 \\
&2) \qquad \ll \xi_{ri}\xi_{r'j} \gg = 0 \\
&3) \qquad \ll \xi^*_{ri}\xi_{r'j} \gg = \delta_{rr'}\delta_{ij}
\end{aligned} \tag{50}$$

When the $\xi_{ri} \in \mathbb{R}$ it is sufficient that the following conditions hold:

$$\begin{aligned}
\ll \xi_{ri} \gg = 0 \\
\ll \xi_{ri}\xi_{r'j} \gg = 0
\end{aligned} \tag{51}$$

In the rest of this section the references 1), 2) and 3) will be used to refer to the general properties. There are many distributions that satisfy these conditions, for example Gaussian or uniform distributions. At the end of this section the choice of distribution will be discussed in some detail. As a first result it can be proven that on average (47) gives the correct result. For an arbitrary Hermitian operator B with matrix elements $B_{ij} = \langle i|B|j\rangle$the following holds:

$$\begin{aligned}
\ll \Theta \gg \equiv &\ll \frac{1}{R} \sum_{r=1}^{R} \langle r|B|r\rangle \gg \\
=&\ll \frac{1}{R} \sum_{r=1}^{R} \sum_{i=0}^{D-1} \xi^*_{ri} \, \langle i|B[\sum_{j=0}^{D-1} \xi_{rj}|j\rangle] \gg \\
=&\ll \frac{1}{R} \sum_{r=1}^{R} \sum_{i=0}^{D-1} \sum_{j=0}^{D-1} \xi^*_{ri}\xi_{rj} \, \langle i|B|j\rangle \gg \\
=& \frac{1}{R} \sum_{r=1}^{R} \sum_{i,j=0}^{D-1} \ll \xi^*_{ri}\xi_{rj} \gg B_{ij} \\
=& \frac{1}{R} \sum_{r=1}^{R} \sum_{i=0}^{D-1} B_{ii} \\
=& \sum_{i=0}^{D-1} B_{ii} \\
=& Tr(B)
\end{aligned} \tag{52}$$

This is a nice result but since $|r\rangle$ is random, the variance of Θ could still be very high. However this is not the case, as will be proven next. The variance of Θ is given by: $(\delta\Theta)^2 = \ll \Theta^2 \gg - \ll \Theta \gg^2$. Since the second term was already derived in (52) only the first term has to be calculated. Before moving on it should be noted that since ξ_{ri} are independent variables:

$$\ll \xi^*_{ri}\xi_{rj}\xi^*_{r'i'}\xi_{r'j'} \gg = \ll \xi^*_{ri}\xi_{rj} \gg \ll \xi^*_{r'i'}\xi_{r'j'} \gg = \delta_{ij}\delta_{i'j'} \qquad r \neq r' \tag{53}$$

When $r = r'$ there is the case that $i = j = i' = j'$ and then the ξ_{ri} are no longer independent variables. In that case:

$$\ll (\xi^*_{ri})^2\xi^2_{ri} \gg = \ll |\xi_{ri}|^4 \gg \tag{54}$$

The last thing needed is a property of the trace:

$$Tr(B^2) = \sum_{i=0}^{D-1}(B^2)_{ii} = \sum_{i,j=0}^{D-1} B_{ij}B_{ji} \tag{55}$$

Which follows immediately from the definition of the matrix product. This is used in the last step of the proof, now for $\ll \Theta^2 \gg$:

$$\ll \Theta^2 \gg := \ll \frac{1}{R}\sum_{r=1}^{R} \langle r|B|r\rangle \, \frac{1}{R}\sum_{r=1}^{R} \langle r|B|r\rangle \gg$$

$$= \frac{1}{R^2} \ll \sum_{r,r'=1}^{R} \sum_{i=0}^{D-1} \xi^*_{ri} \, \langle i|B[\sum_{j=0}^{D-1} \xi_{rj}|j\rangle] \sum_{i'=0}^{D-1} \xi^*_{r'i'} \, \langle i'|B[\sum_{j'=0}^{D-1} \xi_{r'j'}|j'\rangle]$$

$$= \frac{1}{R^2} \ll \sum_{r,r'=1}^{R} \sum_{i,j=0}^{D-1} \xi^*_{ri}\xi_{rj} \, \langle i|B|j\rangle \sum_{i,j=0}^{D-1} \xi^*_{r'i}\xi_{r'j} \, \langle i|B|j\rangle \gg$$

$$= \frac{1}{R^2} \sum_{r,r'=1}^{R} \sum_{i,j,i',j'=0}^{D-1} \ll \xi^*_{ri}\xi_{rj}\xi^*_{r'i'}\xi_{r'j'} \gg B_{ij}B_{i'j'}$$

So far it is just substituting and rearranging,
now the split it made between $r = r'$ and $r \neq r'$

$$= \frac{1}{R^2}(\sum_{\substack{r,r'=1\\ r\neq r'}}^{R} \sum_{i,j,i',j'=0}^{D-1} \ll \xi^*_{ri}\xi_{rj}\xi^*_{r'i'}\xi_{r'j'} \gg B_{ij}B_{i'j'}$$

$$+ \sum_{r=1}^{R} \sum_{i,j,i',j'=0}^{D-1} \ll \xi^*_{ri}\xi_{rj}\xi^*_{ri'}\xi_{rj'} \gg B_{ij}B_{i'j'})$$

$$= \frac{1}{R^2}(\sum_{\substack{r,r'=1\\ r\neq r'}}^{R} \sum_{i,j,i',j'=0}^{D-1} \delta_{ij}\delta_{i'j'} B_{ij}B_{i'j'} + \sum_{r=1}^{R} \sum_{i,j,i',j'=0}^{D-1} \ll \xi^*_{ri}\xi_{rj}\xi^*_{ri'}\xi_{rj'} \gg B_{ij}B_{i'j'})$$

$$= \frac{1}{R^2} R(R-1) \sum_{i,i'=0}^{D-1} B_{ii}B_{i'i'} + \frac{1}{R^2}(\sum_{r=1}^{R} \sum_{\substack{i,j=0\\ i\neq j}}^{D-1} B_{ii}B_{jj}$$

$$+ \sum_{\substack{i,j=0\\ i\neq j}}^{D-1} B_{ij}B_{ji} + \sum_{i=0}^{D-1} \ll |\xi_{ri}|^4 \gg B^2_{ii})$$

Using the above mentioned orthogonality relations for ξ_{ri}
finishing with definitions of the trace

$$= \frac{R-1}{R}(Tr(B))^2 + \frac{1}{R}(\sum_{i,j=0}^{D-1} B_{ii}B_{jj} + \sum_{i,j=0}^{D-1} B_{ij}B_{ji}$$

$$- 2\sum_{i=0}^{D-1} B^2_{ii} + \sum_{i=0}^{D-1} \ll |\xi_{ri}|^4 \gg B^2_{ii})$$

$$= \frac{R-1}{R}(Tr(B))^2 + \frac{1}{R}((Tr(B))^2 + Tr(B^2) + (\ll |\xi_{ri}|^4 \gg -2) \sum_{i=0}^{D-1} B^2_{ii})$$

$$= (Tr(B))^2 + \frac{1}{R}\Big[Tr(B^2) + (\ll |\xi_{ri}|^4 \gg -2) \sum_{j=0}^{D-1} B^2_{jj} \Big] \qquad (56)$$

From which it follows that:

$$(\delta\Theta)^2 = \frac{1}{R}\Big[Tr(B^2) + (\ll |\xi_{ri}|^4 \gg -2)\sum_{j=0}^{D-1} B_{jj}^2\Big] \tag{57}$$

The trace of a matrix is usually $O(D)$, as it is proportional to the dimension of the matrix. Therefore the relative error $\frac{\delta\Theta}{\Theta}$ will usually be $O(\frac{1}{\sqrt{RD}})$. As a result for a larger matrix the amount of random states that need to be evaluated decreases.

Also it can be noted that since the fluctuations depend on $\ll |\xi_{ri}|^4 \gg$, the ξ_{ri} should ideally be chosen such that this term is as close as possible to 1. It can never be less than one, since property 3) still has to be satisfied. When $\xi_{ri} = e^{i\phi}$ with $\phi \in [0, 2\pi]$ a random phase, it follows that $\ll |\xi_{ri}|^4 \gg= 1$. The choice of basis vectors $|i\rangle$ also largely influences the trace terms, making the choice of ξ_{ri} not trivial. In an eigenbasis of B and using the random phase as a choice for ξ_{ri}, $\delta\Theta$ would be 0. In practice the matrix B depends on an operator applied to a Chebyshev polynomial, making it unlikely to work in the eigenbasis. Gaussian distributed ξ_{ri} are also a logical choice since then $\ll |\xi_{ri}|^4 \gg= 2$ and the term $\sum_{j=0}^{D-1} B_{jj}^2$ cancels out in $\delta\Theta$. Now the influence of the basisvectors has dropped out, which is a useful choice too.

3.4 Applying statistical approximation

In the section about calculation of moments it turned out that the three cases of interest for this project could be reduced to a trace form. The moments for the density of states were derived in (37). Now the statistical approximation can be applied,

$$\mu_n = \frac{1}{D}Tr(T_n(\tilde{H})) \approx \frac{1}{RD}\sum_{r=1}^{R} \langle r|T_n(\tilde{H})|r\rangle = \frac{1}{RD}\sum_{r=1}^{R} \langle r|r_n\rangle\,. \tag{58}$$

Where $|r_n\rangle = T_n(\tilde{H})\,|r\rangle$ can be calculated iteratively from $|r\rangle$ using the relation (28). Note that in this case there is symmetry in (25), which can be used to calculate the moments more efficiently than (58). This means that the moments can be calculated by (31) as follows:

$$\mu_0 = \frac{1}{RD}\sum_{r=1}^{R} \langle r|r\rangle \qquad \mu_1 = \frac{1}{RD}\sum_{r=1}^{R} \langle r|r_1\rangle \tag{59}$$

$$\mu_{2n} = \frac{1}{RD}\sum_{r=1}^{R} 2\,\langle r_n|r_n\rangle - \mu_0 \tag{60}$$

$$\mu_{2n+1} = \frac{1}{RD}\sum_{r=1}^{R} 2\,\langle r_{n+1}|r_n\rangle - \mu_1 \tag{61}$$

The second case was for the expectation values of an operator A, the moments were derived to a trace form in (40):

$$\mu_n = \frac{1}{D} Tr(AT_n(\tilde{H})) \approx \frac{1}{RD} \sum_{r=1}^{R} \langle r|AT_n(\tilde{H})|r\rangle = \frac{1}{RD} \sum_{r=1}^{R} \langle r|A|r_n\rangle \tag{62}$$

Note that the operator A breaks the symmetry, so the symmetric relations can no longer be used.

The last case was for the time-dependent moments, for which the trace form was derived in (46).

$$\begin{aligned}
\mu_{n,t=m\cdot dt} &= \frac{1}{D} Tr\left[U(0, m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt, 0)T_n(\tilde{H}_{t=0})\right] \\
&\approx \frac{1}{RD} \sum_{r=1}^{R} \langle r|U(0, m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt, 0)T_n(\tilde{H})|r\rangle \\
&= \frac{1}{RD} \sum_{r=1}^{R} \langle r|U(0, m\cdot dt)A_{t=m\cdot dt}U(m\cdot dt, 0)|r_n\rangle \\
&= \frac{1}{RD} \sum_{r=1}^{R} \langle r_{t=m\cdot dt}|A_{t=m\cdot dt}|r_{n,t=m\cdot dt}\rangle
\end{aligned} \tag{63}$$

A way to implement this is to first calculate the states $|r_{n,t=0}\rangle$ iteratively as in (62) and calculate the moments $\mu_{n,t=0}$ using (63). Then all the states can be propagated in time one step to get $|r_{n,t=dt}\rangle$ and used to compute $\mu_{n,t=dt}$. This process can be repeated to get all the moments $\mu_{n,t=m\cdot dt}$.

3.5 Kernel improvement

In practice a finite number of moments N can be calculated so the infinite series becomes finite. Then the expanded function $f(x)$ is approximated by a finite order Chebyshev polynomial,

$$f(x) \approx \frac{1}{\pi\sqrt{1-x^2}} \left(\mu_0 + 2\sum_{n=1}^{N-1} \mu_n T_n(x)\right) \tag{64}$$

The error near the points where $f(x)$ is not differentiable is not bounded and can fluctuate a lot. This phenomenon is known as Gibbs oscillations. It turns out however that this expansion can be improved by modifying the moments μ_n. This can be done by multiplying the moments by a kernel g_n, after which the modified moments will be $g_n\mu_n$.

$$f_{KPM}(x) = \frac{1}{\pi\sqrt{1-x^2}} \left(g_0\mu_0 + 2\sum_{n=1}^{N-1} g_n\mu_n T_n(x)\right) \tag{65}$$

The problem of finding an optimal kernel g_n has been studied for many years. In the review of KPM [1] it is shown that the most optimal Kernel is almost always the Jackson

Kernel. In the paper this Kernel has been derived and is given by:

$$g_n^J = \frac{(N-n+1)\cos(\frac{\pi n}{N+1}) + \sin(\frac{\pi n}{N+1})\cot(\frac{\pi n}{N+1})}{N+1} \tag{66}$$

In the paper it is also shown that the error $||f(x) - f_{KPM}(x)||_\infty$ using this kernel is of the order of $O(1/N)$. This means that for $N \to \infty$ this approximation converges uniformly.

In the future the kernel improved moments will be noted with as $\bar{\mu}_n = g_n^J \mu_n$.

3.6 Cosine transform and rescaling

After the (modified) moments have been calculated via one of the earlier discussed methods, the last step is to get back to the initial function. First of all x has to be made discrete by choosing some points x_k in which the expansion will be evaluated. The choice of those x_k is important because it will determine how stable the method is. Now $\tilde{f}$ can be determined in all the points x_k. This still is a rescaled version of the function f. This could be done by summing up (106) for all points, but that would not be the most efficient. In discrete form (106) now reads:

$$\tilde{f}(x_k) = \frac{1}{\pi\sqrt{1-x_k^2}} \left(\bar{\mu}_0 + 2\sum_{n=1}^{N-1} \bar{\mu}_n T_n(x_k) \right) \tag{67}$$

The $\bar{\mu}_n$ are the modified moments discussed in the last section. The data points x_k will be chosen as the Chebyshev nodes, which are often useful in numerical analysis. Later on it will be shown that with these data points (67) can be calculated with a discrete cosine transform. This choice also improves stability, especially near the bounds of $[-1, 1]$. The Chebyshev nodes x_k are defined as:

$$x_k = \cos(\frac{\pi(k+1/2)}{K}) \qquad \text{for } k = 0, 1, ..., K-1 \tag{68}$$

Where K is the number of data points. Using the definition of the Chebyshev polynomials (67) can be written as:

$$\begin{aligned} \gamma_k = \pi\sqrt{1-x_k^2}\tilde{f}(x_k) &= \bar{\mu}_0 + 2\sum_{n=1}^{N} \bar{\mu}_n T_n(x_k) \\ &= \bar{\mu}_0 + 2\sum_{n=1}^{N} \bar{\mu} \cos(n \arccos(x_k)) \\ &= \bar{\mu}_0 + 2\sum_{n=1}^{N} \bar{\mu}_n \cos(n \arccos(\cos(\frac{\pi(k+1/2)}{K}))) \\ &= \bar{\mu}_0 + 2\sum_{n=1}^{N} \bar{\mu}_n \cos(\frac{n\pi(k+1/2)}{K}) \end{aligned} \tag{69}$$

Now γ_k is the definition of a discrete cosine transform of the moments $\bar{\mu}_n$, which can be calculated efficiently and is a clear improvement over summing everything up.

To get the unscaled expanded function a final rescaling must be applied.

3.7 Expanded functions to physical quantity

The last sections have shown how to expand the functions $\rho(E)$, $\rho_A(E)$ and $\rho_A(E,t)$ using Chebyshev polynomials. The functions $\rho_A(E)$ and $\rho_A(E,t)$ can be used to calculate $\langle A \rangle$ and $\langle A(t) \rangle$ using (2) and (7).

To get the expectation value of A the function $\rho_A(y_k)$ can be integrated numerically, for example by the trapezoid rule. Since the points y_k are not uniform the trapezoid rule is given by:

$$\int_a^b f(y)\ \mathrm{d}y = \frac{1}{2}\sum_{k=1}^{K}(y_{k+1}-y_k)(f(y_{k-1})-f(y_k)) \tag{70}$$

Now the expectation value of A is given by:

$$\begin{aligned}\langle A \rangle &= \int_a^b \rho_A(E)n(E)\ \ \mathrm{d}E \\ &= \frac{1}{2}\sum_{k=1}^{K}(y_{k+1}-y_k)(\rho_A(y_{k-1})n(y_{k-1})-\rho_A(y_k)n(y_k))\end{aligned} \tag{71}$$

For the time-dependent case numerical integration can also be used as in (70) to get the time-dependent expectation value of A:

$$\begin{aligned}\langle A_{m\cdot dt} \rangle &= \int_a^b \rho_A(E, m\cdot dt)n(E)\ \ \mathrm{d}E \\ &= \frac{1}{2}\sum_{k=1}^{K}(y_{k+1}-y_k)(\rho_{A,t=m\cdot dt}(y_{k-1})n(y_{k-1})-\rho_{A,t=m\cdot dt}(y_k)n(y_k))\end{aligned} \tag{72}$$

3.8 Approximation of the time propagator

As briefly mentioned before, the calculation of the exponent of a matrix is hard, therefore the time propagator must be approximated in a good way. The operator $\exp(-iHt)$ is unitary, so $\exp(-iHt)^\dagger\exp(iHt) = I$ with I the identity matrix. This can be proven using the fact that for square matrices of the same dimension A and B: whenever A and B commute, meaning $AB = BA$, it follows that $\exp(A+B) = \exp(A)\exp(B)$.

$$\begin{aligned}\exp(-iHt)^\dagger\exp(-iHt) &= \exp(iHt)\exp(-iHt) \\ &= \exp(i(H-H)t) \\ &= \exp(0) \\ &= I\end{aligned} \tag{73}$$

In practice calculating an entire matrix exponential is not possible as in (9), so an approximation can be made. It is important that this approximated operator remains unitary, since unitary operators preserve the norm of a vector. For a unitary operator U and a vector v: $||v|| = ||Uv||$. Therefore if the approximated operator is not unitary the wave function will grow exponentially when applying the operator. The following unitary approximation can be made using the identity matrix I:

$$\exp(-iH(t)dt) \approx (I - \frac{idt}{2}H(t))(I + \frac{idt}{2}H(t))^{-1} \tag{74}$$

This corresponds to a first order Pade approximation. In general this operator is still expensive to calculate, since it requires matrix inversion. However this simplifies when applying this operator to a vector. The first part of the operator is trivial, whereas for the second part of the operator a property of the inverse can be used. Then the outcome x of applying the second part of the operator to a vector v is $x = (1 + \frac{idt}{2}H(t))^{-1}v$ and can be found by solving $v = (1 + \frac{idt}{2}H(t))x$. This is a linear system and for sparse matrices there are fast solvers available for such systems of equations.

4 Numerical experiments

4.1 General considerations

In the following sections numerical results of the Kernel Polynomial Method will be shown. One very important Python package for this project is the Kwant package. For this project the main use of Kwant is to get the Hamiltonian corresponding to a tight binding system.

For small system sizes it is possible to calculate all the eigenstates $|k\rangle$ of the Hamiltonian and obtain the exact result. This can be used as a comparison between the KPM approximation and the exact calculation of the trace.

The two important parameters to chose when using KPM are the amount of moments N and the amount of random vectors R. The amount of moments directly defines the amount of points in energy K that can be taken. This will be the main subject of the next section. How many random vectors are needed depends on the situation, which will also be shown in the next sections.

4.2 Density of states

For a finite system the density of states is given by delta peaks at the eigenvalues of the Hamiltonian.

$$\rho(E) = \frac{1}{D}\sum_{k=0}^{D-1}\delta(E - E_k) \tag{75}$$

However since only a finite number of moments N is taken, the expansion of the density of states is a polynomial of order $N-1$. This means that if there are more eigenvalues in a certain range, the expanded function will be larger in magnitude there. As the amount of moments is increased the delta peaks become narrower and the individual eigenvalues can be seen.

To demonstrate this, a tight binding Hamiltonian is obtained using Kwant for a one dimensional chain of atoms. The tight binding Hamiltonian is given by:

$$H = \sum_i \left[V_i c_i^\dagger c_i - t(c_{i+1}^\dagger c_i + c_i^\dagger c_{i+1})\right] \tag{76}$$

Here c_i is an annihilation operator, which annihilates an electron at site i and $c_i^\dagger$ is a creation operator, creating an electron at site i. These operators are widely used in quantum mechanics to decrease (annihilate) or increase (create) the number of electrons in a state by one. V_i is the on-site potential of the atom at site i and t is the hopping, which is assumed to be the same for all links. As can be seen in this expression only nearest neighbor interactions are included. For this example also the potential is constant for all sites so $V_i = V$.

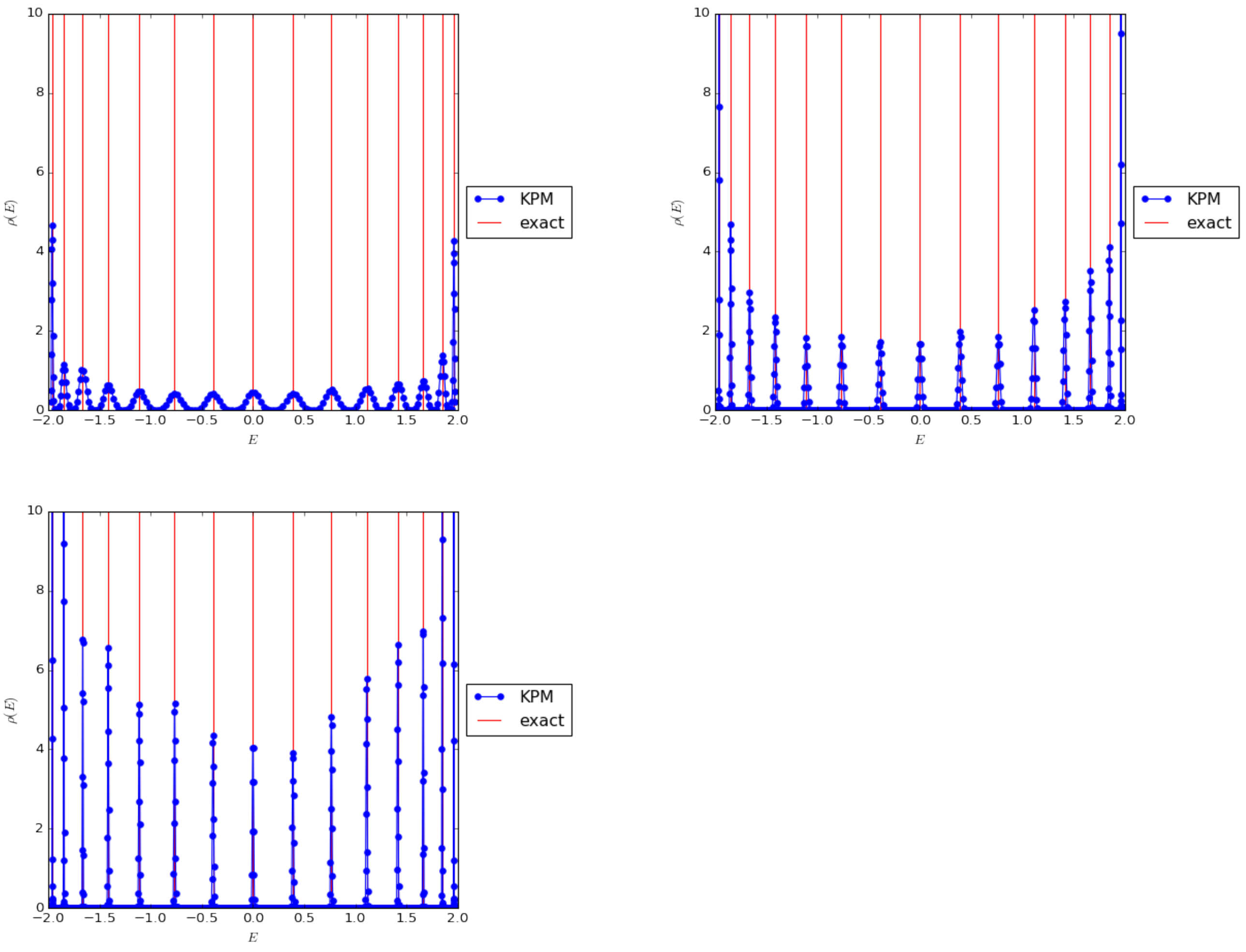


Figure 1: Density of states of a one dimensional chain, for $N = 100$, $N = 400$ and $N = 1000$

The length of the chain L is taken small, so that the individual delta peaks are visible. Now for $L = 15$ and $R = 10$ random vectors the density of states can be expanded for different amounts of moments N.

For a larger system there are more delta peaks, which overlap even when using a large amount of moments. This means that as in the case of an infinite system, this function tells us about the distribution of eigenvalues. When increasing the amount of moments even more, the individual delta peaks become visible for any finite system.

Graphene is a two dimensional sheet of carbon atoms configured in a honeycomb lattice. A lot of research is being done on graphene, for example looking for cheaper ways to produce it, making it feasible for applications. The tight binding Hamiltonian of graphene is for example given in [6].

$$H = -t \sum_{\langle i|j \rangle} (a_i^\dagger b_j + b_j^\dagger a_i) \tag{77}$$

Here a, b are annihilation operators for the two sublattices of graphene, $a^\dagger, b^\dagger$ are creation operators for the sublattices. The sum over $\langle i|j \rangle$ means summation over the sites i, j such that i and j are nearest neighbors.

This Hamiltonian can also be obtained from Kwant. In this case a circular flake of graphene is used with a radius of 200, which gives 290134 by 290134 as the dimension of H. For $R = 10$ and $N = 400$ this results in the following density of states:

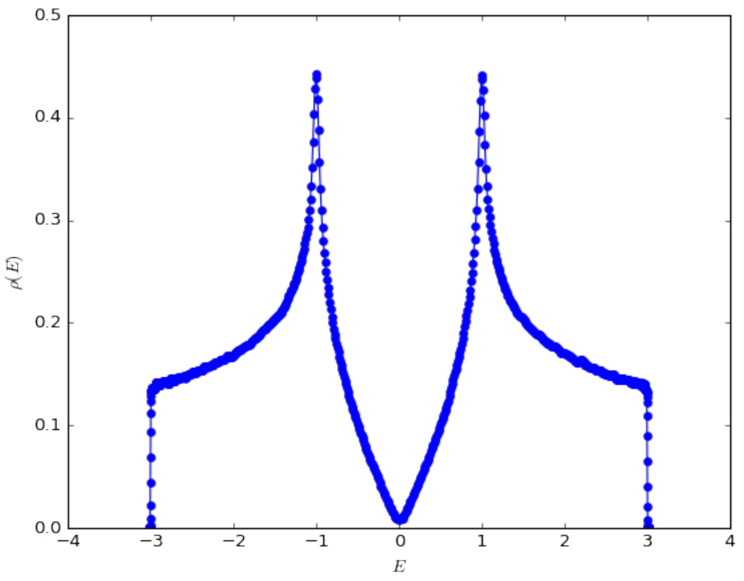
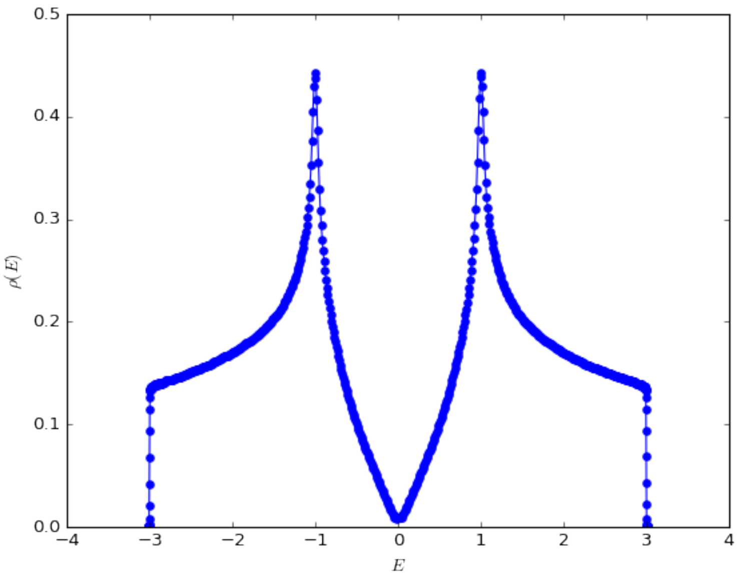


Figure 2: Density of states for circular graphene

Graphene is a zero band gap semiconductor, which can be seen in the plot at $E = 0$. The density of states is 0 at $E = 0$ (but not in an interval around it). It can also be seen that for this finite system the energy bounds are around $E_{min} = -3$ and $E_{max} = 3$. Note that the amount of random states is only 10 whereas if we wanted the trace exactly we would need 290134 states. This system is too large to calculate the exact eigenstates of the Hamiltonian, so that comparison cannot be made here. One test is to see what happens when increasing the amount of random vectors from 10 to 100 and 1000. The amount of moments can also be increased to check if there are any fluctuations in the density of states on a smaller energy scale.



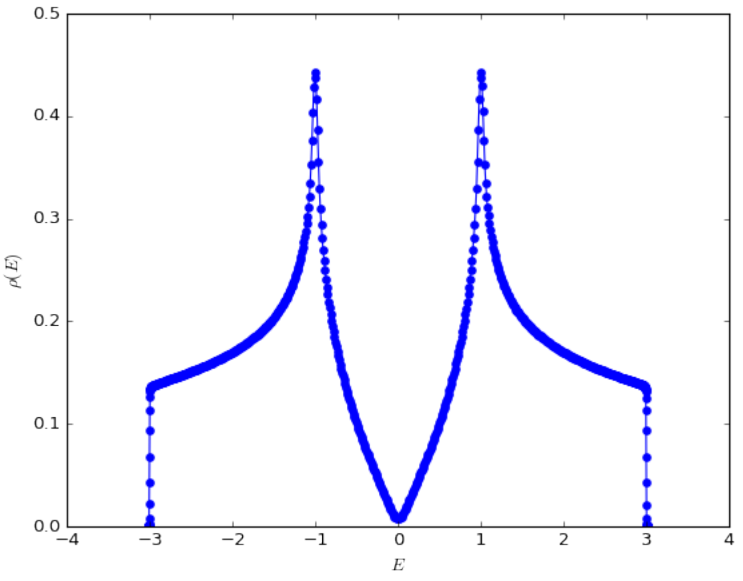
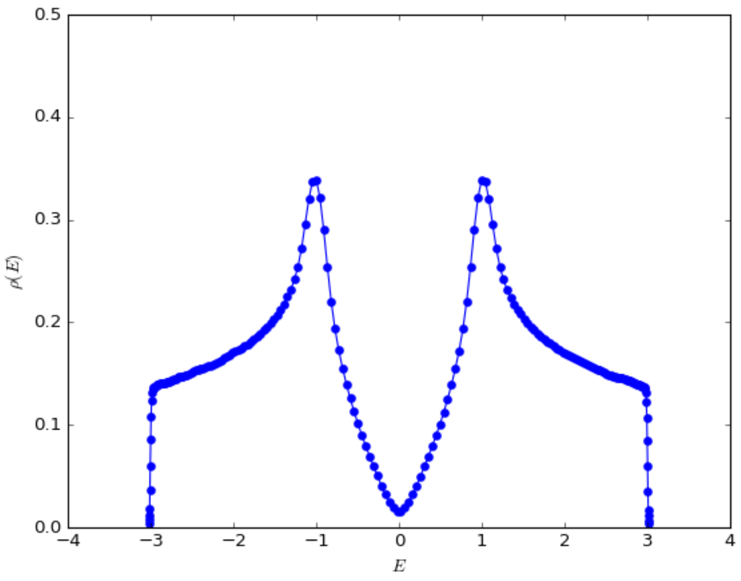


Figure 3: Density of states of circular graphene when increasing the number of random vectors, $R = 100$ and $R = 1000$

In the step from $R = 10$ to $R = 100$ a difference in the plots can be seen, indicating increased convergence. However this change is so small that the case $R = 10$ already gives very good convergence. The plot for $R = 1000$ does not seem any different than the plot for $R = 100$, meaning that further increasing the amount of random vectors changes nothing.

Now the amount of random vectors is kept constant, but the amount of moments is varied.



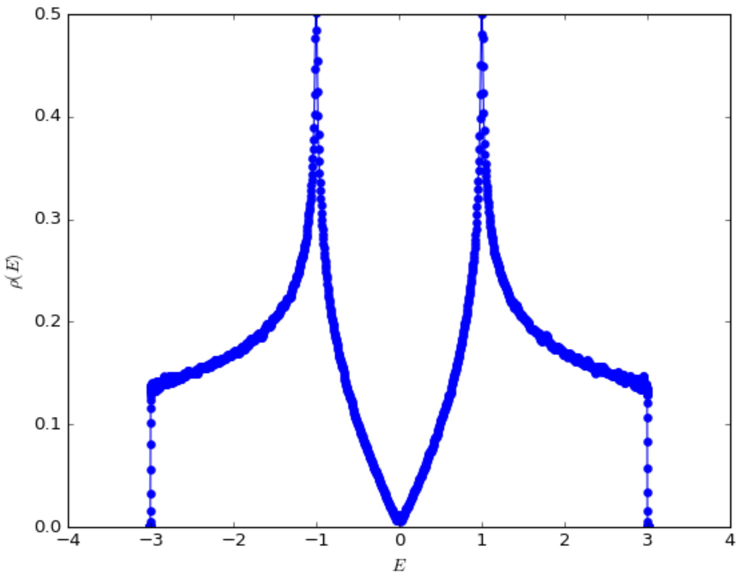


Figure 4: Density of states of circular graphene for different amounts of moments, $N = 100$ and $N = 1000$

As can be seen in the plots the different amount of moments give different results. There will not be any convergence as in the case of increased amount of random vectors. This means that the fluctuations in the denstiy of states of a certain system can only be seen up to the resolution defined by the amount of moments.

Now a magnetic field can be added to the system, perpendicular to the plane of the graphene. This magnetic field enters the Hamiltonian as a phase in the hoppings. Compared to the Hamiltonian without magnetic field (77) the hopping t is no longer constant. This is known as a Peierls substitution, which was introduced in [7].

$$t_{ij} = t \exp\left(i\, \frac{e}{\hbar} B \frac{(y_i + y_j)}{2} (x_i - x_j) \right) \tag{78}$$

In this case x_i and y_i are the spatial coordinates (x, y) of site i and B is the magnitude of the magnetic field. Now the Hamiltonian is given by:

$$H = -\sum_{\langle i|j\rangle} (t_{ij} a_i^\dagger b_j + t_{ij}^\dagger b_j^\dagger a_i) \tag{79}$$

For $R = 10$ and $N = 400$ the following density of states can be obtained:

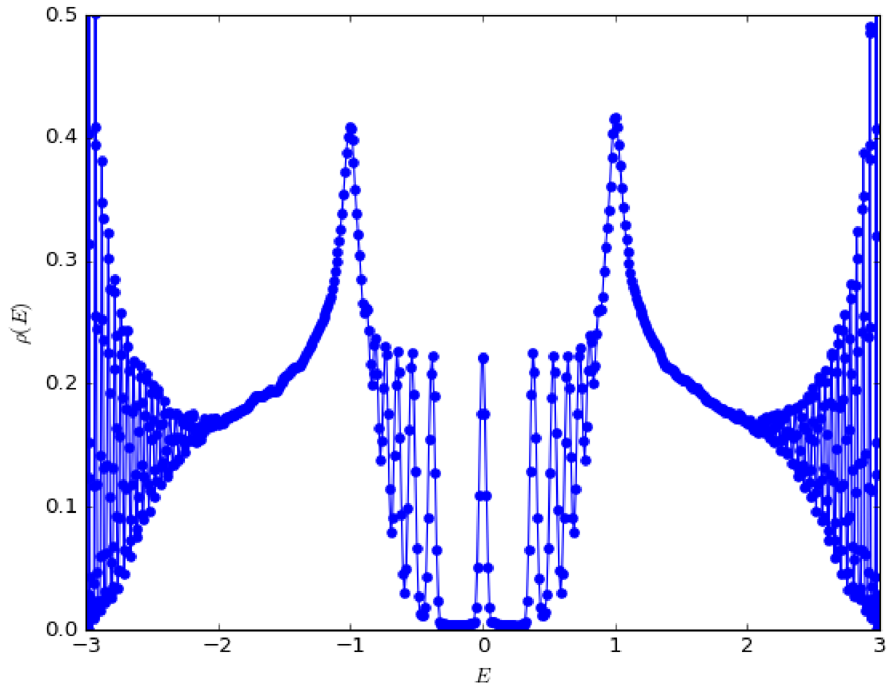
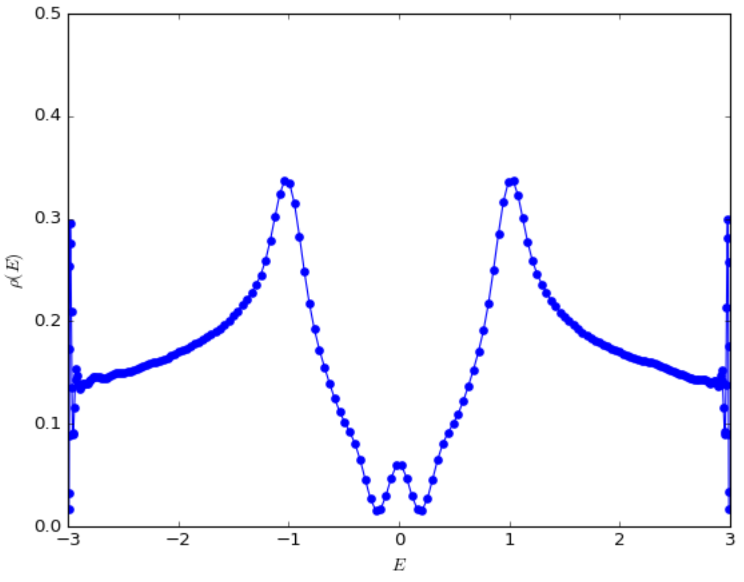


Figure 5: Density of states of circular graphene with magnetic field

Compared to the density of states without the magnetic field there are two regions that differ. Around $E = 0$ and $E = 3$ fast oscillations can be seen in the density of states, which is known as Landau quantisation. Now the amount of moments can be varied to see how this effects the density of states.



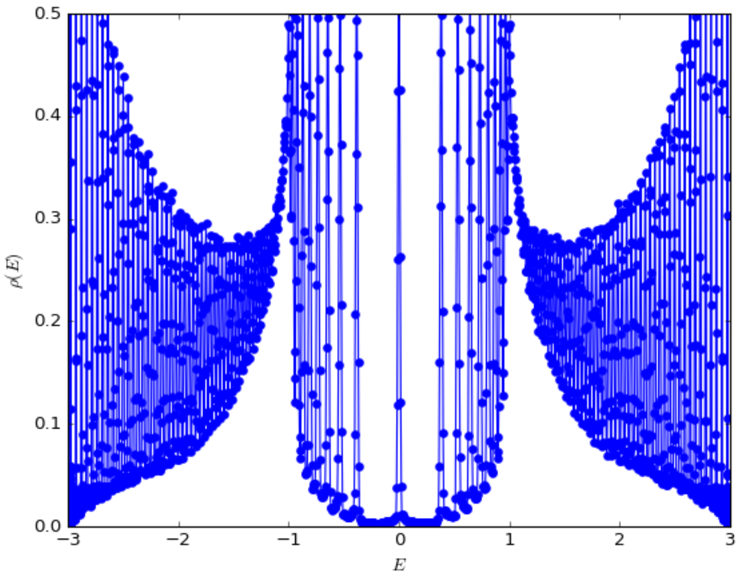


Figure 6: Density of states of circular graphene with magnetic field for different amounts of moments, $N = 100$ and $N = 1000$, $R = 10$

In the case of 100 moments the energy resolution is not small enough to detect the discrete energy levels. For this reason the density of states looks fairly similar to the case without magnetic field. In the case of 1000 moments more oscillations are visible, especially on the energy intervals $[-2, 1]$ and $[1, 2]$.

4.3 Expectation values

The Hamiltonian can be used as operator, $A = H$. In that case the expanded function becomes:

$$\rho_A(E) = \frac{1}{D}\sum_{k=0}^{D-1} \langle k|H|k\rangle\, \delta(E - E_k) = \frac{1}{D}\sum_{k=0}^{D-1} E_k \delta(E - E_k) \tag{80}$$

This function is just $E\rho(E)$ with $\rho(E)$ the density of states shown in the last section. The KPM result of using H as operator and multiplying the density of states with energy can now be compared. The system is again circular graphene, now with a radius of 100, $N = 100$, $R = 30$.

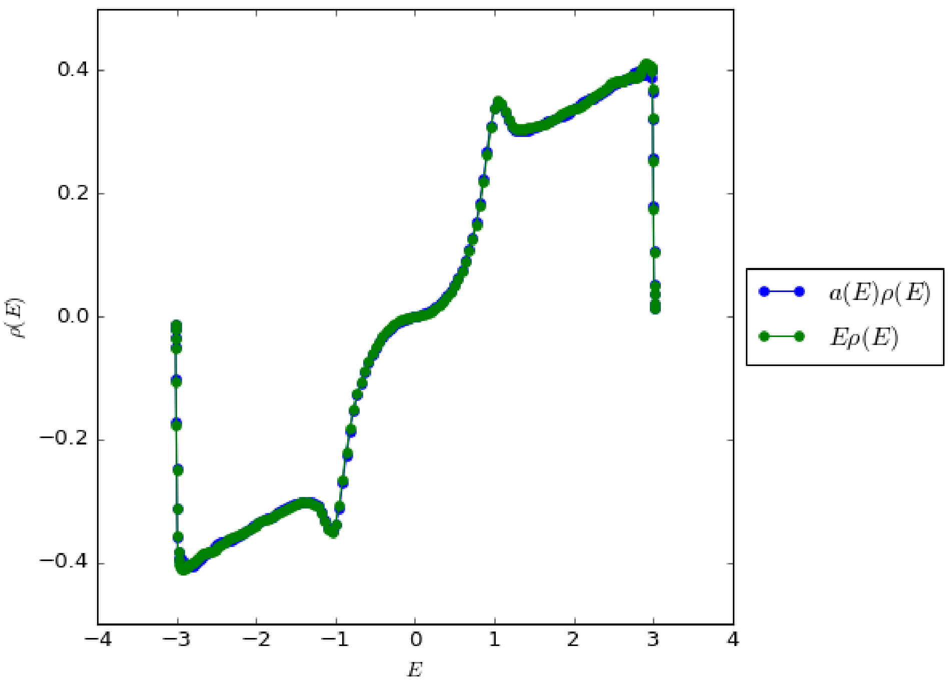


Figure 7: Comparision between the density of states multiplied by energy and the expectation value as a function of energy of the Hamiltonian

As can be seen in the plot the two results are identical, this shows that both methods are valid to calculate the expectation value of the Hamiltonian.

4.4 Time-dependent expectation values

For a time-dependent Hamiltonian again a one dimensional chain is examined. To add time dependence to this system an electric field is added,

$$H(t) = H + xA\sin(\omega t) \tag{81}$$

With A the amplitude of the electric field and ω the frequency. In this case x is a diagonal matrix which indicates the position of each site. This is a general way to add an electric field to a Hamiltonian and can be used for any Hamiltonian obtained from Kwant. For a one dimensional chain this type of electric field can be seen as a bunch of harmonic oscillators with different amplitudes but the same frequency. For example the site at position 1 oscillates with amplitude 1, the site at position 2 oscillates with amplitude 2 etc.

Combining this with the previously mentioned tight binding Hamiltonian for a 1D chain (76) gives:

$$H(t) = \sum_i \left[V_{i,t}c_i^\dagger c_i - t(c_{i+1}^\dagger c_i + c_i^\dagger c_{i+1})\right] \tag{82}$$

Where $V_{i,t} = iA\sin(\omega t)$ and t is a constant because there is no magnetic field.

In this case the total energy of the system is of interest, which is just $\langle H(t)\rangle$ as defined in (7). If the chain is of length 100 the resulting Hamiltonian is small enough to calculate all the eigenstates. Now the total energy can be plotted as a function of time using both KPM and the exact diagonalization.

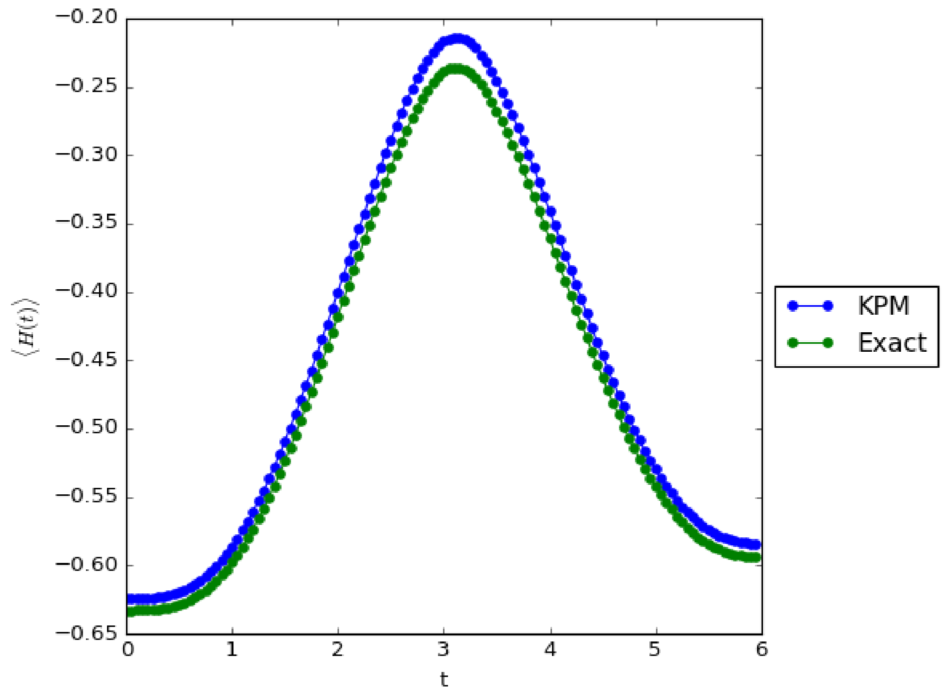


Figure 8: Comparison between exact diagonalization and KPM using 100 random vectors and 100 moments for a one dimensional chain of length 100.The electric field has amplitude 1 and frequency 1.

Note that in this example the KPM approximation is not quite the same as the exact diagonalization. The system is small, which is not in favour of KPM. If a better convergence is needed, more random vectors can be taken. This shows that the amount of random vectors needed depends on the circumstances, as only 10 random vectors were needed for good convergence for the density of states.

The physical interpretation of this plot is of less importance, but the energy of the system starts to oscillate with the frequency of the electric field.

Now the length of the chain can be increased from 100 to 1000, when keeping the other parameters the same.

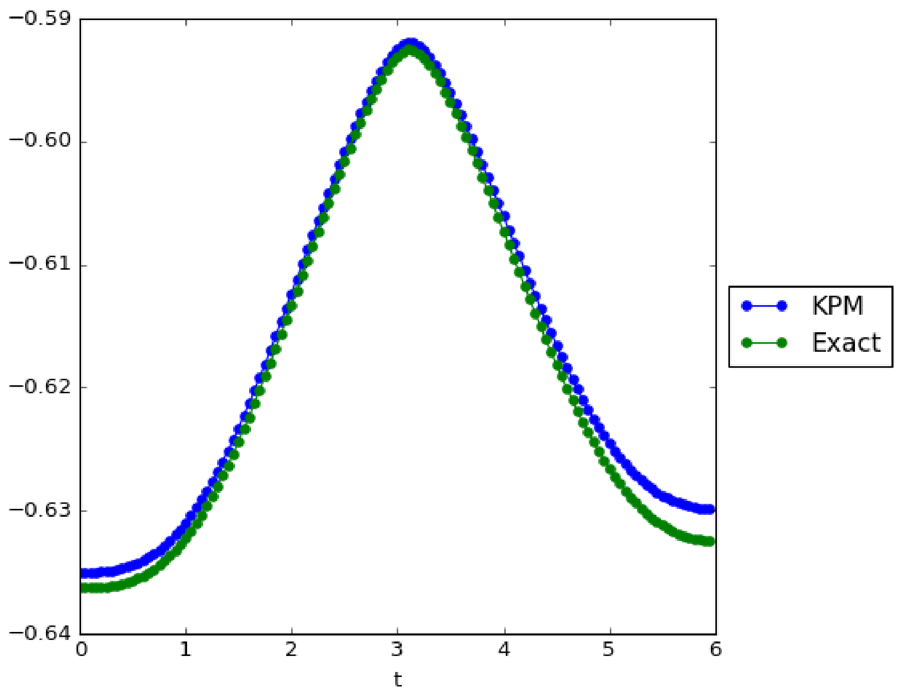


Figure 9: Comparison between exact diagonalization and KPM using 100 random vectors and 100 moments for a one dimensional chain of length 1000.The electric field has amplitude 1 and frequency 1.

This result may not seem better than for $L = 100$ but note that the scale on the vertical axis is a lot smaller in this case. This means that the relative error has decreased by a lot with the increased length of the chain.

5 Conclusion, discussion and future developments

The main goal of the project was to use the Kernel Polynomial Method to calculate expectation values of physical operators and add time-dependence. In order to do that, a mathematical derivation of the KPM was needed, which is the first part of this research. Implementing these mathematical results in Python and running simulations on tight binding systems was the other part. The first part was definitely not groundbreaking because it has been done before, but still essential to understand the KPM and be able to implement it. The added time-dependence has not been seen before in the previous paper on the KPM.

This section will therefore focus on the numerical part of this paper, since the main new results and future developments are there.

The results for the density of states were good and even surprisingly accurate since in most cases only 10-50 random vectors were needed to get a very accurate plot. To be able to plot the density of states for a finite system the delta peaks must be somewhat smoothed, which is automatically done since the KPM approximates it as a finite polynomial. Adding the magnetic field to the system showed Landau quantization, making the energy levels discrete.

For the expectation values the Hamiltonian has been considered as an operator. In this case the density of states times energy is the same as the expectation value of the Hamiltonian, this has also been shown using the KPM. When an electric field was added to the system the total energy of the system started to oscillate in time.

In this case comparing the expectations values calculated by the exact diagonalization of the Hamiltonian to the KPM approximation did not exactly converge. This indicates that to calculate expectation values often a lot more random vectors are needed than to calculate the density of states. This made getting good results especially for large systems hard. For large systems the comparison between exact diagonalization and KPM is not possible anymore, since finding all the eigenstates becomes impossible.

Some tests have also been done to implement the current density operator, of which the expectation values indicate the current going through the different links in the tight binding system. For both the Hamiltonian and the current density operator often thousands of random vectors are needed, which is still good compared to the system size.

In future research the expectation values of other operators could be calculated using the KPM. Since the KPM relies on statistical convergence, experiments with large systems and many random vectors can be done. In this case the question is how the KPM performs compared to other available methods to calculate expectation values.

The implementation of the time propagator is logical, but still the most intensive part of the algorithm; other implementations could be tried. For this part a linear system has to be solved, a standard iterative solver has been used in this project. Since the time propagator is close to identity for small time steps this seems to be a good choice, but there are other possibilities.

6 Apendices

6.1 Appendix A: Chebyshev expansion

Two types of Chebyshev polynomials will be used in this project. Chebyshev polynomials of the first kind are the unique polynomials T_n satisfying
$T_n(\cos\theta) = \cos(n\theta)$.
The second kind U_n satisfy the condition:
$U_n(\cos(\theta)) = \frac{\sin((n+1)\theta)}{\sin\theta}$.
Later it will be shown that these T_n and U_n are indeed polynomials. The next step is to show that T_n and U_n are orthogonal under the following internal product:

$$\langle f|g\rangle = \int_{-1}^{1} w(x)f(x)g(x)\ \mathrm{d}x \tag{83}$$

Here a positive weight function $w(x)$ and two integrable functions $f, g : [a, b] \to \mathbb{R}$ are required. For T_n the weight function is $w(x) = (\pi\sqrt{1-x^2})^{-1}$ and for U_n it is $w(x) = \pi\sqrt{1-x^2}$. When substituting these expressions for $w(x)$ into (83) it follows that:

$$\langle f|g\rangle_1 = \int_{-1}^{1} \frac{f(x)g(x)}{(\pi\sqrt{1-x^2})}\ \mathrm{d}x \tag{84}$$

$$\langle f|g\rangle_2 = \int_{-1}^{1} \pi\sqrt{1-x^2}f(x)g(x)\ \mathrm{d}x \tag{85}$$

Now the following orthogonality relations hold:

$$\langle T_n|T_m\rangle_1 = \frac{(1+\delta_{n,0})}{2}\delta_{n,m} \tag{86}$$

$$\langle U_n|U_m\rangle_2 = \frac{\pi^2}{2}\delta_{n,m} \tag{87}$$

Where $\delta_{n,m}$ is the Kronecker delta, which has the value 1 if $n = m$ and 0 when $n \neq m$. First (86) will be proven:

$$\begin{aligned}
\langle T_n|T_m\rangle_1 &= \int_{-1}^{1} \frac{T_n(x)T_m(x)}{(\pi\sqrt{1-x^2})}\ \mathrm{d}x \\
&= \int_{\pi}^{0} \frac{T_n(\cos(\theta))T_m(\cos(\theta))}{(\pi\sqrt{1-cos^2\theta})}(-sin(\theta)\ \mathrm{d}\theta) \\
&= \frac{1}{\pi}\int_{0}^{\pi} \cos(n\theta)\cos(m\theta)\ \mathrm{d}\theta \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} \cos(n\theta)\cos(m\theta)\ \mathrm{d}\theta \\
&= \begin{cases} 0 & \text{if } n \neq m \\ 1 & \text{if } n = m = 0 \\ \int_{-\pi}^{\pi} \frac{(1+\cos(2n\theta))}{2}\ \mathrm{d}\theta & \text{if } n = m \neq 0 \end{cases} \\
&= \begin{cases} 0 & \text{if } n \neq m \\ 1 & \text{if } n = m = 0 \\ \frac{1}{2} & \text{if } n = m \neq 0 \end{cases} \\
&= \frac{\delta_{n,0}+1}{2}\delta_{n,m}
\end{aligned} \tag{88}$$

Orthogonality of $\cos(n\theta)$ on the interval $[-\pi, \pi]$ was used and also that cosine is an even function. Now for equation (87):

$$\begin{aligned}
\langle U_n|U_m\rangle_2 &= \int_{-1}^{1} U_n(x)U_m(x)(\pi\sqrt{1-x^2})\ \mathrm{d}x \\
&= \int_{\pi}^{0} U_n(\cos(\theta))U_m(\cos(\theta))(\pi\sqrt{1-\cos^2\theta})(-\sin(\theta)\ \mathrm{d}\theta) \\
&= \pi\int_{0}^{\pi} \frac{\sin((n+1)\theta)}{\sin(\theta)}\frac{\sin((m+1)\theta)}{\sin(\theta)}(\sin^2(\theta)\ \mathrm{d}\theta \\
&= \pi\int_{0}^{\pi} \sin((n+1)\theta)\sin((m+1)\theta)\ \mathrm{d}\theta \\
&= \frac{\pi}{2}\int_{-\pi}^{\pi} \sin((n+1)\theta)\sin((m+1)\theta)\ \mathrm{d}\theta \\
&= \begin{cases} 0 & \text{if } n \neq m \\ \int_{-\pi}^{\pi} \frac{1-\cos(2(n+1)\theta))}{2}\ \mathrm{d}\theta & \text{if } n = m \end{cases} \\
&= \begin{cases} 0 & \text{if } n \neq m \\ \frac{\pi^2}{2} & \text{if } n = m \end{cases} \\
&= \frac{\pi^2}{2}\delta_{n,m}
\end{aligned} \tag{89}$$

It was used that sines are odd functions and that the product of two odd functions is even. And also orthogonality of $\sin((n+1)\theta)$ on the interval $[-\pi, \pi]$.

These Chebyshev polynomials can also be written in explicit form, using the identities given at the start of this section:

$$T_n(x) = \cos(n \arccos(x)) \tag{90}$$

$$U_n(x) = \frac{\sin((n+1)\arccos(x))}{\sin(\arccos(x))} \tag{91}$$

From this definition it follows that both types of Chebyshev polynomials are only defined for $x \in [-1, 1]$. From these equations the following can be obtained:

$$T_0(x) = 1 \qquad T_{-1}(x) = T_1(x) = x \tag{92}$$

$$U_0(x) = 1 \qquad U_{-1}(x) = 0 \tag{93}$$

And those can be used to prove the following recursion relations:

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x) \tag{94}$$

$$U_{m+1}(x) = 2xU_m(x) - U_{m-1}(x) \tag{95}$$

Using the substitution $x = cos(\theta)$:

$$\begin{aligned} T_{m+1}(x) &= \cos((m+1)\arccos(\cos(\theta))) \\ &= \cos((m+1)\theta) \\ &= \cos(m\theta)\cos(\theta) - \sin(m\theta)\sin(\theta) \\ &= 2\cos(\theta)\cos(m\theta) - (\cos(m\theta)\cos(\theta) + \sin(m\theta)\sin(\theta)) \\ &= 2\cos(\theta)\cos(m\theta) - \cos((m-1)\theta) \\ &= 2x\cos(m\arccos(x)) - \cos((m-1)\arccos(x)) \\ &= 2xT_m(x) - T_{m-1}(x) \end{aligned} \tag{96}$$

And for $U_m(x)$ again using $x = \cos(\theta)$:

$$\begin{aligned} U_{m+1}(x) &= \frac{\sin((m+2)\arccos(x))}{\sin(\arccos(x))} \\ &= \frac{\sin((m+2)\theta)}{\sin(\theta)} \\ &= \frac{\sin(m\theta)\cos(2\theta) + \cos(m\theta)\sin(2\theta)}{\sin(\theta)} \\ &= \frac{\sin(m\theta)(\cos^2(\theta) - 1) + \cos(m\theta)(2\sin(\theta)\cos(\theta))}{\sin(\theta)} \\ &= \frac{-\sin(m\theta) + 2\cos^2(\theta)\sin(m\theta) + 2\sin(\theta)\cos(\theta)\cos(m\theta)}{\sin(\theta)} \\ &= \frac{2\cos(\theta)(\sin(m\theta)\cos(\theta) + \cos(m\theta)\sin(\theta))}{\sin(\theta)} - \frac{\sin(m\theta)}{\sin(\theta)} \\ &= \frac{2\cos(\theta)(\sin((m+1)\theta)}{\sin(\theta)} - \frac{\sin(m\theta)}{\sin(\theta)} \\ &= \frac{2x\sin((m+1)\arccos(x)}{\sin(\arccos(x))} - \frac{\sin(m\arccos(x))}{\sin(\arccos(x))} \\ &= 2xU_m(x) - U_{m-1}(x) \end{aligned} \tag{97}$$

This also shows that both U_n and T_n are polynomials. These polynomials are very useful in numeric analysis in general and will be used a lot in this project. Two other useful relations for these Chebyshev polynomials are:

$$2T_m(x)T_n(x) = T_{m+n}(x) + T_{m-n}(x) \tag{98}$$
$$2(x^2-1)U_{m-1}(x)U_{n-1}(x) = T_{m+n}(x) - T_{m-n}(x) \tag{99}$$

Using the substitution $x = \cos(\theta)$.

$$\begin{aligned}
T_{m+n}(x) + T_{m-n}(x) &= \cos((m+n)\theta) + \cos((m-n)\theta) \\
&= \cos(m\theta)\cos(n\theta) - \sin(m\theta)\sin(n\theta) + \cos(m\theta)\cos(-n\theta) - \sin(m\theta)\sin(-m\theta) \\
&= \cos(m\theta)\cos(n\theta) + \cos(m\theta)\cos(n\theta) - \sin(m\theta)\sin(n\theta) + \sin(m\theta)\sin(m\theta) \\
&= 2\cos(m\theta)\cos(n\theta) \\
&= 2T_m(\cos\theta)T_n(\cos\theta) \\
&= 2T_m(x)T_n(x)
\end{aligned} \tag{100}$$

And the second one:

$$\begin{aligned}
T_{m+n}(x) - T_{m-n}(x) &= \cos((m+n)\theta) - \cos((m-n)\theta) \\
&= \cos(m\theta)\cos(n\theta) - \sin(m\theta)\sin(n\theta) - \cos(m\theta)\cos(-n\theta) + \sin(m\theta)\sin(-m\theta) \\
&= \cos(m\theta)\cos(n\theta) - \cos(m\theta)\cos(n\theta) - \sin(m\theta)\sin(n\theta) - \sin(m\theta)\sin(m\theta) \\
&= -2\sin(m\theta)\sin(n\theta) \\
&= -2\sin^2\theta\ U_{m-1}(\cos\theta)\ U_{n-1}(\cos\theta) \\
&= 2(\cos^2\theta - 1)\ U_{m-1}(\cos\theta)\ U_{n-1}(\cos\theta) \\
&= 2(x^2-1)U_{m-1}(x)U_{n-1}(x)
\end{aligned} \tag{101}$$

.

Now a Chebyshev expansion can be made. This means that a function $f(x)$ can be expressed as a series of weighed Chebyshev polynomials:

$$f(x) = \sum_{n=0}^{\infty} \alpha_n T_n(x) \tag{102}$$

Here α_n is the projection of the funtion f on T_n: $\alpha = \langle T_n|f\rangle_1 / \langle T_n|T_n\rangle_1$.This expansion can also be done with the polynomial U_n and works exactly the same way. It is now time to take a closer look at the Chebyshev expansion (102). To make an expansion the coefficients α_n will have to be calculated. This requires integration over the weight function, which can be avoided by modifying (102).

$$f(x) = \sum_{n=0}^{\infty} \frac{\langle T_n|f\rangle_1}{\langle T_n|T_n\rangle_1} T_n(x) \tag{103}$$

Now the following orthogonal functions can also be used to expand $f(x)$:

$$\phi_n(x) = \frac{T_n(x)}{\pi\sqrt{1-x^2}} \tag{104}$$

The orthogonality of these function ϕ_n can easily be shown using the proven orthogonality of $T_n(x)$:

$$\begin{aligned}\langle\phi_n|\phi_m\rangle_2 &= \int_{-1}^{1} \frac{T_n(x)T_m(x)}{(\pi\sqrt{1-x^2})^2}(\pi\sqrt{1-x^2})\ \mathrm{d}x \\ &= \int_{-1}^{1} \frac{T_n(x)T_m(x)}{\pi\sqrt{1-x^2}}\ \mathrm{d}x \\ &= \langle T_n|T_m\rangle_1 \\ &= \frac{\delta_{n,0}+1}{2}\delta_{n,m}\end{aligned} \tag{105}$$

This means that these ϕ_n span the entire space and so a function $f(x)$ can also be written as:

$$\begin{aligned}f(x) &= \sum_{n=0}^{\infty} \frac{\langle\phi_n|f\rangle_2}{\langle\phi_n|\phi_n\rangle_2}\phi_n(x) \\ &= \langle\phi_0|f\rangle_2\,\phi_0(x) + 2\sum_{n=1}^{\infty}\langle\phi_n|f\rangle_2\,\phi_n(x) \\ &= \langle\phi_0|f\rangle_2\,\frac{T_n(x)}{\pi\sqrt{1-x^2}} + 2\sum_{n=1}^{\infty}\langle\phi_n|f\rangle_2\,\frac{T_n(x)}{\pi\sqrt{1-x^2}} \\ &= \frac{1}{\pi\sqrt{1-x^2}}\left(\langle\phi_0|f\rangle_2\,T_n(x) + 2\sum_{n=1}^{\infty}\langle\phi_n|f\rangle_2\,T_n(x)\right) \\ &= \frac{1}{\pi\sqrt{1-x^2}}\left(\mu_0 + 2\sum_{n=1}^{\infty}\mu_n T_n(x)\right)\end{aligned} \tag{106}$$

Where μ_n is defined as:

$$\mu_n = \langle\phi_n|f\rangle_2 = \int_{-1}^{1} f(x)T_n(x)\ \mathrm{d}x \tag{107}$$

This result is very useful since now the α_n are modified to μ_n, so to calculate these μ_n the weight function does no longer have to be integrated. With this result it is possible to start looking for applications of this expansion, for which the moments μ_n will have to be calculated. For this there are different techniques since μ_n depends on $f(x)$; this will be looked at in the next section.

6.2 Appendix B: Calculation of moments

In this appendix a few proofs that are not specific to the KPM are included.

Eigenvectors of a polynomial

Denote the set of normalized eigenvectors of $\tilde{H}$ as $\{|k\rangle\}$, with corresponding eigenvalues E_k. For for a polynomial P:

$$P_n(\tilde{H})\,|k\rangle = P_n(\tilde{E}_k)\,|k\rangle \tag{108}$$

To prove this, we substitute $\tilde{H}\,|k\rangle = \tilde{E}_k\,|k\rangle$ n times, using that $\tilde{E}_k$ is a constant:

$$\begin{aligned} P_n(\tilde{H})\,|k\rangle &= \sum_{i=0}^{n} a_i \tilde{H}^i\,|k\rangle \\ &= a_0\,|k\rangle + a_1\tilde{H}\,|k\rangle + a_2\tilde{H}^2\,|k\rangle + ... + a_n\tilde{H}^n\,|k\rangle \\ &= a_0\,|k\rangle + a_1\tilde{E}_k\,|k\rangle + a_2\tilde{H}\tilde{E}_k\,|k\rangle + ... + a_n\tilde{H}^{n-1}\tilde{E}_k\,|k\rangle \\ &\vdots \\ &= a_0\,|k\rangle + a_1\tilde{E}_k\,|k\rangle + a_2\tilde{E}_k^2\,|k\rangle + ... + a_n\tilde{E}_k^n\,|k\rangle \\ &= \sum_{i=0}^{n} a_i\tilde{E}_k^i\,|k\rangle \\ &= P_n(\tilde{E}_k)\,|k\rangle \end{aligned} \tag{109}$$

Independence of basis of trace

Let $O \in \mathbb{C}^{n\times n}$ be an operator and $V = \{|u_1\rangle, |u_2\rangle, ..., |u_n\rangle\}$ be an orthonomal basis of $\mathbb{C}^n$. Define $U = \begin{bmatrix} |u_1\rangle, |u_2\rangle, ..., |u_n\rangle \end{bmatrix}$, it follows that $U^\dagger U = UU^\dagger = I$ so U is unitary. Now use the property of the trace that $Tr(AB) = Tr(BA)$ for any square matrices of the same dimension A, B. Define $\tilde{O} = U^\dagger O U$, then $\sum_i \langle u_i|O|u_i\rangle = \sum_i \langle u_i|\tilde{O}|u_i\rangle = Tr(\tilde{O}) = Tr(U^\dagger O U) = Tr(UU^\dagger O) = Tr(O)$.

6.3 Appendix C: Python code used for numerical experiments

```python
%matplotlib notebook
import numpy as np
import math
import cmath
import numpy.linalg as nla
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import random
import scipy.sparse
import scipy.sparse.linalg as splin
import scipy.fftpack as fftp
from scipy.sparse import coo_matrix
import matplotlib.mlab as mlab
from math import sqrt, pi
import kwant
from cmath import exp
from types import SimpleNamespace

#The Jackson Kernel improvement
def Jackson_kernel(N):
    n = np.arange(N)
    return ((N-n+1)*np.cos(np.pi*n/(N+1))+np.sin(np.pi*n/(N+1))
       /np.tan(np.pi/(N+1)))/(N+1)

#Kwant code for defining a honeycomb lattice like graphene and
   obtaining Hamiltonian

def lattice_honeycomb_hamiltonian(a,pot,r):
    lat = kwant.lattice.honeycomb(a)
    sys = kwant.Builder()
    sys[lat.shape(circle, (0, 0))] = pot
    sys[lat.neighbors()] = hopping
    sys = sys.finalized()
    kwant.plotter.plot(sys)
    return sys.hamiltonian_submatrix(sparse=True)

#Kwant code for defining a one dimensional chain and obtaining
   Hamiltonian

def lattice_chain_hamiltonian(a,pot,L):
    lat = kwant.lattice.chain(a)
    sys = kwant.Builder()
    sys[(lat(i) for i in range(L))] = pot
    sys[lat.neighbors()] = hopping
    sys = sys.finalized()
    return sys.hamiltonian_submatrix(sparse=True)

# Adding magnetic field to the system when required, if no
   magnetic field, hopping = -1 for example.
def hopping(site_i, site_j):
    xi, yi = site_i.pos
    xj, yj = site_j.pos
    return -exp(-0.5j * magn_field_const * (xi - xj) * (yi + yj
       ))

def circle(pos):
    x, y = pos
    return x ** 2 + y ** 2 < r ** 2

#Rescaling the Hamiltonian to the interval [-1,1]
def rescale(H):
    dim=H.shape[0]
    lmax = float(splin.eigsh(H, k=1, which='LA',
       return_eigenvectors=False, ncv=25));
    lmin = float(splin.eigsh(H, k=1, which='SA',
       return_eigenvectors=False, ncv=25));
    a = (lmax - lmin) / 1.99;
    b = (lmax + lmin) / 2;
    H_rescaled = (1/a) * (H - b * scipy.sparse.eye(dim));
    return H_rescaled, a, b

def calc_init_DoS(H,N,num_vec,K):
    """Calculate the density of states of a Hamiltonian

    Parameters:
    -----------
    H : sparse matrix
        Hamiltonian of the system
    N : integer
        Number of Chebyshev moments
    num_vec : integer
        Number of random vectors used for sampling
    K  : integer
        Number of points at which the density of states is
           computed
        Must be larger than N

    Returns:
    --------
```

```
rho : 1D array
    Density of states at K points
xk  : 1D array
    Points at which the density of states is evaluated
"""
#Rescale Hamiltonian
H_rescaled, scale_fact_a, scale_fact_b = rescale(H)
dim =H.shape[0]
#Empty vector mu for constructing the moments
mu = np.zeros(N)
for r in range(num_vec):
#Make a random vector
    rand_vect = np.exp(1j * 2 * np.pi * np.random.rand(dim)
       )
    #Use iterative scheme to calculate the moments
    alpha = []
    alpha.append(rand_vect)
    alpha.append(H_rescaled.dot(rand_vect))
    #mu_single_state is the moment using only one random
       vector, add up all the states to get the moments mu
    mu_single_state = np.zeros(N, dtype=complex)
    mu_single_state[0] = (alpha[0].T.conj()).dot(alpha[0])
    mu_single_state[1] = (alpha[0].T.conj()).dot(alpha[1])
    for n in range(1,N//2):
        alpha.append(2*H_rescaled.dot(alpha[n])-alpha[n-1])
        #Use the symmetrical relation discussed in the
           section calculation of moments of the density of
            states
        mu_single_state[2*n] = 2*(alpha[n].T.conj()).dot(
           alpha[n]) - mu_single_state[0]
        mu_single_state[2*n+1] = 2*(alpha[n+1].T.conj()).
           dot(alpha[n])-mu_single_state[1]
    mu = mu + mu_single_state.real
mu = mu/num_vec/dim
mu_ext = np.zeros(K)
#Apply the Jackson Kernel improvement
mu_ext[0:N] = mu*Jackson_kernel(N)
#Use the discrete cosine transform to get back to the
   original density of states
mu_T = fftp.dct(mu_ext,type=3)
k  = np.arange(0, K)
#Define the Chebyshev nodes xk
xk_rescaled = np.cos(np.pi*(k+0.5)/K)
#Final multiplication and rescaling to get the density of
   states
```

```
    gk = np.pi*np.sqrt(1.-xk_rescaled**2)
    xk = xk_rescaled*scale_fact_a+scale_fact_b
    rho = np.divide(mu_T,gk)/(scale_fact_a)
    return rho, xk

def calc_exp_values_operator(H, A, N, num_vec, K, T=0, dt=1,
   temp=0, integrate = True, E_fermi=0):
    """Calculate time-dependent expectation values

    Parameters:
    -----------
    H : function or sparse matrix
         When H is a function H(t) returns the Hamiltonian at
            time t
    A : function or sparse matrix
         Operator of which the time dependent expectation values
             are computed
         When A is a function A(t,psi_1,psi_2) must return <
            psi_1 | A(t) | psi_2>,
    N : integer
         Number of Chebyshev moments.
    num_vec : integer
         Number of random vectors used for sampling.
    K  : integer
         Number of points for which the expectation value of A
            is calculated
         Must be larger than N
    T : float
         Time interval, optional, default = 0
    dt : float
         Time step, optional, default = 1
    integrate : boolean
         If integrate = True returns the integrated version of
            the expanded function using Fermi-Dirac weight
    temp : float
         Initial temperature of the system, optional, default =
            0


    Returns:
    --------
    Expectation_values : dense matrix
         Time-dependent expectation values, if integrate = false
             returns expectation values as a function of energy
    """
```

```
#First check time dependence and time steps
if T == 0:
    steps = 1
else:
    steps = int(round(T / dt))
#Rescale the (initial) Hamiltonian
if callable(H):
    H_0, scale_fact_a, scale_fact_b = rescale(H(0))
else:
    H_0, scale_fact_a, scale_fact_b = rescale(H)
dim = H_0.shape[0]
eye = scipy.sparse.eye(dim)
#Check whether the operator is local, in which case moments
    are returned for every site (total amount of sites L)
if callable(A):
    L = len(A(0,np.zeros(dim)*1j,np.zeros(dim)*1j))
else:
    L = 1
#Define empty vectors mu, mu_single_state to construct the
   moments
mu = np.zeros((N, steps,L))
mu_single_state = np.zeros((N,L), dtype=complex)
for r in range(num_vec):
#Define a random vector
    rand_vect = np.exp(1j *2 * np.pi * np.random.rand(dim))
    #Use the iterative scheme to calculate the moments at
       the initial time
    alpha = []
    alpha.append(rand_vect)
    alpha.append(H_0.dot(rand_vect))
    for n in range(1, N-1):
        alpha.append(2 * H_0.dot(alpha[n]) - alpha[n-1])
    for n in range(N):
        if callable(A):
            mu_single_state[n,:] = A(0,alpha[0],alpha[n])
        else:
            mu_single_state[n,:] = alpha[0].T.conj() @ A @
               alpha[n]
    mu[:,0,:] = mu[:,0,:] + mu_single_state.real
    #If time dependence: start taking time steps to
       calculate moments at every time.
    for k in range(1,steps):
        U1 = eye - 0.5j * H(k * dt) * dt # time propagators
        U2 = eye + 0.5j * H(k * dt) * dt
        mu_single_state = np.zeros((N,L), dtype=complex)
```

```
            for n in range(N):
                alpha[n]= U1.dot(splin.cgs(U2, alpha[n])[0])
                if callable(A):
                    mu_single_state[n,:] = A(k*dt,alpha[0],
                       alpha[n])
                else:
                    mu_single_state[n,:] = alpha[0].T.conj() @
                       A @ alpha[n]
            mu[:,k,:] = mu[:,k,:] + mu_single_state.real
    mu = mu/num_vec/dim
    mu_ext = np.zeros((K,steps,L))
    #Apply the Jackson Kernel to get the improved moments
    J =Jackson_kernel(N).reshape((N,1,1))
    mu_ext[:N,:,:] = mu*J
    ks  = np.arange(0, K)
    xk_rescaled = np.cos(np.pi*(ks+0.5)/K)
    xk = xk_rescaled*scale_fact_a+scale_fact_b
    gk = np.pi*np.sqrt(1.-xk_rescaled**2).reshape((K,1,1))
    #Use the discrete cosine transform to go from moments to
       original function
    expectation_values = fftp.dct(mu_ext,type=3,axis=0)/gk
    #Return the final expectation values
    E_fermi = (E_fermi-scale_fact_b)/scale_fact_a
    if not integrate:
        return np.squeeze(expectation_values)/scale_fact_a, xk
    else:
        return 1/K*np.sum(expectation_values*
           fermi_distribution(xk_rescaled,temp,E_fermi).reshape
           ((K,1,1)),axis=0)


# System properties, example input
N = 100 # number of moments
num_vec = 30 # number of random states
K = 2*N # number of points
r=100 # radius of graphene
L=15 # length of chain
pot = 0 # potential
a = 1 #lattice constant
E_fermi = 1 # Chemical potential of the system

H = lattice_honeycomb_hamiltonian(a,pot,r)
xk, rho = calc_init_DoS(H,N,num_vec,K,E_fermi)
rhoH, xkH = calc_exp_values_operator(H,H,N,num_vec,K,E_fermi)
```

References

[1] J. P. Dahlhaus, B. M. Fregoso, and J. E. Moore. Magnetization Signatures of Light-Induced Quantum Hall Edge States. *Physical Review Letters*, 114(24):246802, June 2015.

[2] A. Weiße, G. Wellein, A. Alvermann, and H. Fehske. The kernel polynomial method. *Reviews of Modern Physics*, 78:275–306, January 2006.

[3] J.R. Hook and H.E. Hall. *Solid state physics.* The Manchester physics series. Wiley, 1991.

[4] D.J. Griffiths. *Introduction to Quantum Mechanics.* Pearson Prentice Hall, 2005.

[5] D.V. Schroeder. *An Introduction to Thermal Physics.* Addison Wesley, 2000.

[6] 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.

[7] R. Peierls. On the theory of diamagnetism of conduction electrons. *World Scientific*, 80, 1933.