

Quantum Transport in Nanoporous Graphene

Bachelor defense

CHRISTOFFER VENDELBO SØRENSEN (s163965)



JUNE 25, 2019

Outline

Introduction

Project aim and nanoporous graphene

Tight Binding

π -orbitals, π -electrons and the TB approximation

The Hamiltonian

Onsites, hops and the full TB Hamiltonian

Green's functions and recursion

Green's matrix, recursion and LDOS

Transmission

Device Green's functions, left/right geometry, rate matrices and spectral functions.

Transmission in 2D

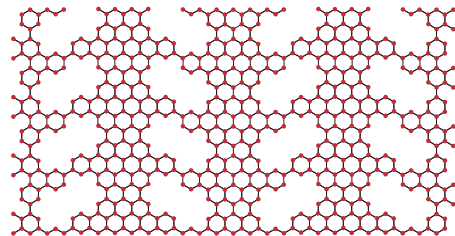
Exploring GNR bridges

Para-O₄-NPG, Para-(OH)₄-NPG, Meta-O₂-NPG, Meta-(OH)₂-NPG

“Development of tight-binding routines in Python in order to understand electron transport in novel nanoporous graphene devices (NPGs)”

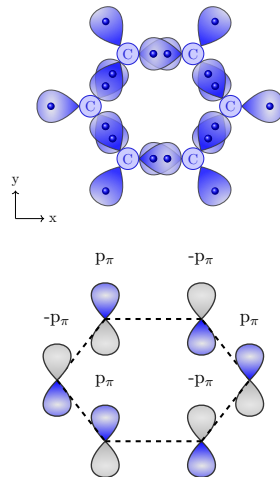
Nanoporous graphene

- ▶ Planar graphene sheets
- ▶ Periodically removed atoms
- ▶ Ribbons and bridges
- ▶ Ballistic electron movement
- ▶ Potential for controlling currents on nanoscale



π -orbitals and π -electrons

- ▶ In plane electrons are bound
- ▶ 1 p_z -electron per site
- ▶ “Tightly bound” hops between sites



TB approximation

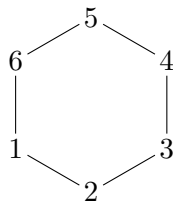
- ▶ Electrons tightly bound to sites
- ▶ Hops with potential
- ▶ Average electron energy on site
- ▶ The Hamiltonian is a hop matrix

$$V_{pp\pi} = \langle \phi_{\pi}(m) | \hat{\mathbf{H}} | \phi_{\pi}(n) \rangle$$

$$\epsilon_0 = \langle \phi_{\pi}(i) | \hat{\mathbf{H}} | \phi_{\pi}(i) \rangle$$

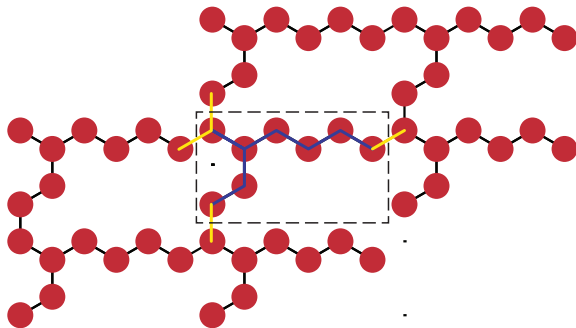
Hamiltonian for benzene

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



Creating the first Hamiltonian

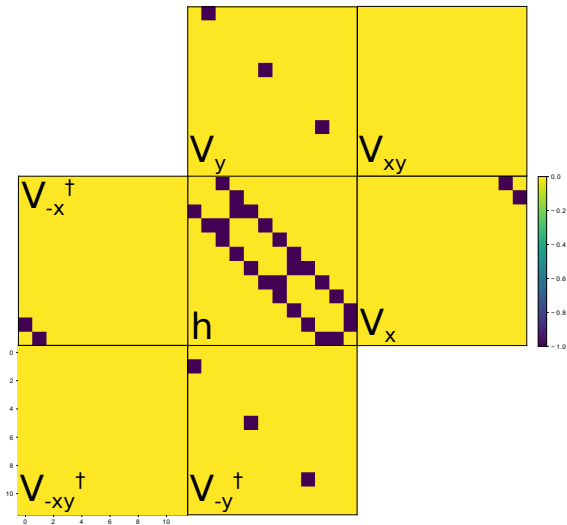
- ▶ Atom coordinates
- ▶ Interatomic distances
- ▶ Applying potential



```
33 h = np.zeros((xyz.shape[0], xyz.shape[0])) # Empty matrix
34 for i in range(xyz.shape[0]): # Take an atomic coordinate
35     for j in range(xyz.shape[0]): # Take another atomic coordinate
36         h[i, j] = LA.norm(np.subtract(xyz[i], xyz[j])) # Measure distances
37 h = np.where(h < 1.6, Vppi, 0) # Replace distances under 1.6 angstrom with Vppi
38 h = np.subtract(h, Vppi * np.identity(xyz.shape[0])) # Remove the diagonal
```


Hopping matrices

- Shift by lattice vector
- Resulting matrices:
 $\mathbf{h}_0, \mathbf{V}, \mathbf{V}^\dagger$



Full Hamiltonian and first band plots

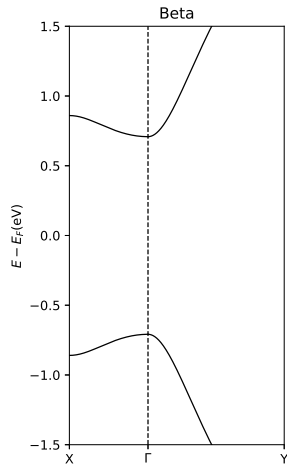
$$\mathbf{H}(k_x, k_y)\phi_k = \epsilon_n(k_x, k_y)\phi_k$$
$$\mathbf{H}(k_x, k_y) = \mathbf{h}_0 + (\mathbf{V}_x e^{-ik_x} + \mathbf{V}_x^\dagger e^{ik_x} + \mathbf{V}_y e^{-ik_y} + \mathbf{V}_y^\dagger e^{ik_y} \\ + \mathbf{V}_{xy} e^{-ik_x} e^{-ik_y} + \mathbf{V}_{xy}^\dagger e^{ik_x} e^{ik_y})$$

```
73 Ham = Ham + (V1 * np.exp(-1.0j * x) # Onsite hops and hops in x
74           + np.transpose(V1) * np.exp(1.0j * x) # Hops in -x
75           + V2 * np.exp(-1.0j * y) # Hops in y
76           + np.transpose(V2) * np.exp(1.0j * y) # Hops in -y
77           + V3 * np.exp(-1.0j * x) * np.exp(-1.0j * y) # Hops in both x and y
78           + np.transpose(V3) * np.exp(1.0j * x) * np.exp(1.0j * y)) # Hops in both -x and -y
79 e = LA.eigh(Ham)[0] # Eigen energies from the Hamiltonian
80 v = LA.eigh(Ham)[1] # Eigen vectors from the Hamiltonian
```

Full Hamiltonian and first band plots

$$\text{X: } \mathbf{H}_X = \mathbf{h}_0 + (\mathbf{V}_x e^{ik_x} + \mathbf{V}_x^\dagger e^{-ik_x} + \mathbf{V}_y + \mathbf{V}_y^\dagger + \mathbf{V}_{xy} e^{ik_x} + \mathbf{V}_{xy}^\dagger e^{-ik_x})$$

$$\text{Y: } \mathbf{H}_Y = \mathbf{h}_0 + (\mathbf{V}_x + \mathbf{V}_x^\dagger + \mathbf{V}_y e^{-ik_y} + \mathbf{V}_y^\dagger e^{ik_y} + \mathbf{V}_{xy} e^{-ik_y} + \mathbf{V}_{xy}^\dagger e^{ik_y})$$



- Solution to the Scödinger Equation
- Propagator

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

↓

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

► Semi-infinite chain

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

$$\begin{aligned} \mathbf{G}_{00}(z) &= \left[(z - \varepsilon') - \mathbf{V}(z\mathbf{1} - \mathbf{H}_c)\mathbf{V}^\dagger \right]^{-1} \\ &= (z - \varepsilon' - \Sigma(z))^{-1} \end{aligned}$$

where

$$z = E + i\eta$$

$$a_0 = \mathbf{V}^\dagger, \quad b_0 = \mathbf{V}$$

$$e_{s0} = \mathbf{h}_s, \quad e_0 = \mathbf{h}$$

in loop:

$$a_1 = a_0 \times g_0 \times a_0$$

$$b_1 = b_0 \times g_0 \times b_0$$

$$e_1 = e_0 + a_0 \times g_0 \times b_0 + b_0 \times g_0 \times a_0$$

$$e_{1s} = e_{0s} + a_0 \times g_0 \times b_0$$

$$g_1 = (z - e_1)^{-1}$$

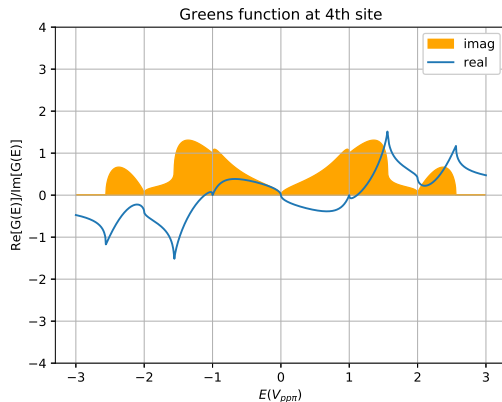
$$\Sigma_R = e_s - h$$

$$\Sigma_L = e - h - \Sigma_R$$

$$\mathbf{G}_{00} = (z - e_s)^{-1}$$

Recursion

```
92 while np.max(np.abs(a0)) > 1e-6: # Loop with hop matrix as threshold
93     ag = a0 @ g0 # Product defined here and used multiple times
94     a1 = ag @ a0 # New hop matrix (transposed)
95     bg = b0 @ g0 # Product defined here and used multiple times
96     b1 = bg @ b0 # New hop matrix
97     e1 = e0 + ag @ b0 + bg @ a0 # New onsite for "other cells"
98     es1 = es0 + ag @ b0 # New onsite
99     g1 = LA.inv(z - e1) # New Green's function
100     a0 = a1 # Overwrite old variable
101     b0 = b1 # Overwrite old variable
102     e0 = e1 # Overwrite old variable
103     es0 = es1 # Overwrite old variable
104     g0 = g1 # Overwrite old variable
105     q = q + 1 # Counter (for diagnostic purposes)
106 e, es = e0, es0 # Define the onsite Hamiltonians
107 SelfER = es - h # Self-energy from the right
108 SelfEL = e - h - SelfER # Self-energy from the left
109 G00 = LA.inv(z - es) # Green's functions
```



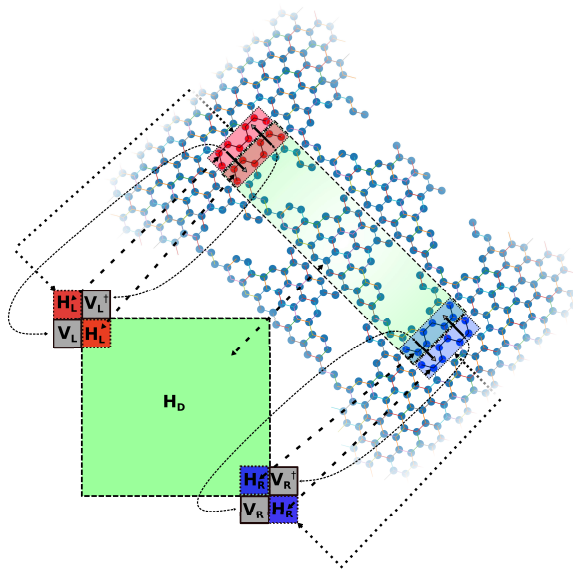
```

64  G00 = np.zeros((En.shape[0]), dtype=complex) # Empty data matrix for Green's functions
65  for i in range(En.shape[0]): # Loop iterating over energies
66      G, SelfER, SelfEL = RecursionRoutine(En[i], h, V, eta) # Invoking the RecursionRoutine
67      G = np.diag(G) # The Green's functions for each site is in the diagonal of the G matrix
68      G00[i] = G[4] # Chosen Green's function (here the 4th site)

```

Transmission is the probability of an electron being transported through a specific region for a specific range of energies.

Translating from system to matrices



Introduction
○○

Tight Binding
○○○

The Hamiltonian
○○○

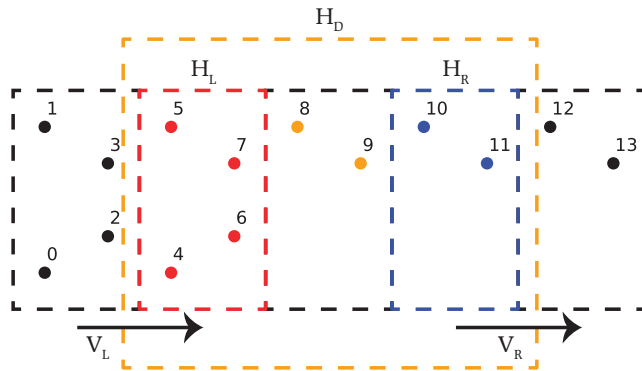
Green's functions and recursion
○○○

Transmission
○○●○○
○○○

Exploring GNR bridges
○○○○○○

Questions
○

The left and right self-energy



```
210     for i in En: # Iteration over each energy
211         gl, scrap, SEL = RecursionRoutine(i, HL, VL, eta=eta) # From the left
212         gr, SER, scrap = RecursionRoutine(i, HR, VR, eta=eta) # From the right
```

Getting Transmission

$$\mathbf{G}_D = [\mathbf{1}(E + i\eta) - \mathbf{H}_D - \mathbf{\Sigma}_L(E) - \mathbf{\Sigma}_R(E)]^{-1}$$
$$\mathbf{\Gamma}_{L,R} = i \left(\mathbf{\Sigma}_{L,R} - \mathbf{\Sigma}_{L,R}^\dagger \right)$$

```
225 GD["GD{:d}".format(q)] = scp.linalg.inv( # Device Green's functions are saved
226     scp.identity(HD.shape[0]) * (i + eta) - HD - SEL - SER) # in a dictionary.
227 GammaL["GammaL{:d}".format(q)] = 1j * (SEL - SEL.conj().transpose()) # Rate matrices are
228 GammaR["GammaR{:d}".format(q)] = 1j * (SER - SER.conj().transpose()) # likewise saved.
```

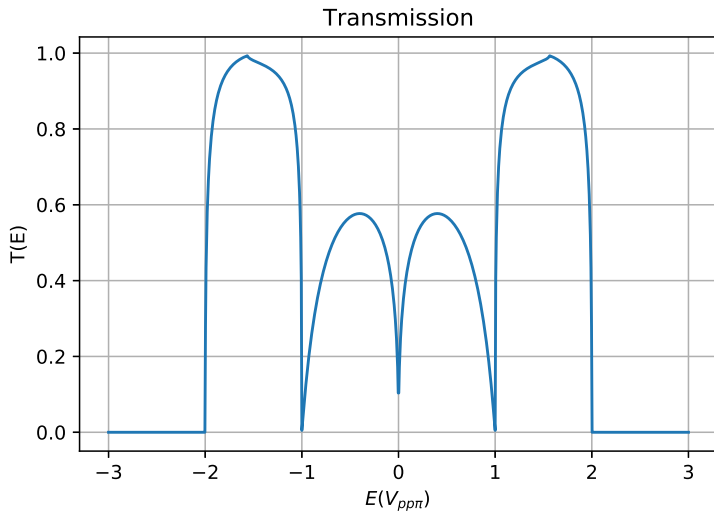
Getting Transmission

$$T(E) = \text{Tr} \left[\Gamma_R \mathbf{G}_D \Gamma_L \mathbf{G}^\dagger \right] (E)$$

```
240     for i in range(En.shape[0]): # Iteration for every energy point.
241         T[i] = np.trace((GammaR["GammaR{:d}".format(i)] @ GD["GD{:d}".format(
242             i)] @ GammaL["GammaL{:d}".format(i)] @ GD["GD{:d}".format(i)].conj(
243             ).transpose()).todense()) # Calculate transmission (equation V.9).
```

Getting Transmission

- Text1
- Text2
- Text3



Introduction
oo

Tight Binding
ooo

The Hamiltonian
ooo

Green's functions and recursion
ooo

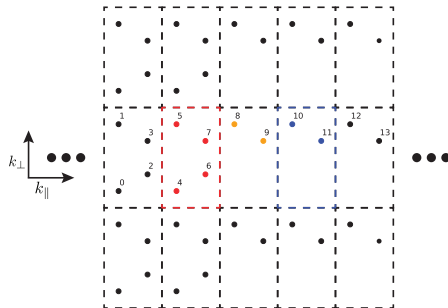
Transmission
ooo●
ooo

Exploring GNR bridges
oooooo

Questions
o

Transmission in 2D

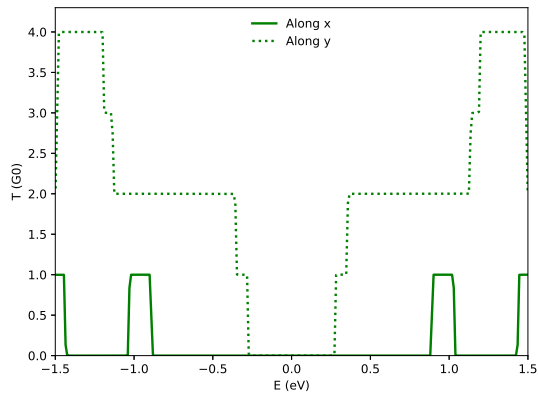
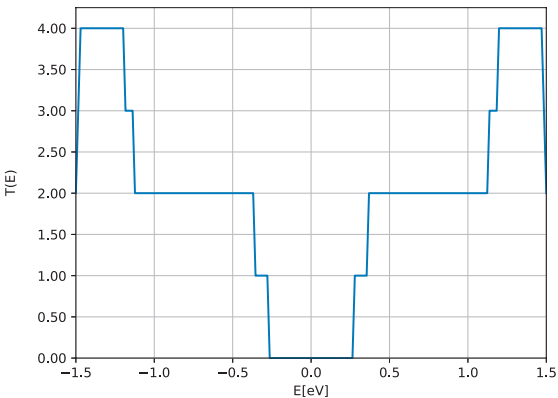
$$\mathbf{H} = \mathbf{h} + \mathbf{V}e^{ik_{\perp}} + \mathbf{V}^{\dagger}e^{i(-k_{\perp})}$$



```
250 h, p = Onsite(xyz=xyz, Vppi=-1, f=1) # Calculate onsite hops
251 V = Hop(xyz=xyz, xyz1=xyz + np.array([0, UY, 0]), Vppi=-1) # Calculate hops
252 print('Number of hopping elements: {}'.format(np.sum(np.abs(V)))) # Number of hops
253 Ham = h + V * np.exp(1j * i) + np.transpose(V) * np.exp(-1j * i) # Hamiltonian for ith kp
```

Code validity

$$k = \frac{\pi}{2}$$



Introduction
○○

Tight Binding
○○○

The Hamiltonian
○○○

Green's functions and recursion
○○○

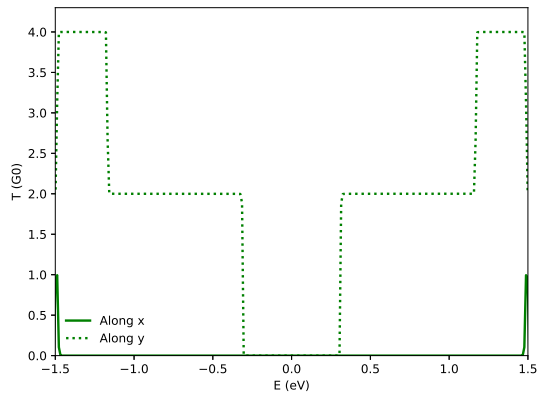
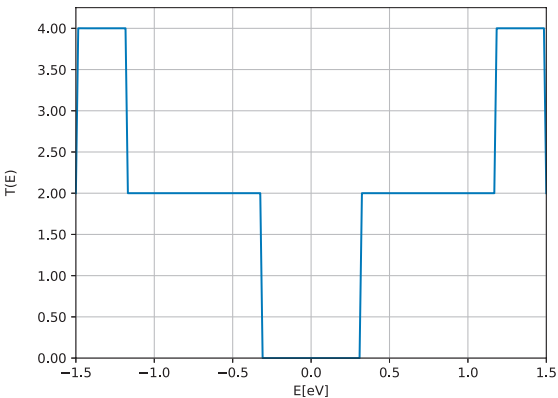
Transmission
○○○○
●○○

Exploring GNR bridges
○○○○○

Questions
○

Code validity

$$k = \pi$$



Introduction
○○

Tight Binding
○○○

The Hamiltonian
○○○

Green's functions and recursion
○○○

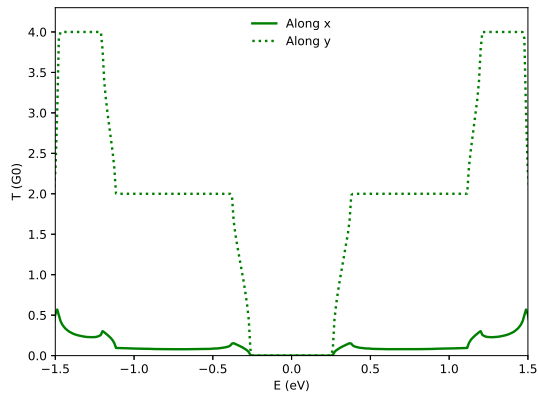
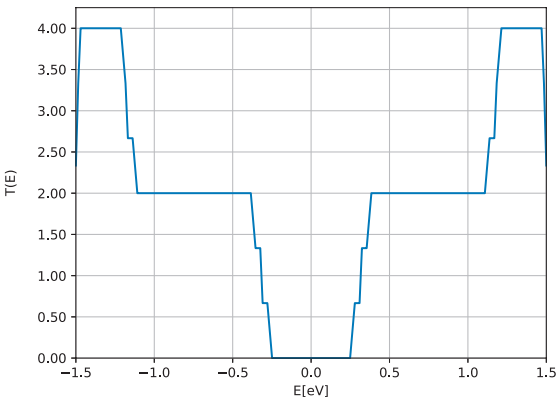
Transmission
○○○○
●○○

Exploring GNR bridges
○○○○○○

Questions
○

Code validity

$$k = \text{AVG}\left(0, \frac{\pi}{2}, \pi\right)$$



Introduction
○○

Tight Binding
○○○

The Hamiltonian
○○○

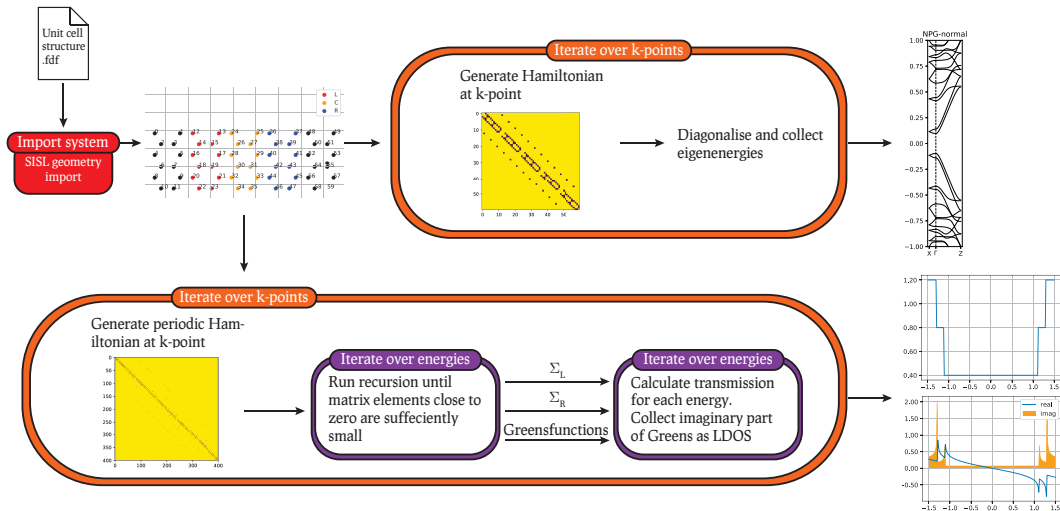
Green's functions and recursion
○○○

Transmission
○○○○
●●○

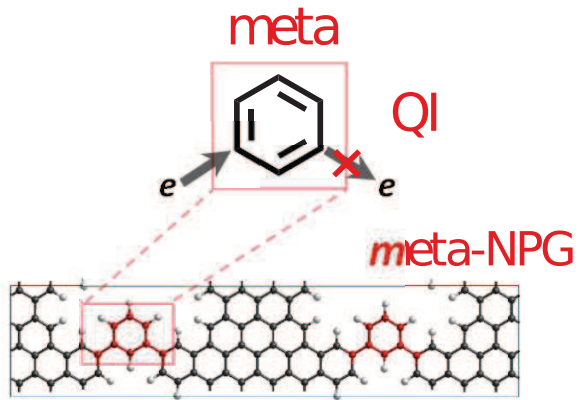
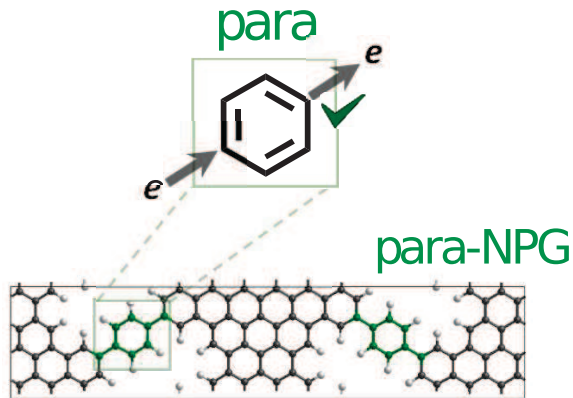
Exploring GNR bridges
○○○○○

Questions
○

Summary of code structure

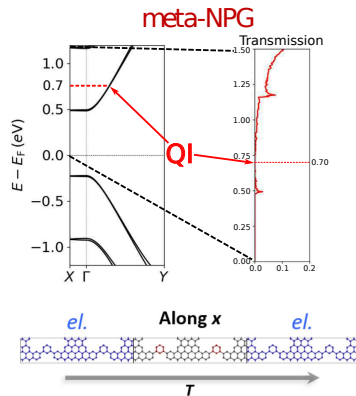
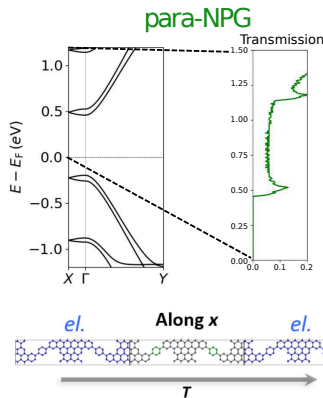


Para and meta bridges



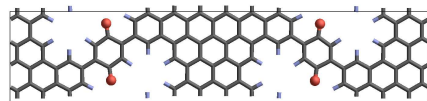
Para and meta bridges

- ▶ Path length difference
- ▶ Quantum interference
- ▶ Transmission
- ▶ coupling/decoupling of GNR

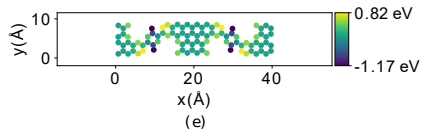
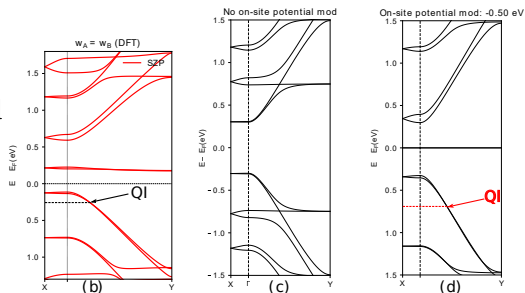


Para-O₄-NPG

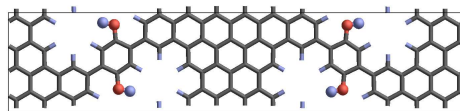
- Functionalisation with oxygen
- Quantum interference at valence band
- Reproducing DFT result
- Decoupling of GNR
- Bands at Fermi level



(a)

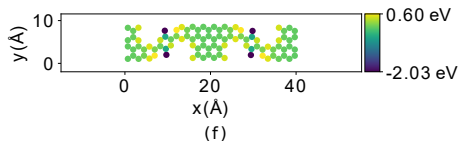
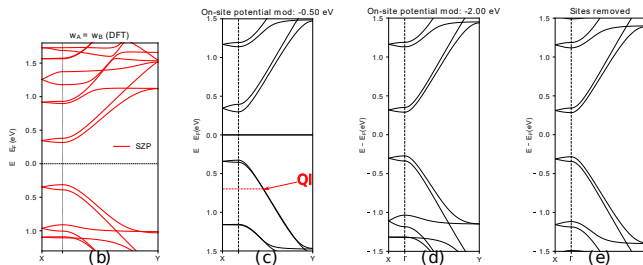


Para-(OH)₄-NPG



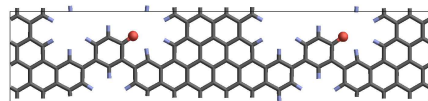
(a)

- Hydrogenation of oxygen
- Band splitting of DFT
- GNR coupling and resemblance with Para-NPG
- Reproducing DFT results

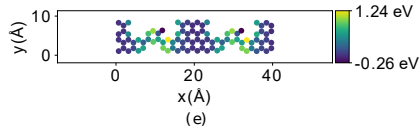
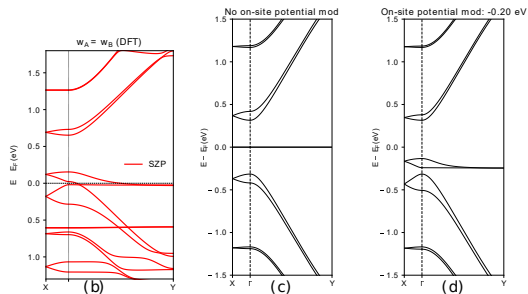


Meta-O₂-NPG

- Split in the valence for DFT
- Similarity in valence/conduction difference for meta/para
- Trying different potentials
- Flat bands at Fermi level

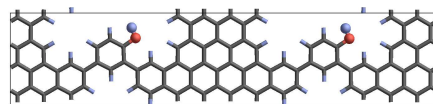


(a)

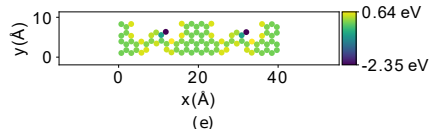
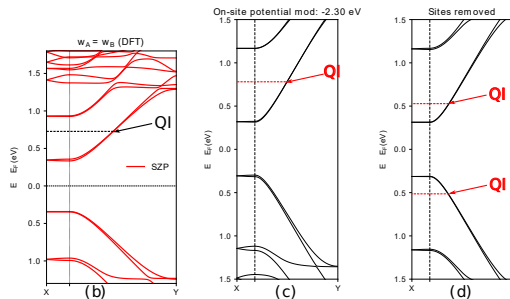


(e)

Meta-(OH)₂-NPG



(a)



“Development of tight-binding routines in Python in order to understand electron transport in novel nanoporous graphene devices (NPGs)”

Questions

CHRISTOFFER VENDELBO SØRENSEN (s163965)



JUNE 25, 2019

Introduction
oo

Tight Binding
ooo

The Hamiltonian
ooo

Green's functions and recursion
ooo

Transmission
oooo
ooo

Exploring GNR bridges
oooooo

Questions
●

► Appendix