Quantum Transport in Nanoporous Graphene

Rasmus Kronborg Finnemann Wiuff (s163977)[*]

and Christoffer Vendelbo Sørensen (163965)[†]

Technical University of Denmark[‡]

(Dated: May 1st 2019)

Abstract: Abstract...

[*] E-mail at rwiuff@dtu.dk

[†] E-mail at chves@dtu.dk

[‡] Homepage of the Technical University of Denmark http://www.dtu.dk/english/;

Project Repository: https://github.com/rwiuff/QuantumTransport

CONTENTS

I. INTRODUCTION

Ever since its isolation and initial characterisation, graphene has been widely researched for potential applications. [Insert litt.] On a more recent timescale so called nano-porous graphene devices (NPGs) has been proposed for various applications. [Insert litt.] These devices are made up of single layered graphene with periodic holes (hence the porous) with which the intact graphene constitutes ribbons and bridges in the structures. Because of graphenes electrical properties [Insert litt], one should be able to finely control the electron currents in the devices and thus create nanometer circuits for use as e.g. chemical detectors. Because of its novelty, the fabrication of such devices are limited. It is first considered for fabrication and practical testing when theoretical simulations shows promising results. Common simulation tools for the electron transport in simple devices (albeit in large scales) are those from the SIESTA project (TBtrans), whith results analysed using SISL[1]. SIESTA generally deal with DFT calculations, which can be extrapolated using tight-binding for larger scales[2]. However DFT programs run complex calculations and might seem as a blackbox for non-physcisists. In order to better understand electron transport this project deals with a simpler approach to electron transport using only tight-binding by developing a set of tools in Python, using NumPy and numerical calculations. We utilise Greens functions and a very efficient recursion formula to gather transmission plots and band structure plots for various NPGs, whilst comparing our results with those obtained by classical DFT programs. The main scope is the development of the tight-binding scripts, comparing results with those of DFT calculations and discuss whether a clean tight-binding approach can sufficiently be used for the relatively simple NPGs. To summarise:

1. Apply quantum mechanics for electron transport in NPGs.
2. Use numerical methods (recursion algorithms, linear algebra) with NumPy to implement tight-binding.
3. Calculate band structures and transmission plots for various devices.
4. Gather single-particle Green's functions and LDOS of said devices.

II. QUANTUM TRANSPORT

In this section, the basics of the tight-binding approximation for electron transport will be explained and demonstrated. The justification for using tight-binding in the proposed nanoporous graphene device simulations will also be laid out.

A. Ballistic quantum transport

As graphene is a two dimensional material that consists of carbon atoms arranged in a hexagonal pattern, features in such a material can approach nanometer and sub nanometer scales. Because of the small scale the electrical properties of the material is vastly different from normal materials. Usually when describing the electrical properties of a material, drift-diffusion current models are used. They describe electric charges per area and current per area. However graphene can be considered a one dimensional conductor. This makes drift-diffusion models insufficient because it is based on scattering of multiple electrons and the mean free path between scattering which occurs in 2- or 3-D materials and not graphene.

Instead it is common to use the *ballistic model* when working with graphene. In the ballistic model, electrons are not affected by phonons, i.e. no electrons will be excited by phonons, and that the electrons move through the material as waves. This is because of the strong binding between atoms in graphene, reducing the effect of phonons, even at room temperature. The fact that the electrons moves as waves will prove important later on because it gives rise to *Quantum Interference* which is an essential tool to engineer graphene-based devices. Furthermore the model looks at only one electron at a time in the presence of an electron gas. This model has been used with big success for regular graphene and it seems that the ballistic model also gives a good approximation for NPGs.

B. π-orbitals and π-electrons

The main scope of this paper is dealing with electron transport in novel nanoporous graphene devices. When modeling such transport one needs to address the orbital structure of carbon lattices and later this will motivate the use of tight binding approximation and Green's functions. The two con-

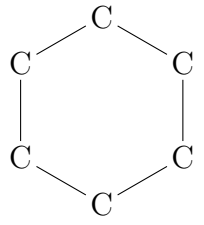


Figure 1: *Graphene lattices consists of hexagonal arrangements of carbon atoms.*

cepts of Tight-binding approximation and Green's functions will be elaborated further in the coming sections. In its basic form graphene can be divided into rings of carbon atoms as shown in Fig. 1. In the (x, y)-plane the carbon atoms are bound in sp^2 orbitals as shown in Fig. 2.

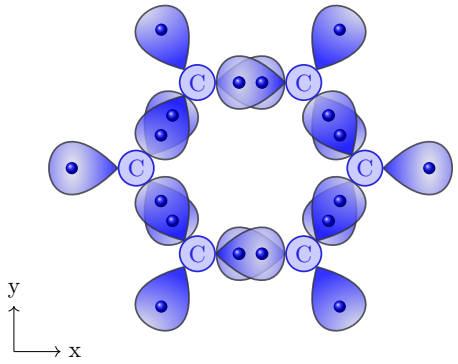


Figure 2: *Carbon atoms in a hexagonal lattice are sp^2 hybridised in the (x, y)-plane.*

This hybridisation lock all but one valence electron for the carbon atoms. These electrons exists in a p-orbital in the z-direction. Fig. 3 shows the valence orbitals of carbon.

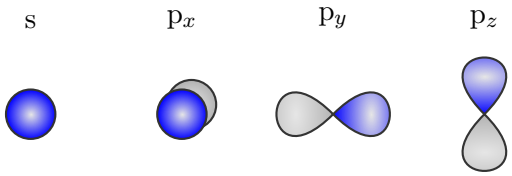


Figure 3: *The valence orbitals of carbon.*

The last electron in the p_z orbital does not mix with the tightly bound s, p_x and p_y electrons and moves freely. Thus these electrons have higher energies compared to the sp^2 electrons and occupy states at the Fermi level. These electrons dominates transport in the graphene lattice. The p_z orbital is also known as the π-orbital and as such the electron lying there is called a π-electron. Through a carbon lattice the π-electrons will travel through π-orbitals. For a benzene ring the π-electrons at the highest occupied molecular state will travel through the p_π-orbitals switching sign as they travel as shown in Fig. 4.

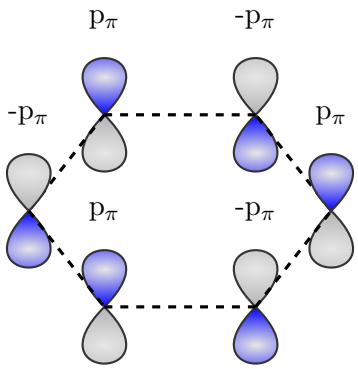


Figure 4: *When jumping from one carbon atom to another, the π-electron goes between p_π-orbitals. Such a jump is described by two matrix elements in the system's Hamiltonian.*

C. Tight-binding

Now that the transport carrying electrons are defined, one must choose a formalism for the transport itself. Introducing: **"The Tight-Binding approximation"**. In this approximation the electrons are considered being tightly bound to the atoms. Contrary to a free electron gas approximation, the electrons does not spend time in between orbitals, but jump from orbital in atom a to orbital in atom b. In this world view the Hamiltonian is represented as a matrix of hopping elements for a collection of neighbouring atomic orbitals, i.e. molecular orbitals, as well as the energy contained within each orbital (which will be addressed later on). This can be done by describing the orbitals as a Linear Combination of Atomic Orbitals (LCAO). The solution to the Schrödinger equation is then:

$$\Psi_{\mathrm{MO}} = \sum_{\alpha,R} c_{\alpha,R}\phi_\alpha(R) \tag{II.1}$$

where $\phi_\alpha(R)$ is an atomic orbital at position R, with α denoting the valence of the orbital $(2s, 2p_x, 2p_y, 2p_z)$. In electron transport the states close to the Fermi level is of interest. These are namely the highest occupied molecular orbitals (HOMO), or the lowest unoccupied molecular orbitals (LUMO). As stated earlier only the π-electrons is then of interest. The electrons' motion can be described with the hopping matrix of elements:

$$V_{pp\pi} = \langle \phi_\pi(1)| \hat{H} |\phi_\pi(2)\rangle \tag{II.2}$$

Physically this means that there is a potential between the π orbitals of neighbouring atoms 1 and 2. In our tight-binding approximation we consider only hop between nearest

neighbours. The element

$$\epsilon_0 = \langle \phi_\pi(1) | \hat{H} | \phi_\pi(1) \rangle \tag{II.3}$$

is the average energy of the electron on atom 1 and, it is common to define the hopping energy relative to this:

$$\epsilon_0 = 0 \tag{II.4}$$

If the atoms or their environment differs, so does the on-site potential.

D. The benzene molecule

As an example the Hamiltonian of benzene is considered. In Fig. 5 one can see the indices of a benzene molecule. Remember that $\langle \phi_\pi(1) | \hat{H} | \phi_\pi(1) \rangle = 0$ and Eq. (II.2), the Hamiltonian reads:

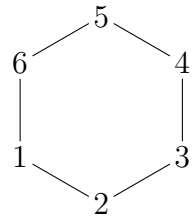


Figure 5: *Indices of a benzene molecule*

$$\mathbf{H} = V_{pp\pi} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{II.5}$$

As a helping aid, Eq. (II.5) shows the atomic indices of the atom on the top and to the left of the matrix. This will give an understanding of how to work with such matrices. The structure of the benzene molecule is rotationally symmetric and rotating the indices one sixth must yield the same Hamiltonian. Consider the energy eigenvector:

$$\phi = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{pmatrix} \tag{II.6}$$

There must exist an operator that rotates the indices as such:

$$C_6 \phi = \begin{pmatrix} c_2 & c_3 & c_4 & c_5 & c_6 & c_1 \end{pmatrix} \tag{II.7}$$

The rotated Hamiltonian is the same, and thus C_6 and $\mathbf{H}$ commute. The rotated vector must be an eigenvector with the same energy and it should be possible to find simultaneous

eigenvectors to C_6 and $\mathbf{H}$.

$$C_6\phi = \begin{pmatrix} c_2 & c_3 & c_4 & c_5 & c_6 & c_1 \end{pmatrix} = \lambda \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{pmatrix} \tag{II.8}$$

This operator C_6 is represented with the matrix:

$$\mathbf{C}_6 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{II.9}$$

It can quickly be shown that the normalised eigenvectors to C_6 are

$$\phi_n = \frac{1}{\sqrt{6}} \begin{pmatrix} \lambda_n^0 & \lambda_n^1 & \lambda_n^2 & \lambda_n^3 & \lambda_n^4 & \lambda_n^5 \end{pmatrix}, \quad \lambda_n = \exp\{-i2\pi n/6\}, \quad n = 0, 1, 2, 3, 4, 5 \tag{II.10}$$

These eigenvectors are also eigenvectors for $\mathbf{H}$ with the eigenvalues:

$$\varepsilon_n = \lambda_n + \lambda_{n-1} = 2\cos n\pi/3 \tag{II.11}$$

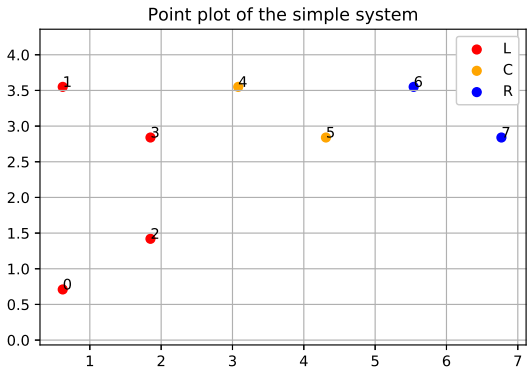
Thus thanks to the rotational symmetry it was possible to find the eigenvectors and eigenenergies for the Hamiltonian.

III. HAMILTONIAN FOR PERIODIC SYSTEMS

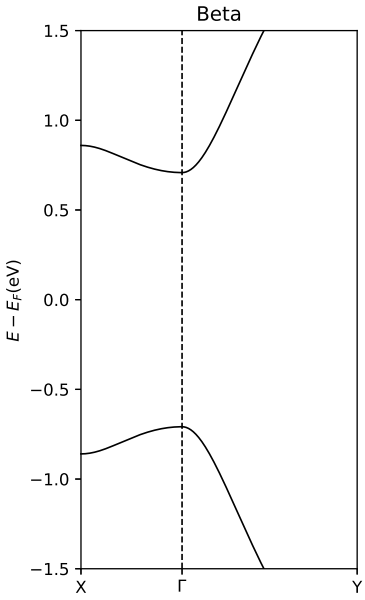
In the following sections, the focus will be to present and explain how to calculate band structures, local density of states and transmission, through *python*-programming, using Tight Binding approximation for any periodic structure. For simplicity, all initial examples and calculations will be done on a simple system, to make sure the different steps are easy to follow. The system can be seen in Fig. 6a.

A. Creating the on-site Hamiltonian and hopping matrices

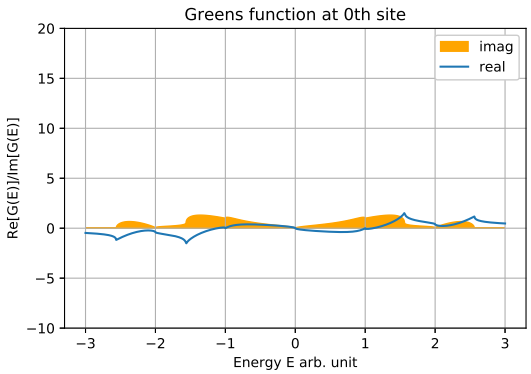
The first and most essential parts needed for calculations is the *on-site* Hamiltonian $\mathbf{h_0}$ and the *hopping* matrices $\mathbf{V}$, $\mathbf{V}^\dagger$. The starting point is a matrix, containing set of coordinates x_0, y_0, z_0, representing atom positions and a set of unit vectors $\mathbf{u}_x$, $\mathbf{u}_y$, $\mathbf{u}_z$. The unit vectors will be the basis of the unit cell containing all atom coordinates. From



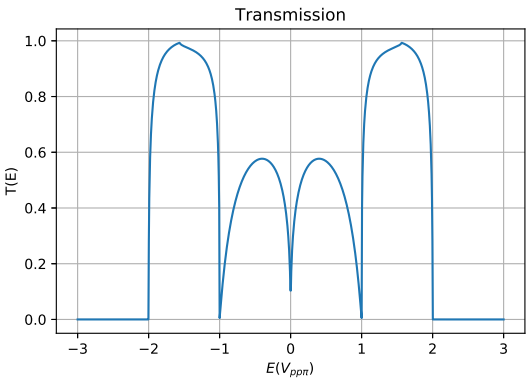
(a) *Figure showing the simple system. Every atom in the unit cell has an index, here from 0 to 7. The blue and red colours mark the left and right contacts of the system.*



(b) *Figure showing the band structure of the simple system.*



(c) *A plot showing the real and imaginary part of Green's function at the zeroth site resulting from the recursion routine on the simple system. Note that the yellow imaginary part is the representation of the local density of states.*



(d) *Figure showing the resulting transmission through the simple system*

Figure 6: *Figure of the different calculations executed on the simple system, using the developed scripts.*

now on, only the x and y coordinates will be considered as graphene is considered purely a 2D material. The on-site Hamiltonian represents the interaction of atoms within the unit cell and the hopping matrices represents the interaction of atoms between periodically repeated unit cells. In Fig. 9 a visual representation of NPG with periodically repeated

unit cells can be seen. For the rest of the report, the on-site Hamiltonian will all ways be the centre cell while the hopping matrices will be the cells surrounding the on-site Hamiltonian (See Fig. 7 for a generalised visual representation of the concept).
In practice, the scenario is that only a data set of coordinates given from scratch. The first step is to get the on-site Hamiltonian. As mentioned the on-site Hamiltonian represents the interaction of atoms within the unit cell. The approach is then to find the atoms which interact. The interaction is based on the inter-atomic distance between atoms, so naturally one wants to find the distance between all atoms in the unit cell. To do this a subtraction of all possible combinations of two sets of atom coordinates must be done. Then taking the norm of all individual results to get the distance. A function called *Onsite* have been developed to do just this. In Listing 1 the function can be seen.

```
31
32 def Onsite(xyz, Vppi):
33     h = np.zeros((xyz.shape[0], xyz.shape[0]))
34     for i in range(xyz.shape[0]):
35         for j in range(xyz.shape[0]):
36             h[i, j] = LA.norm(np.subtract(xyz[i], xyz[j]))
37     h = np.where(h < 1.6, Vppi, 0)
38     h = np.subtract(h, Vppi * np.identity(xyz.shape[0]))
```

Listing 1: *The outer operator in numpy is manifested as two nested loops. On lines xx-xx each atomic distance is calculated. Line xx replaces all nearest neighbour distances with an input potential, leaving the rest as zero. Lastly the diagonal is subtracted from the matrix.*

The function produces a matrix which contain all distances between all atoms in the unit cell. The Tight Binding model dictates that only atoms with a specific inter-atomic distance interact. Therefore the function has implemented a threshold (Listing 1 line xx) to determine whether a given distance is too great for interaction or small enough for interaction. All distances above the threshold will be changed to a 0-element in the on-site Hamiltonian matrix, representing zero interaction and all distances below the threshold will be changed to 1 to represent interaction between atoms. Finally The on-site Hamiltonian is multiplied with a on-site potential (scalar). The on-site potential $V_{pp\pi}$ differs depending on the system. Now the on-site Hamiltonian is complete and the product

is a matrix containing 0's and 1's to represent interaction between atoms in a unit cell. Moving on to the hopping matrices one first has to realise that the interactions are happening in a 2D plane. This has to be kept in mind when describing interaction between unit cells repeated in all directions in the plane. Effectively this means that six hopping matrices should be created. One in the x-direction, one in the y-direction, one in the xy-direction and their hermitian conjugates. Graphically this corresponds to a structure of this kind:

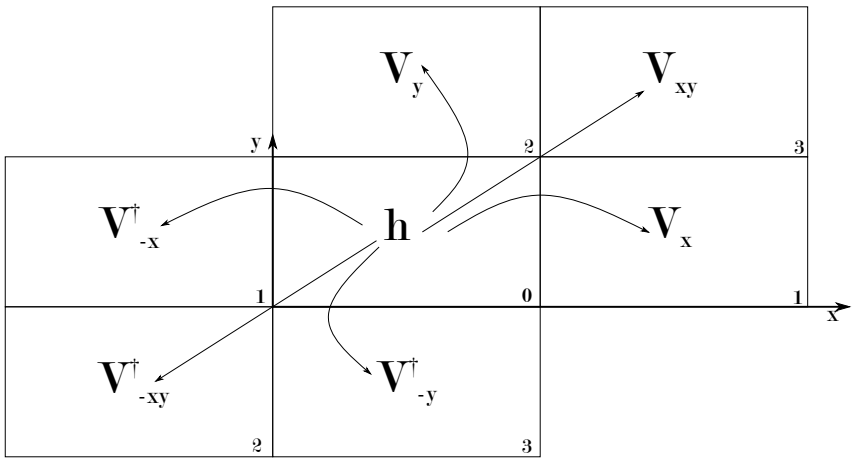


Figure 7: *Representative figure of how the on-site Hamiltonian along with its hopping matrices are structured*

In practice this is done by shifting the original x_0, y_0 coordinates by the given unit vectors $\mathbf{u}_x, \mathbf{u}_y$. By addition of the unit vectors to the original coordinate matrix one can get three new coordinate matrices $\mathbf{xy}_{shift-x} = x_0, y_0 + \mathbf{u}_x$, $\mathbf{xy}_{shift-y} = x_0, y_0 + \mathbf{u}_y$ and $\mathbf{xy}_{shift-xy} = x_0, y_0 + \mathbf{u}_x + \mathbf{u}_y$. With these three matrices what follows is basically the same method used to get the on-site Hamiltonian. The only difference being that it will be distances between atoms in the on-site Hamiltonian and the shifted matrices respectively. That way it is the distance, and thus the interaction, between the on-site Hamiltonian and the repeated unit cells that is calculated. The three resulting hopping matrices are denoted $\mathbf{V}_{1x}$, $\mathbf{V}_{2y}$ and $\mathbf{V}_{3xy}$. They represent interaction (hopping) in the "forward" direction (left-to-right) (See Fig. 7). To create the hopping matrices hopping in the "backwards" (right-to-left) direction (See Fig. 7) one simply has to transpose the hopping matrices. These matrices are denoted with a dagger: $\mathbf{V}^\dagger_{1x}$, $\mathbf{V}^\dagger_{2y}$ and $\mathbf{V}^\dagger_{3xy}$. The matrices are programmed using the developed function *Hop*, which can be seen in Appendix A, Listing 8. To show how the on-site Hamiltonian and the hopping matrices look, see Fig. 8 for a figure of the resulting matrix-maps from calculation on the small

simple system. It has been stitched together like in Fig. 7. To see how the function running the calculation for the hopping matrices look in Appendix A, Listing 8.

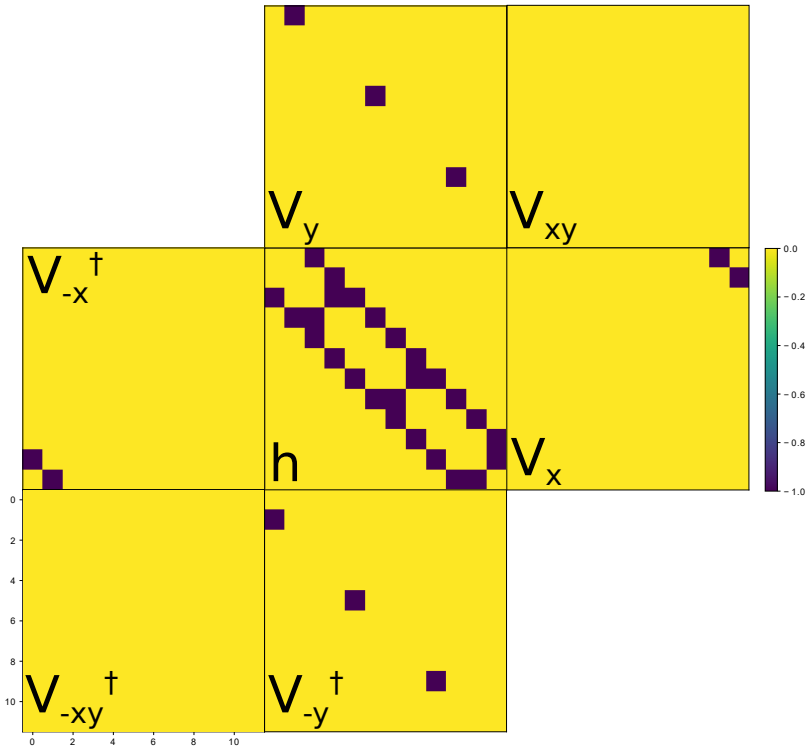


Figure 8: *Matrix maps from calculation on an arbitrary graphene system with a unit cell of 12 atoms. The on-site Hamiltonian along with all its hopping matrices are stitched together like in figure Fig. 7. All the dark spots represent a hopping of an electron to its nearest neighbour i.e. a 1 element and yellow represents a 0 element*

B. Defining the full Hamiltonian and solving the Schrödinger equation

Now that the on-site Hamiltonian along with its hopping matrices have been created, the next step is to create the full Hamiltonian in order to solve the Schrödinger equation for the system as a whole. This is an eigen-value/vector problem. The Schrödinger equation needs to be solved to get the eigen energies for the system as they will be used to produce band structure plots later on. In essence the full Hamiltonian denoted **H** is a sum of the on-site Hamiltonian and its corresponding hopping matrices multiplied by a complex

exponential function that has the appropriate phase relative to the hopping matrix:

$$\begin{aligned}\mathbf{H}(k_x, k_y) = \mathbf{h}_0 + (&\mathbf{V}_{1x}e^{-ik_x} + \mathbf{V}_{1x}^{\dagger}e^{ik_x} \\ &+ \mathbf{V}_{2y}e^{-ik_y} + \mathbf{V}_{2y}^{\dagger}e^{ik_y} \\ &+ \mathbf{V}_{3xy}e^{-ik_x}e^{-ik_y} + \mathbf{V}_{3xy}^{\dagger}e^{ik_x}e^{ik_y})\end{aligned} \tag{III.1}$$

Here k represents a continuous variable between 0 and π along the $\frac{1}{x}$- and $\frac{1}{y}$-axis in reciprocal space. Using the full Hamiltonian, the Schrodinger equation can be solved

$$\mathbf{H}(k_x, k_y)\boldsymbol{\phi}_k = \boldsymbol{\epsilon}_n(k_x, k_y)\boldsymbol{\phi}_k \tag{III.2}$$

Where $\boldsymbol{\phi}_k$ is the and $\boldsymbol{\epsilon}(k_x, k_y)$ is the eigen energies.

In practice this is done by defining a function, here called *Hkay*, that takes the on-site Hamiltonian, the hopping matrices, and $k_{x/y}$ as inputs and outputs the eigenvalues, using numpy's *numpy.linalg.eigh*. The number of eigenvalues in the output corresponds to the dimension of the full Hamiltonian. In Listing 2 the code for the function is shown.

```
50              h[-2, -2] = p
51          if struct <= 2:
52              p = float(input('Potential: '))
53              h[-1, -1] = p
54              h[-2, -2] = p
55              h[-3, -3] = p
56              h[-4, -4] = p
57      return h, p
58
59
```

Listing 2: *Function producing the full hamiltonian, corresponding to Eq. (III.1) the inputs x and y corresponds to the k_x,k_y .*

C. Producing band structures

In order to calculate and visualise the band structure of the simple system, one need to define the full Hamiltonian **H** in two directions. When working with band structures and

periodic systems it is common to note points in space with respect to the *Brillouin Zone* which is a primitive cell in reciprocal space. Therefor a continuous variable k is introduced. it extends in two directions in $(-k_x)$ and (k_y), which correspond to lengths between the symmetry points X, Γ and Y in the Brillouin zone. Here Γ is the origin $(0, 0)$. Practically this corresponds to making two plots, one for each pair of symmetry points. The y-values in each plot correspond to the eigenvalues obtained by the *Hkay* function described in Section III B. The number of eigen energies, and effectively the number of bands in the plot is dictated by the dimension of the Hamiltonian. $n \times n$-matrix $\rightarrow$ n eigen energies (bands). However plots produced in this report will only show a few of these bands in a small energy range. In the case of the simple system, the full Hamiltonian for obtaining the eigen energies that corresponds to directions X and Y are:

$$X: \ \mathbf{H}_X = \mathbf{h}_0 + (\mathbf{V}_{1x}e^{ik_x} + \mathbf{V}_{1x}^{\dagger}e^{-ik_x} + \mathbf{V}_{2y} + \mathbf{V}_{2y}^{\dagger} + \mathbf{V}_{3xy}e^{ik_x} + \mathbf{V}_{3xy}^{\dagger}e^{-ik_x}) \tag{III.3}$$

$$Y: \ \mathbf{H}_Y = \mathbf{h}_0 + (\mathbf{V}_{1x} + \mathbf{V}_{1x}^{\dagger} + \mathbf{V}_{2y}e^{-ik_y} + \mathbf{V}_{2y}^{\dagger}e^{ik_y} + \mathbf{V}_{3xy}e^{-ik_y} + \mathbf{V}_{3xy}^{\dagger}e^{ik_y}) \tag{III.4}$$

Using the eigenvalues as y-values in the two plots, putting the two plots together will yield a final plot of the band structure shown in Fig. 6b.

IV. GREENS FUNCTIONS, SELF ENERGY AND THE RECURSION ROUTINE

The Green's function and self energies play the central role when it comes to obtaining the LDOS as well as electron transport in a system. In fact, the imaginary part of the Green's function is the LDOS for a specific site in a system. What the Green's function and self energy actually is and how they come about will here be explained formally, to motivate the practical use in the following sections.

A. Green's functions and selfenergy

Firstly on should note that some of the concepts in this section will be explained using a figure of NPG as an example (Fig. 9). However, in the following section (Section IV B) the focus will revert back to the simple system from Fig. 6a.

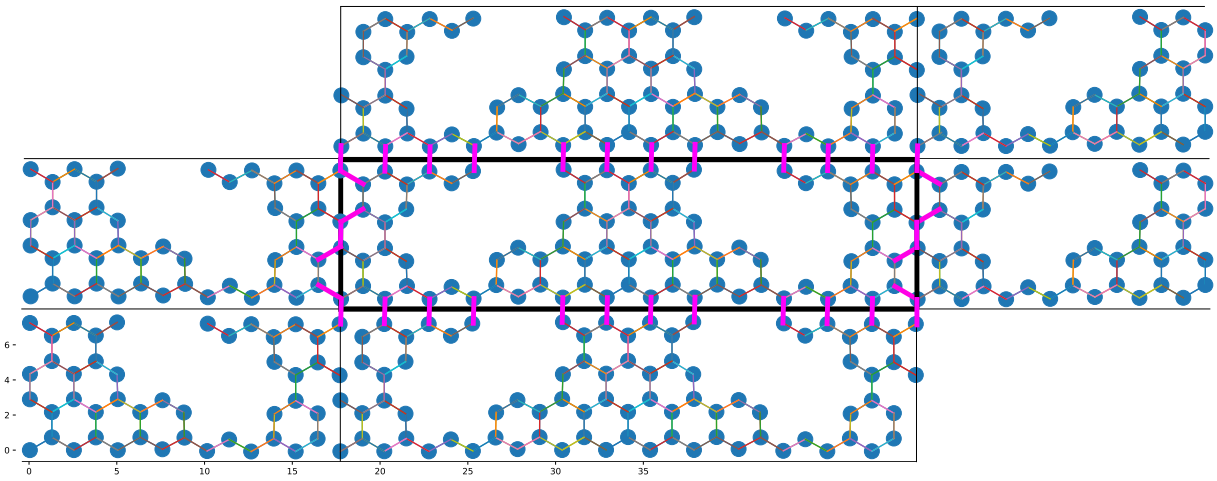


Figure 9: *Visual representation of the periodic NPG-structure. The atoms surrounded by the black box in the centre represents the unit cell. The neighbouring boxes are unit cells repeated periodically. Note that the two cells left and right with respect to the centre cell has been cut in half for figure space. The pink lines crossing the black box represents the link between the nearest neighbours in the adjacent cell.*

Imagine a system like the one in Fig. 9. It contains a unit cell in the centre, marked by a black border, surrounded by repeated unit cells in all directions. The aim is to explain how electrons move through this region. Suppose all cells surrounding the centre cell are considered "contacts" in the sense that they represent a semi-infinite chain of molecules and that they are the source of electrons (or states) that is injected in to the centre cell. What the Green's function is doing is that it "takes the states through" the centre region. It propagates the states in this particular area. In other words, the Green's function is the solution to the Schrödinger Equation in this area and the equation has the form

$$[(E + i\eta)\mathbf{1} - \mathbf{H}]\mathbf{G}(E) = \mathbf{1} \tag{IV.1}$$

From this equation one can also get the Green's function as

$$\mathbf{G}(E) = \mathbf{1}([(E + i\eta)\mathbf{1} - \mathbf{H}])^{-1} \tag{IV.2}$$

$$= [(E + i\eta)\mathbf{1} - \mathbf{H}]^{-1} \tag{IV.3}$$

The Green's functions in these equations are represented as matrices that contain all the individual Green's functions for the unit cell as well a the Green's functions for the rest of the chain. As seen in the equations, all that is needed to get the Green's function for a unit cell, in theory, is an energy and the Hamiltonian of the unit cell. Note that the solution to the Green's function matrix is a diagonal matrix with the two first off

diagonals because of rules for nearest neighbour interaction dictated by the Thigh Binding approximation. As the Green's functions for the all unit cells in a potentially semi-infinite system are needed, in practice, one has to turn to more sophisticated methods to obtain all the Green's functions, namely recursion. More on that shortly. For now this is the introduction to the Green's function. How it relates to a unit cell in a system and that it is the source of the LDOS in a unit cell.

As described one can use the Green's functions to get the propagation of states through a specific on-site Hamiltonian. However, if the system contains a range of cells, possibly infinitely many, the Hamiltonian would be of infinite size and the inversion in Eq. (IV.2) would be impossible to do practically. The solution to this, is to model a semi-infinite tight binding chain of atom/molecules and then use *recursion* on this chain. The way the recursion is done is to remove every second cell in the chain. Because the chain is semi-infinite, the yield would just be a new semi-infinite chain. Continuing this way the system can be reduced to a finite size which can actually be worked on. Say one continues to remove every second element in the chain, then in the end, the cells would be too far apart to interact an no hopping between cells would occur. At this point the recursion should stop. More on how this is done practically later. For now one just have to keep in mind that the removing cells in the chain effectively changes to coupling between them and this is where *self energy* comes in. The self energy is what describes the effective coupling between a cell and the rest of the semi-infinite chain. And it can be derived by looking at a cell at the very end of the semi-infinite chain and see how it couples to the rest. First one needs the Green's functions. The Green's matrix for this single cell would be given by the equation in Eq. (IV.2). This is before when only one cell and thus one matrix had to be considered. But now, there is an semi-infinite amount of cells and an semi-infinite amount of matrices to consider. However, the cell in the end of the chain only interacts with the cell next to it and so on. Considering this one can write op an equation equivalent to that of Eq. (IV.1) but as system of matrix equations for the chain.

$$\begin{pmatrix} z\mathbf{1}-\mathbf{H}_c & -\mathbf{V}^\dagger \\ -\mathbf{V} & (z-\varepsilon')\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{X} & \mathbf{G}_{0c} \\ \mathbf{G}_{c0} & \mathbf{G}_{00} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \tag{IV.4}$$

where ε' is the on-site Hamiltonian of the first cell, z is $E + i\eta$, $\mathbf{G}_{0c/c0}$ is the Green's matrices coupling the cell to the rest of the chain and $\mathbf{X}$ is the Green's matrices for the rest of the chain. This is also assuming one knows the Green's function within the chain $\mathbf{G}_c$ and that the chain has constant hopping and on-site elements $\mathbf{H}_c, \mathbf{V}, \mathbf{V}^\dagger$. Solving this

system for $\mathbf{G}_{00}$ and eliminating $\mathbf{G}_{0c}$, which is unknown, one gets

$$\mathbf{G}_{00}(z) = (z - \varepsilon' - \Sigma(z))^{-1} \tag{IV.5}$$

where $\Sigma(z)$ is the self-energy. One can isolate the self energy from the equations above to

$$\mathbf{\Sigma}(z) = \mathbf{V}[z\mathbf{1} - \mathbf{H}_c]^{-1}\mathbf{V}^\dagger \tag{IV.6}$$

And this concludes the formal introduction to Green's functions and self energy.

B. Obtaining first cell self-energy and Green's matrix through programming

For simplicity and in order to check whether the routine would yield the expected results, the system in Fig. 6a is use as an example. The goal is to get the Green's functions for the centre unit cell in the semi-infinite chain and the self energies coupling to rest of the chain right and left. Specifically for the simple system one should imagine first having one centre unit cell like Fig. 6a and then repeating it infinitely in the left and right direction. The fact that there is a left *and* right self energy is that the unit cell lies within the semi-infinite chain and not at the very end as described in Section IV A. To be assured, this does not conflict with any of the preciously mentioned formalism and the left and right self energies are quite easily obtained as one shall see shortly. As mentioned the goal is to get the Green's functions of a specific unit cell and the self energies related to it. If the Green's matrix $\mathbf{G}$ represents the whole chain, then the equation of the whole system would be equivalent to that of Eq. (IV.1). Considering the Green's functions for specific unit cell in question, it would correspond to one column in the system of equations, say the first. One can define the on-site Hamiltonian $\mathbf{h}_0$ for the specific unit cell and its hopping matrices $\mathbf{V}, \mathbf{V}^\dagger$. The two hopping matrices correspond to hopping left or right in the chain respectively. These can be obtained using the functions already developed in Section III. Throughout this section they will be named $a_0 = \mathbf{V}^\dagger,\ b_0 = \mathbf{V},\ e_{s0} = \mathbf{h}_s$. The recursion is an iterative process and so the zero index indicates the starting point of the iterations and the s index indicates that it is the Hamiltonian of the specific wanted cell. One can also define a Green's function for a single unit cell as $g_0 = (z - e_0)^{-1}$ just like Eq. (IV.1) where $e_0 = \mathbf{h}$ which is the on-site Hamiltonian of the other cells. With these elements a system of equations, similar to Eq. (IV.1) can be setup. The first difference being that the identity matrix is replaced by its first column, because the solution of interest is that one first column in the Green's matrix. The second is that the first element in the Hamiltonian

matrix $\mathbf{H}$ is related to the specific single unit cell $\mathbf{h}_s$. Next a range of multiplications of the different elements stated so far will be shown, and afterwards it will be explained how these affect the system of equations to give recursion. The multiplications are:

$$\begin{aligned} a_1 &= a_0 \times g_0 \times a_0 \\ b_1 &= b_0 \times g_0 \times b_0 \\ e_1 &= e_0 + a_0 \times g_0 \times b_0 + b_0 \times g_0 \times a_0 \\ e_{1s} &= e1_{0s} + a_0 \times g_0 \times b_0 \\ g_1 &= (z - e_1)^{-1} \end{aligned} \tag{IV.7}$$

These equations constitutes the first iteration in the recursion and they can be repeated indefinitely. In the matrix system of equations these multiplications effectively shifts all elements in the matrix by one column and because the matrix is diagonal, it will leave the first column of the matrix empty. The column can then be removed and this is exactly what corresponds to removing a cell in the semi-infinite chain. Keeping on doing these multiplications, raising the index by +1 every time, one can move through reduce the system as a whole removing of columns (cells) in the system of equations. In the end one will obtain re-normalised Hamiltonians and hopping matrices which is then used to get the Green's functions and self energies through these simple equations:

$$\begin{aligned} \mathbf{\Sigma}_R &= e_s - h \\ \mathbf{\Sigma}_L &= e - h - \mathbf{\Sigma}_R \\ \mathbf{G00} &= (z - e_s)^{-1} \end{aligned} \tag{IV.8}$$

Programming this recursion is fairly simple as all which is needed is a while loop which iterates over the equations in Eq. (IV.7) until a threshold has been reached. The threshold is determined by the value of the hopping matrix a_0. As it reaches a value close to zero, there is no longer any effective interacting (hopping) between the cells because of removal of cells and the recursion should stop. In Listing 3 the code for the routine is shown. Line xx-xx is the listing of elements before iteration, xx-xx is the while loop with the equations from Eq. (IV.7). Note that some intermediate multiplications are made f.ex. *ag = a0 @ b0*. This is for run-time optimisation only. In line xx-xx the iteration indexed 0 gets redifned so that it corresponds to the most recent iteration. Finally in xx-xx the definition of the outputs as per Eq. (IV.8) is stated.

```
62      for i in range(xyz1.shape[0]):
63          for j in range(xyz.shape[0]):
64              hop[i, j] = LA.norm(np.subtract(xyz1[i], xyz[j]))
65      hop = np.where(hop < 1.6, Vppi, 0)
66      return hop
67
68
69  def Hkay(Ham, V1, V2, V3, x, y):
70      Ham = Ham + (V1 * np.exp(-1.0j * x)
71                   + np.transpose(V1) * np.exp(1.0j * x)
72                   + V2 * np.exp(-1.0j * y)
73                   + np.transpose(V2) * np.exp(1.0j * y)
74                   + V3 * np.exp(-1.0j * x) * np.exp(-1.0j * y)
75                   + np.transpose(V3) * np.exp(1.0j * x) * np.exp(1.0j * y))
76      e = LA.eigh(Ham)[0]
77      v = LA.eigh(Ham)[1]
78      return e, v
79
80
81  def RecursionRoutine(En, h, V, eta):
82      z = np.identity(h.shape[0]) * (En - eta)
83      a0 = V.conj().transpose()
84      b0 = V
85      es0 = h
86      e0 = h
87      g0 = LA.inv(z - e0)
88      q = 1
89      while np.max(np.abs(a0)) > 1e-6:
```

Listing 3: *The while loop in the recursion routine. The matrix elements are overwritten with the new variables until the resulting matrix is small enough to diagonalise*

This concludes how recursion works and how the first cell Green's function as well as the

self-energies is obtained.

C. Plotting the real and imaginary part of the first cell Green's function

One of the results possible to obtain via the recursion routine is the Green's function of the centre unit cell in relation to the rest of the chain. As mentioned the imaginary part of the elements Green's matrix is the LDOS of the different sites in the unit cell. With a relatively simple approach, the Green's matrix elements can be obtained as a function of energy, using a *for loop*, looping over a range of energies which is then used as input in the *RecursionRoutine* function (Listing 3), see Listing 4:

```
64 G00 = np.zeros((En.shape[0]), dtype=complex)
65 for i in range(En.shape[0]):
66     G, SelfER, SelfEL = RecursionRoutine(En[i], h, V, eta)
67     G = np.diag(G)
68     G00[i] = G[0]
```

Listing 4: *Code showing the loop which produces the complex Green's function (or y) values for a range of energies used in the plot.*

This gives information about the LDOS at a specific energy and place in space, namely a specific atom in the unit cell. The resulting plot for the simple system (atom index 0) can be seen in Fig. 6c. As seen in the plot... Note that the plot only represents the LDOS for a specific site on the molecule and that they may change radically from site to site (see Appendix A, Fig. 18 for an example using the same system as Fig. 6a). The site can be changed by choosing another index in Listing 4 line 68, which corresponds to the atom indices in Fig. 6a.

V. TRANSMISSION ROUTINE

This section will mark the conclusion of the preliminary work done in Sections II to IV. All the functions producing on-site Hamiltonians, hopping matrices, full Hamiltonians, band structures as well as self energy and Green's functions by recursion, will be used to get the transmission through the material.

First of all a sentence as to what transmission is: Transmission is the probability of an

electron being transported trough a specific region for a specific range of energies and thus how the region affects the overall current flow of electrons through the system as a whole. Below is an equation stating it formally

$$P(t)_{mn} = \left| \langle m|e^{i\mathbf{H}t/\hbar}|n\rangle \right|^2 \tag{V.1}$$

where m, n is the density of states in each side of the region of interest (states going in/out) and $e^{i\mathbf{H}t/\hbar}$ is the solution to the Green's function.

To give an overview and explain the different concepts of transmission this section will rely heavily on Fig. 10 where all the different parts of the system have been translated from the actual material into mathematical formalism in the shape of matrices. The first and central piece is the so-called "Device Region" with the on-site Hamiltonian $\mathbf{H}_D$ (Green area in Fig. 10). The device region contains at least one central unit cell as well as a "left" and "right" unit cell (Red and blue area in Fig. 10). The left and right unit cells represent the contact region of the device i.e. the two parts that connects to the rest of the system/molecule. They have on-site Hamiltonians $\mathbf{H}_L, \mathbf{H}_R$ and they interact with rest of the system via hopping matrices $\mathbf{V}_{L/R}, \mathbf{V}_{L/R}^\dagger$. As $\mathbf{H}_D$ contains $\mathbf{H}_L$, $\mathbf{H}_R$ they can be picked out of $\mathbf{H}_D$, without further calculation (See Fig. 10) once the $\mathbf{H}_D$ has been calculated. The cells next to the contact region can be reduced into a single Hamiltonian by recursion to have same dimension as $\mathbf{H}_L, \mathbf{H}_R$. Note that $\mathbf{H}_L, \mathbf{H}_R$ need not be the same dimensions. Related to $\mathbf{H}_L, \mathbf{H}_R$ is the left and right self energies $\mathbf{\Sigma}_{L/R}$ and on-site Green's matrices $\mathbf{g}_L$, $\mathbf{g}_R$. These can be obtained using the theory and developed methods from Section IV. However the aim is to obtain the Green's matrix for the device region, $\mathbf{G}_D$, as it is the one needed to fully describe the transmission in the region of interest. In other words, propagation of the states in the green area in Fig. 10. To obtain it, one simply has to keep in mind that the effective coupling of the device region to the rest of the system is determined by the two contact regions $(\mathbf{H}_L, \mathbf{H}_R)$ and thus the correction to the device Green's functions will be determined by the self energy of those contact regions. So the Green's matrix will be given by the same equation as Eq. (IV.2) but with a self energy correction:

$$\mathbf{G}_D = [\mathbf{1}(E + i\eta) - \mathbf{H}_D - \mathbf{\Sigma}_L(E) - \mathbf{\Sigma}_R(E)]^{-1} \tag{V.2}$$

Looking back at Eq. (V.1) the Green's function needed has been obtained, but what about the states going in and out of the device region $\langle m|$ and $|n\rangle$? First thing needed to describe how the density of states pass through is by which rate they pass. To do this the

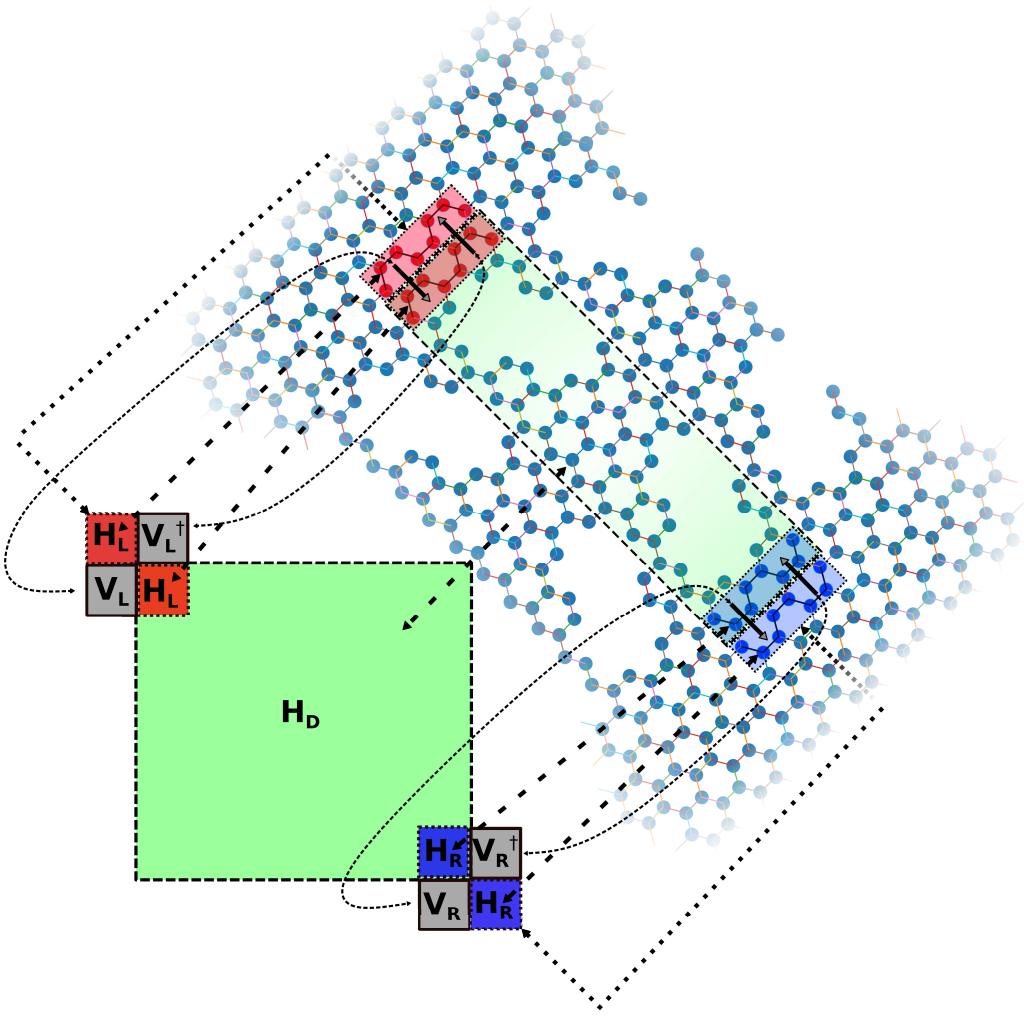


Figure 10: *Illustration showing how the different parts of the system are translated into matrix blocks in NPG. The green box is the unit cell of the device with the Hamiltonian $\boldsymbol{H}_D$. It includes one red and blue box which themselves are unit cells of the left and right contacts and have the Hamiltonians $\boldsymbol{H}_L$, $\boldsymbol{H}_R$. The two other unit cells lying outside the device region represents what could be an infinite contact region reducible by recursion. Finally the two fat black arrows (not dotted) on each side of the device represents the hopping between the device- and contact region. Note that the direction of hopping corresponds to a specific hopping matrix. F.ex. left-to-right is the ordinary hopping matrix ($\mathbf{V}$) while right-to-left is its conjugate ($\mathbf{V}^\dagger$) (for both left and right side of the device).*

rate operators $\boldsymbol{\Gamma}_L, \boldsymbol{\Gamma}_R$ are introduced... They are given by

$$\boldsymbol{\Gamma}_{L/R} = i(\boldsymbol{\Sigma}_{L/R} - \boldsymbol{\Sigma}_{L/R}^\dagger) \tag{V.3}$$

Now the rate matrices have another important function as they can be used to rewrite the equation for the *spectral function*:

$$\mathbf{A}(E) = -2\mathrm{Im}[\mathbf{G}(E)] \tag{V.4}$$

$$\downarrow \tag{V.5}$$

$$\mathbf{A}(E) = \mathbf{G}(E)\mathbf{\Gamma}(E)\mathbf{G}^{\dagger}(E) \tag{V.6}$$

To explain what the spectral function is, it is convenient to know that it too can be divided into two parts, a left and a right one.

$$\mathbf{A}(E) = \mathbf{A}(E)_L + \mathbf{A}(E)_R \tag{V.7}$$

$$\mathbf{A}(E)_L = \mathbf{G}(E)\mathbf{\Gamma}(E)_L\mathbf{G}^{\dagger}(E) \tag{V.8}$$

$$\mathbf{A}(E)_R = \mathbf{G}(E)\mathbf{\Gamma}(E)_R\mathbf{G}^{\dagger}(E) \tag{V.9}$$

Taking the left spectral function as an example, it represents the density of states (note Eq. (V.4)) of a wave coming from the left entering the device. Now one can also write the equation

$$\mathbf{A}_L(E)\mathbf{\Gamma}_R(E) = \mathbf{G}(E)\mathbf{\Gamma}(E)_L\mathbf{G}^{\dagger}(E)\mathbf{\Gamma}_R(E) \tag{V.10}$$

This corresponds to the density of states coming from the left, which then pass on through the right electrode by the rate $\mathbf{\Gamma}_R(E)$ and because of time-reversal symmetry one can also get the density of states coming from the other direction.

$$\mathbf{A}(E)_L\mathbf{\Gamma}(E)_R = \mathbf{A}(E)_R\mathbf{\Gamma}(E)_L \tag{V.11}$$

This ultimately leads to transmission because transmission is in essence an expression of how much of the density of states passes trough the device and as explained Eq. (V.10) is exactly the density of states, coming from the left (or right) and then passes through the right (left) by rate $\mathbf{\Gamma}_{L/R}(E)$. So using the Green's function, left/right self energies as well as the left/right rate matrices, the transmission, as a function of energy, can be obtained via the following equation:

$$T(E) = \mathrm{Tr}[\mathbf{\Gamma}_R\mathbf{G}_D\mathbf{\Gamma}_L\mathbf{G}_D^{\dagger}](E) \tag{V.12}$$

Where Tr is the trace of the matrix product.

A. Transmission in 1D

Again the developed routine in this section will be used on the simple system (Fig. 6a) as an example and in order to make sure that the obtained results are as expected. Thereafter it will be generalised to suit all kinds and sizes of system. First thing is to define the device in the same manner as Fig. 10 so that the device Hamiltonian $\mathbf{H}_D$ can be obtained through the already defined function *Onsite*. The left and right Hamiltonian $\mathbf{H}_{L/R}$ are thus picked out as described earlier. An implementation has been made to the script to allow the user to see the left and right contact cells graphically as red and blue marked atom indices in the plot of the unit cell (See Fig. 6a). This allows the user to get an overview of dimensions of $\mathbf{H}_L, \mathbf{H}_R$ so they can be picked out of the device Hamiltonian correctly. From $\mathbf{H}_L, \mathbf{H}_R$ the corresponding hopping matrices are defined using the *Hop* function. Then the developed *EnergyRecursion* function is used to obtain the device Green's matrix. This function is a more elaborate version of the earlier mentioned *RecursionRoutine* and it takes $\mathbf{H}_D, \mathbf{H}_L, \mathbf{H}_R, \mathbf{V}_L, \mathbf{V}_R$, a range of energies and η as inputs. It uses the old recursion routine to calculate the self energies for the left and right cells ($\mathbf{\Sigma}_{L/R}$) (see line xxx-xxx Listing 5) and then uses those to calculate the device Green's function $\mathbf{G}_D$ as well as the left and right rate matrices $\mathbf{\Gamma}_{L/R}$, using equations Eq. (V.2), **??** (see Listing 5 line xxx-xxx).

```
156         input('Left contact atomic indices (#-#): '), dtype=int, sep='-')
157     R = np.fromstring(
158         input('Right contact atomic indices (#-#): '), dtype=int, sep='-')
159     L = np.arange(L[0], L[1] + 1, 1, dtype=int)
160     R = np.arange(R[0], R[1] + 1, 1, dtype=int)
161     C = np.arange(L[1] + 3, R[0], 1, dtype=int)
162     RestL = np.arange(0, L[0], dtype=int)
163     RestR = np.arange(R[1] + 3, xyz.shape[0], 1, dtype=int)
164     print(RestL)
165     print(L)
166     print(C)
167     print(R)
168     print(RestR)
169
```

```
170     Lxyz = xyz[L]
171     Rxyz = xyz[R]
172     Cxyz = xyz[C]
173     RmArray = np.append(L, C).astype(int)
174     RmArray = np.append(RmArray, R).astype(int)
175     Restxyz = np.delete(xyz, RmArray, 0)
176
177     plt.scatter(Lxyz[:, 0], Lxyz[:, 1], c='red', label='L')
178     plt.scatter(Cxyz[:, 0], Cxyz[:, 1], c='orange', label='C')
179     plt.scatter(Rxyz[:, 0], Rxyz[:, 1], c='blue', label='R')
180     plt.scatter(Restxyz[:, 0], Restxyz[:, 1], c='k')
181     plt.legend()
182     plt.title('Point plot of the simple system')
183     plt.axis('equal')
184     for i in range(xyz[:, 0].shape[0]):
185         s = i
```

Listing 5: *Code showing how the device Green's functions as well as the left and right rate matrices are computed. First defining the different parts as described in the text above (line xx-xx), then looping over a range of energies (line xx-xx), using the old Recursion Routine and then inputting the self-energies in the equations from Eq. (V.2) and Eq. (V.3) (line xx-xx).*

The output of the *EnergyRecursion* function is the two rate matrices $\mathbf{\Gamma}_{L/R}$ as well as the device Green's function $\mathbf{G}_D$ and as per **??** the matrices needed for transmission have been obtained. As seen in Listing 6 the function *Transmission* simply carries out the matrix product and subsequent trace of the matrices resulting from *EnergyRrecursion* and outputs a range of transmission probabilities which is then plotted against an energy range. Do mind that this is still just 1D in the sense that the transmission only moves in one direction. A plot of the transmission for the simple 1D system (the one in Fig. 6a) can be seen in Fig. 6d.

```
188     plt.grid(b=True, which='both', axis='both')
189 #    savename = 'pointplot.eps'
```

```
190 #    plt.savefig(savename, bbox_inches='tight')
191     plt.show()
192     return RestL, RestR, L, R, C
193
194
195 def EnergyRecursion(HD, HL, HR, VL, VR, En, eta):
```

Listing 6: *Code showing how the transmission probabilities are created. Taking in the rate matrices, device Green's function and a range of energies it takes the trace of the matrix product of for a range of energies as in Eq. (V.12) (line xx-xx).*

B. Development of transmission to 2D

Lastly the transmission routine needs to be generalised so it can handle transport in two directions. The most convenient approach is to work with five real space unit cells as a starting point. One centre cell, a right and a left cell representing the contacts and then two additional cells on the left and right, representing the rest of the contact region. This would be the minimum amount of cells needed to generalise the transmission. One might have more center cells if the structure changes from one of those cells to the other. First these five unit cells will be defined using already developed tools. Then a big Hamiltonian, including all coordinates from the five cells representing real space are created, again using existing functions. One could call this big Hamiltonian $\mathbf{H}_{\text{Bigreal}}(x_0y_0z_0, x_0y_0z_0)$. It is a function of two sets of identical coordinates, namely the ones used to create the Hamiltonian itself. This has also been the case for all previous calculations, but the following steps will make it clear why it is explicitly stated for this Hamiltonian. The left/right on-site Hamiltonians and hopping matrices can thus be picked out of $\mathbf{H}_{\text{Bigreal}}(x_0y_0z_0, x_0y_0z_0)$ as before to get the Green's functions, self energies for transmission in real space, just as in the previous section. However, before the Green's function, self energies and transmission are calculated, the full left/right on-site Hamiltonian as well as their hopping matrices are defined as functions of a variable k which is added as a phase to the hopping matrices. As an example the following is the equation for the full right side Hamiltonian using the right on-side Hamiltonian with its hopping matrices: $\mathbf{H}_R(k) = \mathbf{V}_R e^{ik} + \mathbf{V}_R^\dagger e^{-ik} + \mathbf{h}_R$. This added k-variable phase corresponds to transmission in the direction transverse to that in real

space and it has 2π periodic boundary conditions. This means that transport of electrons in the transverse direction in only dependent on a phase. However, one still needs the interaction between the centre on-site Hamiltonian (or device Hamiltonian) and the cells repeated in the transverse direction. Firstly the hopping between $\mathbf{H}_{\text{Bigreal}}(x_0y_0z_0, x_0y_0z_0)$ and a transversely shifted Hamiltonian is defined as $\mathbf{W} = \mathbf{H}_{\text{Bigtrans}}(x_0y_0z_0, x_1y_1z_1)$ (Here using $\mathbf{W}$ as not to confuse it with $\mathbf{V}$ which is hopping between cells in real space). The index of 1 in the second set of coordinates in $\mathbf{W}$ is to mark that the original coordinates have been shifted in the transverse direction. With the hopping matrices $\mathbf{W}$ for the transverse direction defined a Hamiltonian dependent on k-point values can be defined as:

$$\begin{aligned}\mathbf{H}(k) &= \mathbf{h} + \mathbf{W}e^{ik} + \mathbf{W}^{\dagger}e^{-ik} \\ &= \mathbf{H}_{\text{Bigreal}}(x_0y_0z_0, x_0y_0z_0) + \mathbf{H}_{\text{Bigtrans}}(x_0y_0z_0, x_1y_1z_1)e^{ik} + \mathbf{H}^{\dagger}_{\text{Bigtrans}}(x_0y_0z_0, x_1y_1z_1)e^{-ik}\end{aligned} \tag{V.13}$$

Now that a Hamiltonian, dependent of a variable k has been defined, it is now possible to get self energies, Green's functions that is k-dependent as well. This means that the transmission will also k-dependent. Thus transmission in 2D is essentially defined by a continuous variable in a range energies in real space and a continuous variable between $-\pi, \pi$ in inverse space (transverse direction). This hereby concludes the all the initial effort to explain the formalism needed to get transmission in a two dimensional material such as NPG. Following is a walk through as to how this last step has been implemented through code programming.

To get the transmission in 2D the function *PeriodicHamiltonian* is nested in a for loop, looping over transverse k-points (See Listing 7 line xx-xx). The function *PeriodicHamiltonian* basically translates the device Hamiltonian in the transverse direction (See Appendix A, Listing 9 for the code piece). Picked out from the resulting Hamiltonian is the left/right device Hamiltonian and left/right hopping matrix (See Listing 7 line xx-xx. Still within the loop the *EnergyRecursion* is used to get the device Greens function, and left/right rate matrices and lastly the transmission is done with the *Transmission* function (Listing 7 line xx). The function *Transmission* basically computes **??** as it stands, it can be seen in Appendix A, Listing 10. All these functions are used per k-point which effectively gives transmission in two dimensions.

```
34 for i in kP:
35     print('-------------------------------------------------------------------------')
36     print('Calculating for k-point:    {}'.format(i))
```

```
    Ham = PeriodicHamiltonian(xyz, UY, i)
    HL = Ham[L]
    HL = HL[:, L]
    HR = Ham[R]
    HR = HR[:, R]
    VL = Ham[L]
    VL = VL[:, RestL]
    VR = Ham[RestR]
    VR = VR[:, R]
    # gs = GridSpec(2, 2, width_ratios=[1, 2])
    # a = plt.figure(figsize=(7, 4))
    # ax1 = plt.subplot(gs[:, 1])
    # plt.imshow(Ham.real)
    # ax2 = plt.subplot(gs[0, 0])
    # plt.imshow(HL.real)
    # ax3 = plt.subplot(gs[1, 0])
    # plt.imshow(HR.real)
    # a.show()
    # b = plt.figure(figsize=(7, 4))
    # plt.subplot(121)
    # plt.imshow(VL.real)
    # plt.subplot(122)
    # plt.imshow(VR.real)
    # b.show()
    # input('Press any key to continue')

    GD, GammaL, GammaR = EnergyRecursion(Ham, HL, HR, VL, VR, En, eta)

    G = np.zeros((En.shape[0]), dtype=complex)
    bar = Bar('Retrieving Greens function ', max=En.shape[0])
    for i in range(En.shape[0]):
        G[i] = GD["GD{:d}".format(i)].diagonal()[0]
```

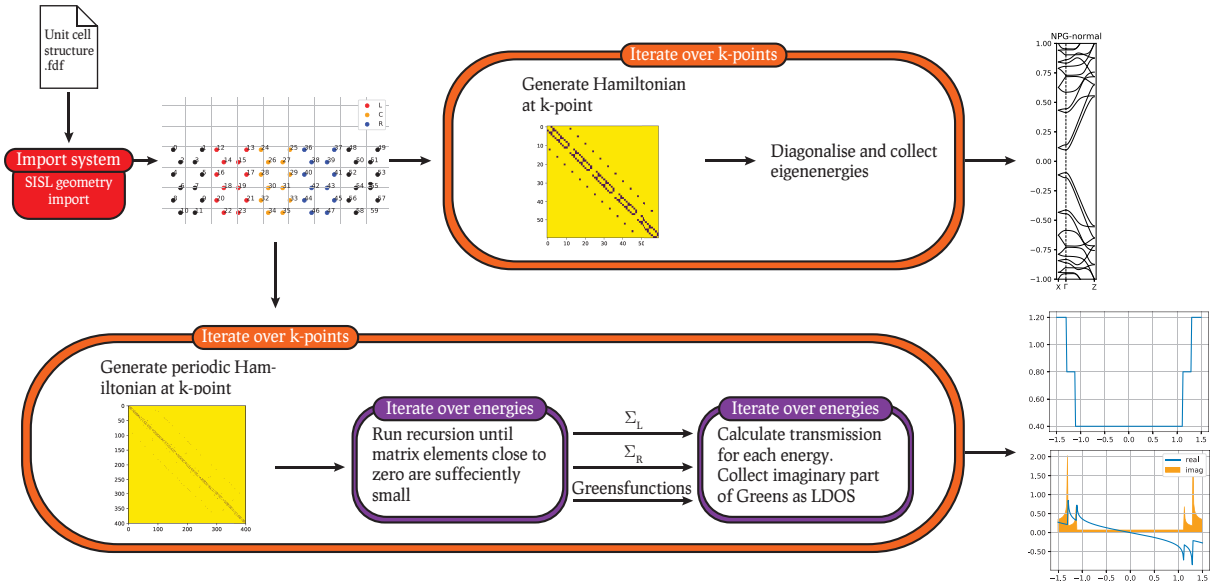


Figure 11: *Flowchart depicting the routines run in python. SISL is used to import the geometry. Afterwards the coordinates are either used for band structure plots or for transmission plots. For the band structures, a Hamiltonian at each desired k-point is generated and diagonalised in order to get the eigen energies. These energies are then plotted. With transmission, a periodic Hamiltonian at various transverse k-points are generated and reduced to self energies in the transport direction (using the recursion algorithm). The self energies and the Green's functions retrieved here are then multiplied as to get the transmission.*

```
69          bar.next()
70      bar.finish()
71
72      T = Transmission(GammaL=GammaL, GammaR=GammaR, GD=GD, En=En)
```

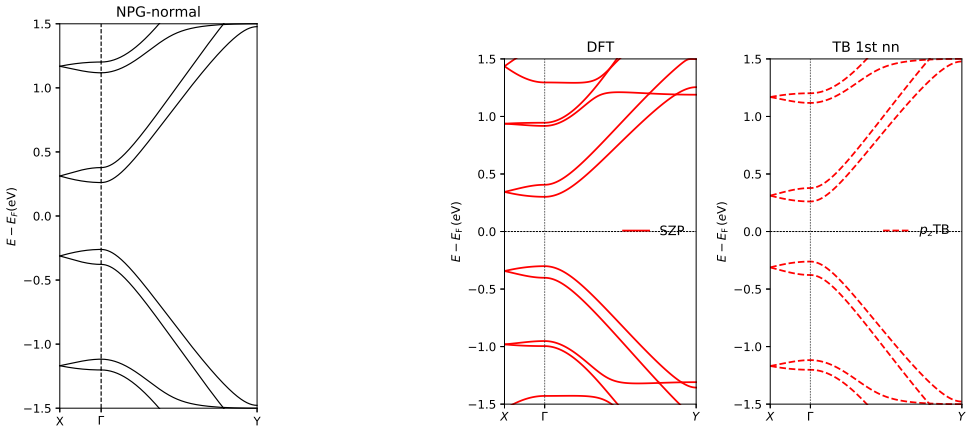
Listing 7: *Code piece showing the transmission routine.*

Plots for transmission per k-point will be shown for NPG in Section V D.

C. Summary of Methodology

D. Comparing Tight Binding with DFT and TBtrans for transmission and band structure calculations in NPG

All the scripts necessary for calculation have been developed and they can now be used on a system of NPG. The following results are based on this structure similar to that of Fig. 9. Firstly the band structure obtained using the script described in Section III B is shown together with band plots obtained from DTF and TBtrans calculations.

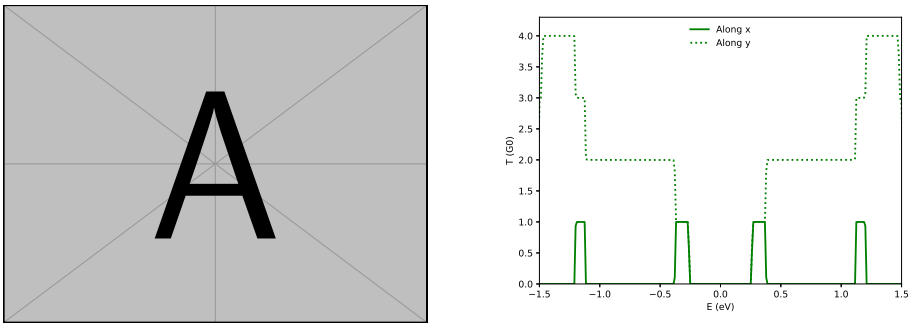


(a) *Figure showing the band structure for NPG with normal bridges obtained with the script described in Section III B*

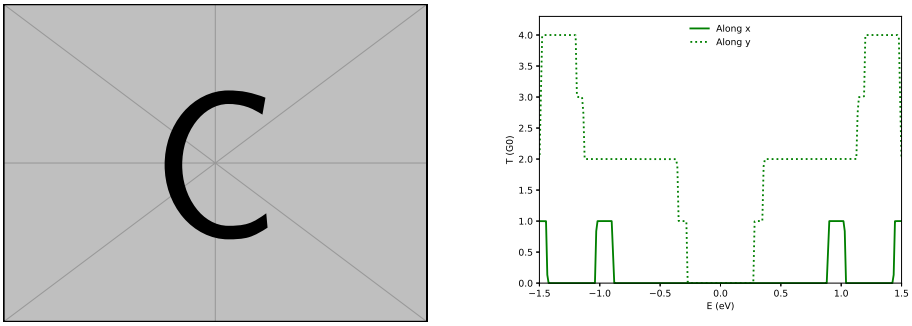
(b) *Figure showing the bands structures obtained from DFT and TBtrans calculations.*

Figure 12: *Figure showing how the band plots compare for DFT, TBtrans and the developed script.*

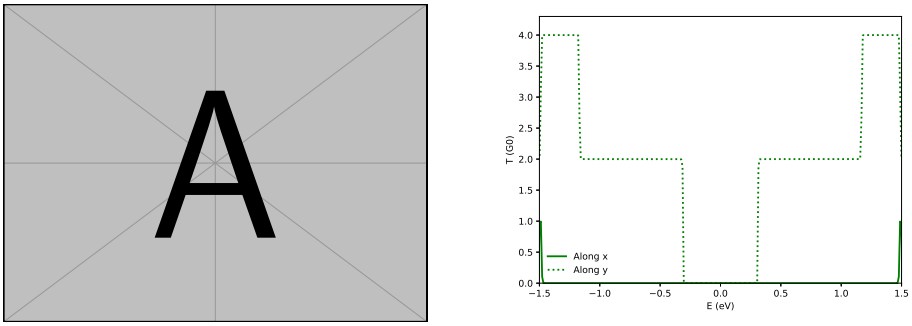
The band plot obtained in Fig. 12a shows almost 1-to-1 correspondence with the plot obtained using TBtrans Fig. 12b (right) and is also very similar to the plot obtained from DFT Fig. 12b (left), proving that the script is capable of creating band structures for NPG-systems. Next is a comparison of the transmission in NPG for different k-points in reciprocal space. The plots are made for transmission through NPG in real space (x-direction) for each three different k-points in the reciprocal space ($\frac{1}{y}$-direction). The three k-points are 0, $\frac{\pi}{2}$, π. Additionally an average over these k-points is plotted as well. In Fig. 13 transmission plots obtained with the script described in Section V B is compared with transmission plot obtained through DTF, using the same k-points.



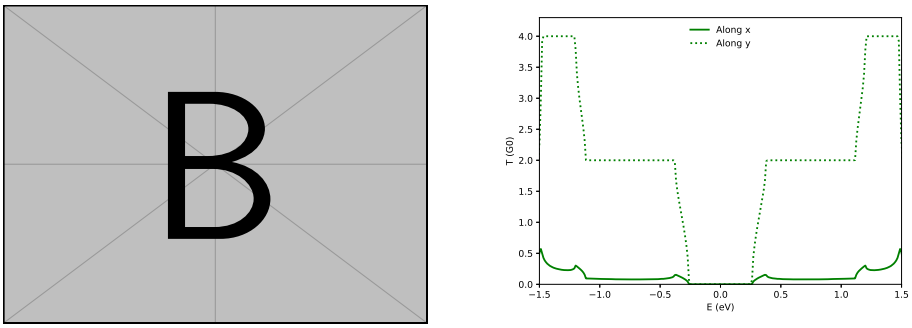
(a) *Transverse k-point 0*



(b) *Transverse k-point* $\frac{\pi}{2}$



(c) *Transverse k-point* π



(d) *Average over k-points* $0,\ \frac{\pi}{2},\ \pi$

Figure 13: *Figure showing the comparison between plots of transmission through NPG obtained using the developed scripts (left/blue) and obtained using DFT (right/green).*

As one can see in Fig. 13 The plots show very good correspondence with the DFT calculation. Again proving that the scripts developed on the basis of the Tight Binding approximation can produce valid results, when it comes to band structure and transmission through NPG.

This concludes the preliminary work. The tools for calculation of band structures, LDOS and transmission has been successfully developed, using *python*-programming and the Tight Binding approximation. Following will be a range of tests on different NPG systems, using the developed tools.

VI. EXPLORING FUNCTIONALITY OF GNR BRIDGES

In this section a range of tests will be conducted on different NPG structures in order to uncover the effect of manipulation of the bridges between the GNR's in the NPG. From an applied perspective, one of the main motivations is to find out how these bridges can be manipulated in order to control the current through the material. Some studies has already shown how one could possibly confine current flow to a single GNR channel by manipulation of bridges between GNR's utilising *Quantum Interference* effects, thus providing a solution to an important aspect in carbon-based nanocircuitry design, namely confinement of electron flow. The study was focused on the difference in effect of having *meta* and *para* bridges between GNR's. Meta and para bridges are essentially benzene ring oriented in two different ways (more on that in the following sections). This section will, in part, be an attempt to reproduce some of the results of the study and also to see what happens when if other species of atoms are added to these benzene rings in the meta and para bridges. In Table I a schematic overview of the different tests can be seen. The results from these tests will be presented in the form of band structure plots and they will be compared to band structure plots produced with DFT and TBtrans. Additionally a schematic overview of the different systems, showing the on-site potentials of each atom will be presented in the figures as well. Following is a section introducing meta and para NPG in detail.

A. Differences in para and meta bridges

In broad terms the difference in the meta and para structures lies in the path an electron will travel through the benzene ring to get across the bridge. In the para bridge, the path

Test no.	Meta/Para	Symmetry	No. of species	Added species	Name
1	Para	Symmetric	4	Oxygen	PS4O
2	Para	Symmetric	4	Hydroxide	PS4OH
3	Meta	Symmetric	2	Oxygen	MS2O
4	Meta	Symmetric	2	Hydroxide	MS2OH
5	Meta	Asymmetric	2	Oxygen	MA2O
6	Meta	Asymmetric	2	Hydroxide	MA2OH

Table I: *Table showing an overview of all the structures that will be tested in this section. How the different species will be manipulated during the tests, will be stated in plots produced for results. The code names follow the table column-wise. In Appendix A, Fig. 20 a collection of figures showing each scenario stated in this table.*

across the aromatic ring is symmetric and so the electron will pass above or below with equal probability. Since the para bridge has three bonds in each direction across the ring, the path length in the para bridge is the same on each side (See **??**). This will cause constructive interference of the states once the waves meet on the other side of the ring, which results in spreading in the density of states from one GNR to the next. For the meta bridge the way across the aromatic ring is not symmetric in the sense that there is two bonds across the path below and four bonds across on the path above (see **??**). This will cause a shift by half a wavelength between the two paths and thus create destructive interference between waves meeting on the other side of the ring. The effect of this is confinement in the density of states in the GNR's.

B. Tests with modified meta and para NPG

In the following sections various tests on the different systems will be conducted using the developed program. The motivation for these test is proposal that NPG can be "tuned" by addition or removal of hydrogen. Tuning with hydrogen is practically feasible, so by using the developed program, it will be possible to simulate different scenarios which will uncover what is going on, chemically as well as physically, in the different NPG systems when manipulated. The tests consist of three kind of manipulations. Firstly it will be manipulation of the on-site value for the added oxygen atoms (Addition of Oxygen will be

simulated by addition of another carbon site to the structure, effectively adding another pi-electron). Secondly it will be by removal of Hydroxide sites and thirdly by manipulation of the on-site values of Hydroxide. These manipulations may be carried out separately.

C. Test 1 para-NPG with added oxygen

The first test will be manipulation of the system 'PS4O'. The basis structure is para-NPG where 4 oxygen sites have been added, two on each benzene ring (see Appendix A, Fig. 20, no. 5). Practically carbon atoms added instead of oxygen. They will be manipulated by lowering their on-site potential. This is to simulate adding oxygen. In Fig. 14 an overview of the structure can be seen along with the band structure for the system, obtained through DFT and TBtrans. Following, in Fig. 15 band plots obtained with the developed program

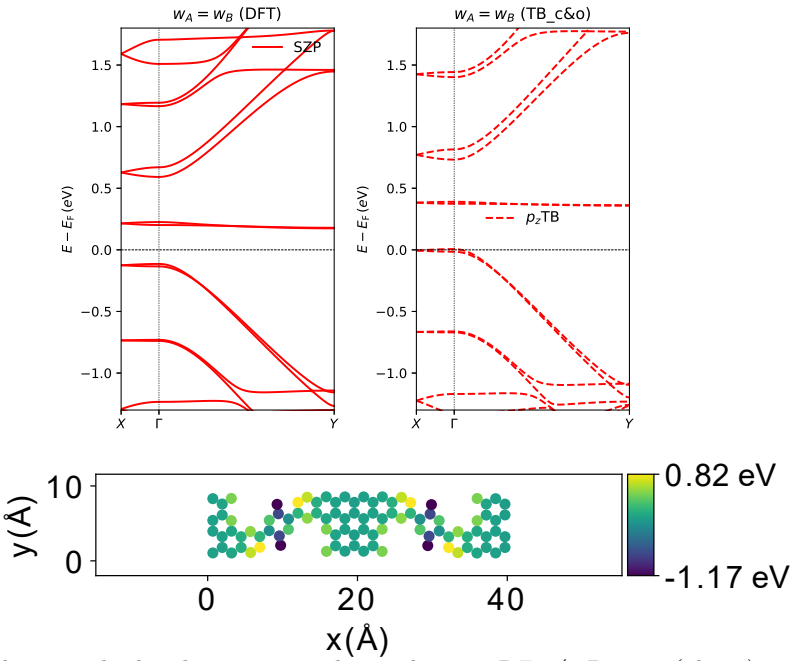


Figure 14: *Figure showing the band structures obtained using DFT/TBtrans (above) and a potential map of the system (below)*

is shown. Firstly a band plot without any changes to the on-site potential and next to it a band plot where the on-site potential of the added sites have been changed. as seen Section VI C the band structure produced does not give the right result right away. This is because that the script assumes the same potential everywhere in the system. But as seen on the potential map in Fig. 14 especially the potential of the added sites (dark blue) are lower than the rest of the system. To compensate for this, the on-site potential values

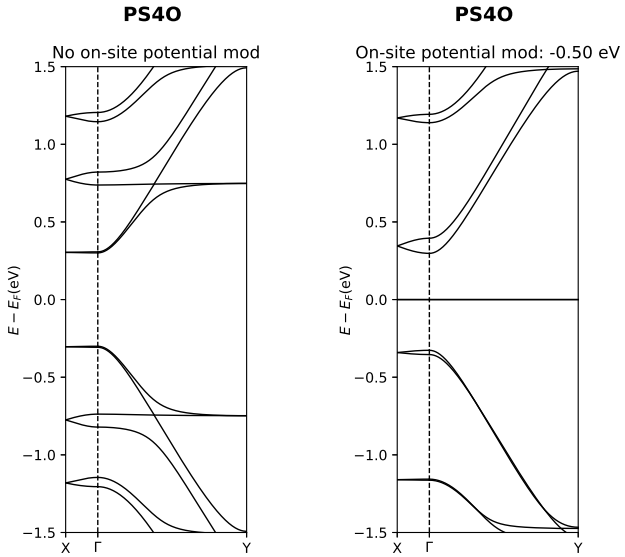


Figure 15: *Figure showing band structures obtained using developed program. One with no on-site potential mods (left) and one with the on-site potential changed to* −0.500 eV *(right).*

of those specific sites have been changed by −0.500 eV in the script (Section VI C) and the result is much closer to that in Fig. 14.

D. Test 2 para-NPG with added hydroxide

Next test will be with the 'PS4OH' system. Again the basis structure is para-NPG, but instead of four oxygen atoms added it will by hydroxide group (see Appendix A, Fig. 20, no. 6). Here the test is to show the difference between removing a OH site entirely (effectively the hydrogen in hydroxide removes the extra pi-electron in oxygen from the system, which can be simulated by removing the atoms entirely) and lowering the on-site potential of the oxygen in the OH group significantly in relation to the rest of the system. Again the results from DTF/Tbtrans is shown first in Fig. 16 and afterwards results from developed programs. Multiple on-site potentials as well as removal of the specific atomic sites were tested to see which one came closest to that of the band structures in Fig. 16. The removal was done by removing specific coordinates in the given files before it was run through the script. A seen in Section VI D the effect of removing the sites gives a plot, somewhat in agreement with Fig. 16. However the lowest and highest bands does not show. The two other plots in Fig. 17 show how gradually changing the on-site potential gets better and better agreement with DFT/TBtrans. A on-site potential of −2.50 eV proved to be a bit too high, though still showing good agreement. Lowering the potential to −2.00 eV which

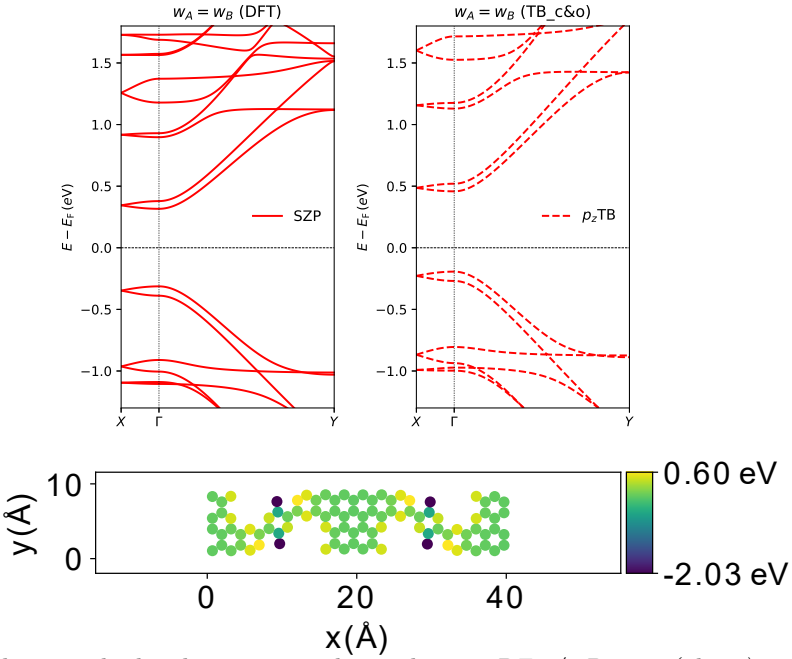


Figure 16: *Figure showing the band structures obtained using DFT/TBtrans (above) and a potential map of the system (below)*

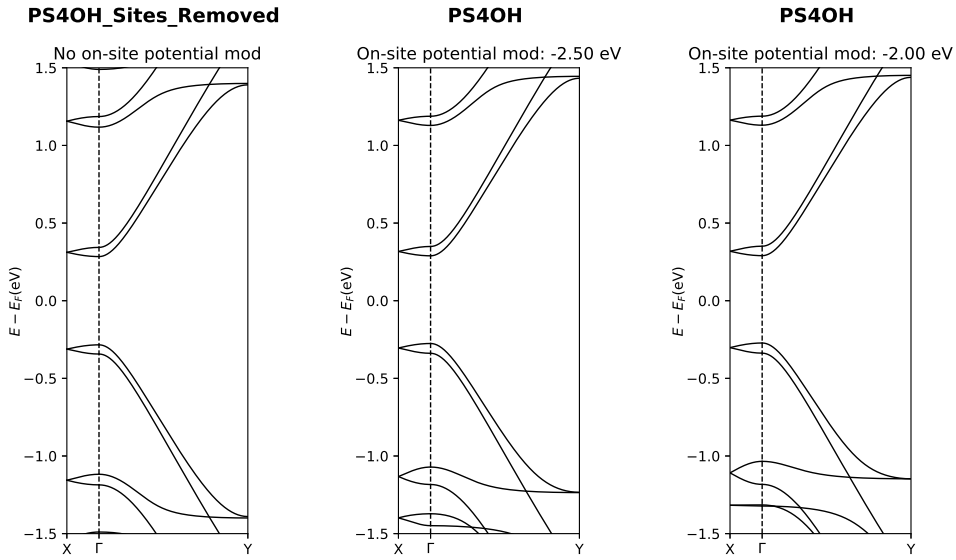


Figure 17: *Figure showing band structures obtained using developed program. Left: band plot where the added sites have been removed, centre: band plot where the on-site potential has been changed by* -2.50 eV*, right: band plot where the on-site potential has been changed by* -2.00 eV*.*

is the value similar to that given by the lowest potential in the potential map Fig. 16, yielded a result that is pretty much spot on, considering the valence bands, when compared with DFT/TBtrans.

E. Test 3

F. Test 4

G. Test 5

H. Test 6

ACKNOWLEDGMENTS

The authors would like to thank...

---

[1] N. R. Papior, sisl: v0.9.5 (2018).

[2] G. Calogero, N. R. Papior, B. Kretz, A. Garcia-Lekue, T. Frederiksen, and M. Brandbyge, Electron Transport in Nanoporous Graphene: Probing the Talbot Effect, Nano Letters **19**, 576 (2019).

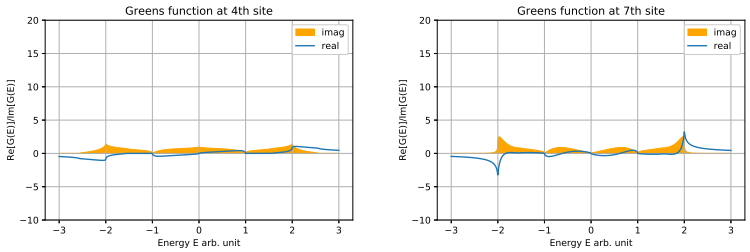
LIST OF FIGURES

LIST OF TABLES

LISTINGS

Appendices

Appendix A: Additional figures



(a) *Figure showing a plot of the Green's function at the 4th site*

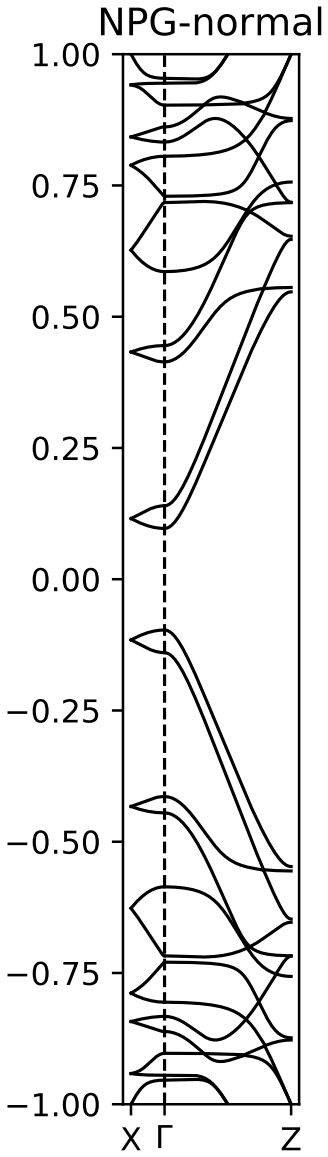
(b) *Figure showing a plot of the Green's function at the 7th site*

Figure 18: *Two plots showing how the Green's function changes as the site is changed. The 4th and 7th sites are corresponding to atoms of those indices (4, 7) in Fig. 6a. Note how the LDOS changes (imaginary part) for the different sites.*

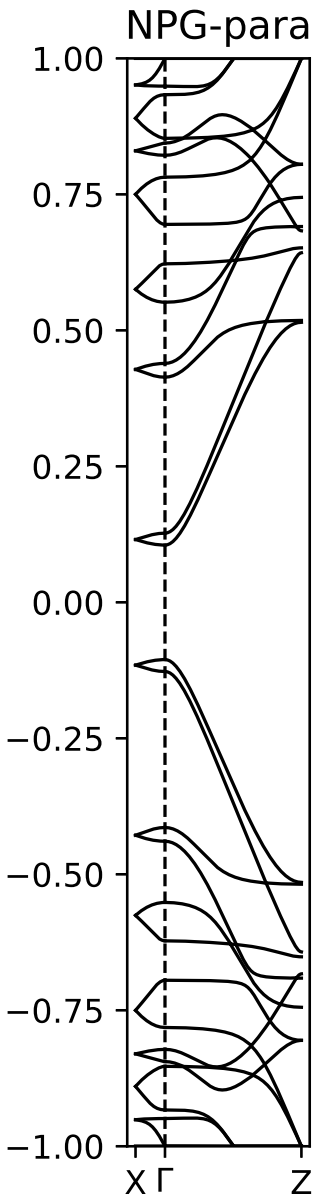
```
41          p = 0
42      if o == 'y':
43          charr = np.array(['PS4O', 'PS4OH', 'MS2O', 'MS2OH', 'MA2O', 'MA2OH'])
44          for i in range(charr.shape[0]):
45              print('{}: {}'.format(i+1, charr[i]))
46          struct = int(input('Choose structure: '))
47          if struct >= 3:
```

Listing 8: *Function creating the hopping matrices between two sets of coordinates*

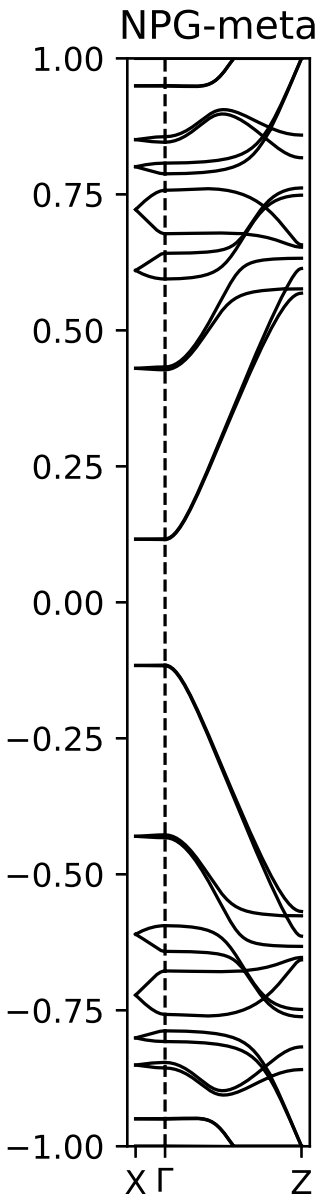
```
245 def PeriodicHamiltonian(xyz, UY, i):
246     h = Onsite(xyz=xyz, Vppi=-1)
247     V = Hop(xyz=xyz, xyz1=xyz + np.array([0, UY, 0]), Vppi=-1)
248     # plt.imshow(V.real)
249     # plt.show()
250     print('Number of hopping elements: {}'.format(np.sum(np.abs(V))))
251     Ham = h + V * np.exp(1j * i) + np.transpose(V) * np.exp(-1j * i)
```



(a) *Plot showing the band structure in the energy range -1 to 1 for NPG with normal bridges between symmetry points X and Z with respect to* Γ



(b) *Plot showing the band structure in the energy range -1 to 1 for NPG with para bridges between symmetry points X and Z with respect to* Γ



(c) *Plot showing the band structure in the energy range -1 to 1 for NPG with meta bridges between symmetry points X and Z with respect to* Γ

Figure 19: *Figure showing para, meta and normal NPG band structures*

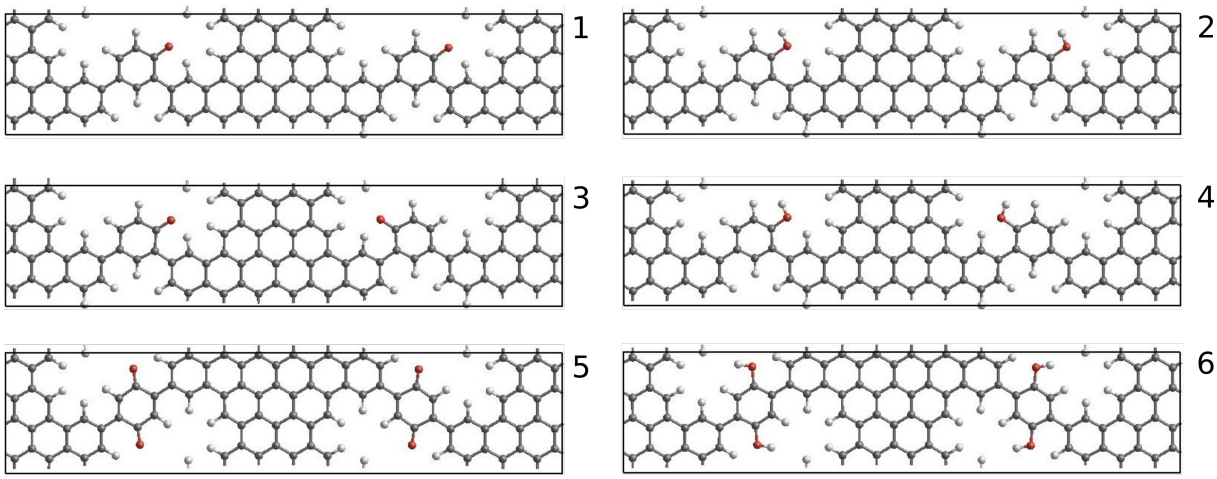


Figure 20: *Figure showing all the structures stated in Table I.*

```
252     return Ham
```

Listing 9: *Code piece showing how the periodic hamiltonian, shifted in the transverse direction i created usind the given unit vector in the y direction.*

```
234 def Transmission(GammaL, GammaR, GD, En):
235     T = np.zeros(En.shape[0], dtype=complex)
236     bar = Bar('Calculating Transmission   ', max=En.shape[0])
237     for i in range(En.shape[0]):
238         T[i] = np.trace((GammaR["GammaR{:d}".format(i)] @ GD["GD{:d}".format(
239             i)] @ GammaL["GammaL{:d}".format(i)] @ GD["GD{:d}".format(i)].conj().transpose(
240         bar.next()
241     bar.finish()
242     return T
```

Listing 10: *Code piece showing the line that calculates the transmission*