

---

# **2deg\_QH\_SC**

***Release 3.0***

**A. David, J. S. Meyer, M. Houzet**

**Oct 10, 2023**



# CONTENTS

<b>1</b>	<b>Installation and Usage</b>	<b>3</b>
1.1	Miniconda, GitHub repository and conda environment . . . . .	3
1.2	Running the scripts by using the command line . . . . .	3
1.3	Running the scripts by using PyCharm . . . . .	4
1.4	Running the scripts by using Visual Studio Code . . . . .	4
1.5	Updating the documentation . . . . .	5
<b>2</b>	<b>Main Scripts</b>	<b>7</b>
2.1	calculations.py . . . . .	7
2.2	manuscript_figures.py . . . . .	8
<b>3</b>	<b>Modules</b>	<b>9</b>
3.1	functions.py . . . . .	9
3.2	system.py . . . . .	14
3.3	utils.py . . . . .	16
<b>4</b>	<b>Indices and tables</b>	<b>23</b>
<b>5</b>	<b>License</b>	<b>25</b>
	<b>Python Module Index</b>	<b>27</b>
	<b>Index</b>	<b>29</b>



Documentation of the code used in the paper “Geometrical effects on the downstream conductance in quantum-Hall–superconductor hybrid systems” available on [arXiv](#) and published in [Phys. Rev. B](#). The code is available in a [Github Repository](#).

Check the [Installation and Usage](#) section to install and use the code. The [Main Scripts](#) section includes the script *calculations.py* which can be used to run the different calculations and the script *manuscript\_figures.py* that generates the manuscript figures. The [Modules](#) section describes the different modules used to run the calculations.



## INSTALLATION AND USAGE

Here we explain how to run the scripts. After following the steps below try to run some calculations with the script *calculations.py* or reproduce the manuscript figures with the script *manuscript\_figures.py*.

### 1.1 Miniconda, GitHub repository and conda environment

Install [miniconda](#) , clone the [repository](#) and cd into the root directory *2deg\_QH-SC-main* after unzipping. Then create the Conda environment that contains all dependencies with

```
conda env create -f environment.yml
```

You can now use this environment to run the scripts. Below is detailed how to run the scripts using the command line, Pycharm or Visual Studio Code.

### 1.2 Running the scripts by using the command line

To run a script from the terminal use the following command line:

```
<path_to_python_exe> <path_to_python_script>
```

The python executable should be located at :

~/opt/miniconda3/envs/2deg\_QH-SC/bin/python (macOS)

~/miniconda3/envs/2deg\_QH-SC/bin/python (Linux)

~/miniconda3/envs/2deg\_QH-SC/python (Windows)

For example, if you are on macOS and you want to run the script *calculations.py* (while being in the root directory of the repo), use

```
~/opt/miniconda3/envs/2deg_QH-SC/bin/python calculations.py
```

**Note:** You can use a global shell variable to create a shortcut to the Python executable path. For that, open a terminal and modify the bash configuration file located in your HOME directory (the directory in which you are when you open the terminal) (*.bashrc*, *.bash\_profile*, or *.profile*). For example on macOS

```
nano .bash_profile
```

Add the following in the file

```
export mypython=~/.opt/miniconda3/envs/2deg_QH-SC/bin/python
```

and save it with Ctrl+X and Y and Enter. Then close the terminal and open a new one to make the modification effective. You can now use the variable `$mypython` for the path such that the above example reads

```
$mypython calculations.py
```

---

## 1.3 Running the scripts by using PyCharm

With **PyCharm** you can follow these steps :

1. Launch PyCharm and choose **Create New Project**
2. **Locate the project** at the root directory *2deg\_QH-SC-main*.
3. Mark **Existing interpreter** (or **Previously configured interpreter**) and click on the selection icon  
...
4. Select **Conda Environment** and choose the location of the python executable. It should be located at :  
    ~/.opt/miniconda3/envs/2deg\_QH-SC/bin/python (macOS)  
    ~/miniconda3/envs/2deg\_QH-SC/bin/python (Linux)  
    ~/miniconda3/envs/2deg\_QH-SC/python (Windows)
5. Click on **Ok** then on **Create** and select **Create from existing sources**
6. You are ready to run the scripts!
7. (Optional) You can see progress bars during the calculations by activating the **Emulate terminal in output console** option. For that, got to **Run->Edit Configurations** and check the option.

## 1.4 Running the scripts by using Visual Studio Code

With **Visual Studio Code** you can follow these steps :

1. Launch Visual Studio Code and install the *Python* extension if it's not done yet.
2. From the main page choose **Open...** , select the root directory *2deg\_QH-SC-main*, and click on **Yes, I trust the authors**
3. Open the *Command Palette* with Ctrl+Shift+P, search **Python: Select Interpreter** and choose the one associated to the '2deg\_QH-SC' environment.
4. You are ready to run the scripts!



## 1.5 Updating the documentation

The *documentation* directory contains a pdf and a local html version of the documentation. They can respectively be found at *documentation/build/latex/2deg\_gh-sc.pdf* and *documentation/build/html/index.html*. You can update them by using

```
conda activate 2deg_QH-SC
cd documentation
make html
make latexpdf
```

---

**Note:** When you compile the documentation it runs the python scripts so make sure the calculations are commented before using `make html` or `make latexpdf`.

---



## MAIN SCRIPTS

The scripts present in the *main\_code* directory are described in this section.

### 2.1 calculations.py

Calculations.

This script contains different sections, each of which generates the data and the plot of a calculation. Comment the quotation marks by using hashtags to run the calculations.

Contents:

- Visualization
  - Kwant system
  - Density  $u^2 - v^2$
- Energy Spectrum
  - Tight-binding spectrum
  - Microscopic spectrum
  - Spectrum comparison
- Momentum at the Fermi level  $k_0$ 
  - $k_0$  v.s.  $\nu$
  - $k_0$  v.s.  $Z$  at various fillings
- Andreev Transmission and Hole Probability
  - $\tau$  v.s.  $\theta_{qh}$  at various fillings
  - $\tau$  v.s.  $\theta_{sc}$  at various fillings
  - $\tau$  v.s.  $\mu_{qh}/\Delta$  at various fillings
  - $f_{h,p}$  v.s.  $Z$  at various fillings
- Downstream conductance
  - Conductance comparison
- Track states
  - $\nu_{crit}$  v.s.  $\mu_{qh}/\delta$
  - Asymptotic  $\nu_{crit}$  v.s.  $\mu_{sc}/\mu_{qh}$

- Asymptotic  $\nu_{\text{crit}}$  v.s.  $Z$
- Finite-temperature
  - Momentum difference v.s. energy
  - $\tau$  v.s. energy
  - Finite-temperature tight-binding conductance v.s.  $L$
  - Finite-temperature tight-binding conductance v.s.  $L$  at various temperatures

The resulting data and plots are saved in the ‘files’ directory.

## 2.2 manuscript\_figures.py

This script generates the figures as used in the manuscript.

To re-compute the data use the option `from_data=False` in the plot functions. Due to the option `fig_name` the plots are saved in the ‘figures’ directory.

## MODULES

The different modules are described in this section.

### 3.1 functions.py

Definition of the functions required for the calculations.

`modules.functions.U(n, z)`

Definition of the parabolic cylinder function  $U(n, z)$  as defined in Abramowitz & Stegun book.

**Parameters**

- **n** (*float*) – First argument of the parabolic cylinder function.
- **z** (*float*) – Second argument of the parabolic cylinder function.

**Returns** The value of the parabolic cylinder function  $U(n, z)$ .

**Return type** float

`modules.functions.U_m(x, E, k, m_qh, mu_qh, omega)`

Hole-like solution in QH region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_qh** (*float*) – Effective mass in the QH region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **omega** (*float*) – Cyclotron frequency.

**Returns** The value of the parabolic cylinder function associated to holes.

**Return type** float

`modules.functions.U_p(x, E, k, m_qh, mu_qh, omega)`

Electron-like solution in QH region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.

- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_qh** (*float*) – Effective mass in the QH region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **omega** (*float*) – Cyclotron frequency.

**Returns** The value of the parabolic cylinder function associated to electrons.

**Return type** *float*

`modules.functions.chi_m(x, E, k, m_qh, mu_qh, nu)`

Hole-like wave function in QH region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_qh** (*float*) – Effective mass in the QH region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **nu** (*float*) – Filling factor.

**Returns** The value of hole-like wave function in QH region.

**Return type** *float*

`modules.functions.chi_p(x, E, k, m_qh, mu_qh, nu)`

Electron-like wave function in QH region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_qh** (*float*) – Effective mass in the QH region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **nu** (*float*) – Filling factor.

**Returns** The value of electron-like wave function in QH region.

**Return type** *float*

`modules.functions.effective_tau(tau_0, L_b, v_b, mu_b, delta_b, phi_b)`

Compute the effective conversion probability at a QH-SS corner.

The value `tau_0` is the one obtained from the microscopic model while the parameters labelled with ‘\_b’ correspond to the effective barrier.

**Parameters**

- **tau\_0** (*str*) – Hole probability computed with the microscopic model.
- **L\_b** (*str*) – Length of the barrier.
- **v\_b** (*str*) – Velocity in the barrier.
- **mu\_b** (*str*) – Chemical potential in the barrier.

- **delta\_b** (*str*) – Superconducting gap of the barrier.
- **phi\_b** (*str*) – Superconducting phase of the barrier.

`modules.functions.fermi_momenta(m_qh, m_sc, mu_qh, mu_sc, nu, delta, V_barrier)`

Positive momentum solutions of the secular equation  $f(E, k) = 0$  at the Fermi level (i.e. at  $E = 0$ ).

**Parameters**

- **m\_qh** (*float*) – Effective mass in the QH region.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **nu** (*float*) – Filling factor.
- **delta** (*float*) – Superconducting gap.
- **V\_barrier** (*float*) – Height of the delta-potential barrier.

**Returns** The positive solutions of the equation  $f(E=0, k) = 0$ .

**Return type** list

`modules.functions.hole_probability(m_qh, m_sc, mu_qh, mu_sc, nu, delta, V_barrier)`

Compute the hole content  $f_h^+$  at the quasi-electron crossing, i.e., at  $k = -k_0$ .

**Parameters**

- **m\_qh** (*float*) – Effective mass in the QH region.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **nu** (*float*) – Filling factor.
- **delta** (*float*) – Superconducting gap.
- **V\_barrier** (*float*) – Height of the delta-potential barrier.

**Returns** Hole probability at the quasi-electron crossing.

**Return type** float

`modules.functions.hopping(site1, site2, a, t, mu_qh, nu)`

Define hopping energies in QH and SC regions.

**Parameters**

- **site1** – Kwant site.
- **site2** – Kwant site.
- **a** (*float*) – Lattice spacing.
- **t** (*float*) – Hopping energy at zero field.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **nu** (*float*) – Filling factor.

**Returns** Hopping energy.

**Return type** float

`modules.functions.hopping_qh(site1, site2, a, t, mu_qh, nu)`

Define hopping energy in QH region.

**Parameters**

- **site1** – Kwant site.
- **site2** – Kwant site.
- **a** (*float*) – Lattice spacing.
- **t** (*float*) – Hopping energy at zero field.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **nu** (*float*) – Filling factor.

**Returns** Hopping energy.

**Return type** float

`modules.functions.hopping_sc(site1, site2, t)`

Define hopping energy in SC region.

**Parameters**

- **site1** – Kwant site.
- **site2** – Kwant site.
- **t** (*float*) – Hopping energy at zero field.

**Returns** Hopping energy.

**Return type** float

`modules.functions.kronecker_delta(x)`

Kronecker delta function.

**Parameters** **x** (*float*) – x-coordinate.

**Returns**

1. if  $x=0$  and 0. otherwise.

**Return type** float

`modules.functions.onsite(site, a, t, mu_qh, mu_sc, delta, Z)`

Define onsite energies in QH and SC regions including a delta-potential barrier.

**Parameters**

- **site** – Kwant site.
- **a** (*float*) – Lattice spacing.
- **t** (*float*) – Hopping energy at zero field.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **delta** (*float*) – Superconducting gap.
- **Z** (*float*) – Barrier strength.

**Returns** Onsite energy.

**Return type** float



`modules.functions.onsite_qh(site, t, mu_qh)`

Define onsite energy in QH region.

**Parameters**

- **site** – Kwant site.
- **t** (*float*) – Hopping energy at zero field.
- **mu\_qh** (*float*) – Chemical potential in the QH region.

**Returns** Onsite energy.

**Return type** float

`modules.functions.onsite_sc(site, t, mu_sc, delta)`

Define onsite energy in SC region.

**Parameters**

- **site** – Kwant site.
- **t** (*float*) – Hopping energy at zero field.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **delta** (*float*) – Superconducting gap.

**Returns** Onsite energy.

**Return type** float

`modules.functions.phi_m(x, E, k, m_sc, mu_sc, delta)`

Hole-like wave function in SC region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **delta** (*float*) – Superconducting gap.

**Returns** The value of hole-like wave function in SC region.

**Return type** float

`modules.functions.phi_p(x, E, k, m_sc, mu_sc, delta)`

Electron-like wave function in SC region.

**Parameters**

- **x** (*float*) – x-coordinate.
- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **delta** (*float*) – Superconducting gap.

**Returns** The value of electron-like wave function in SC region.

**Return type** float

`modules.functions.secular_equation(k, E, m_qh, m_sc, mu_qh, mu_sc, omega, delta, V_barrier)`

Value of  $f(k, E)$  used to compute the energy spectrum of the CAES and the Fermi momenta by solving the secular equation  $f(k, E) = 0$ .

**Parameters**

- **E** (*float*) – Energy measured from the Fermi level.
- **k** (*float*) – Momentum along the QH-SC interface.
- **m\_qh** (*float*) – Effective mass in the QH region.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **omega** (*float*) – Cyclotron frequency.
- **delta** (*float*) – Superconducting gap.
- **V\_barrier** (*float*) – Height of the delta-potential barrier.

**Returns** The value of  $f(k, E)$ .

**Return type** float

`modules.functions.velocity(m_qh, m_sc, mu_qh, mu_sc, nu, delta, V_barrier)`

Compute the velocity of the CAES.

**Parameters**

- **m\_qh** (*float*) – Effective mass in the QH region.
- **m\_sc** (*float*) – Effective mass in the SC region.
- **mu\_qh** (*float*) – Chemical potential in the QH region.
- **mu\_sc** (*float*) – Chemical potential in the SC region.
- **nu** (*float*) – Filling factor.
- **delta** (*float*) – Superconducting gap.
- **V\_barrier** (*float*) – Height of the delta-potential barrier.

**Returns** The value of the velocity.

**Return type** float

## 3.2 system.py

Classes constructing different Kwant's systems.

`class modules.system.Device(theta_1, theta_2, params)`

Construct a QH-SC junction with arbitrary QH angles and a rectangle-shaped SC.

**Parameters**

- **theta\_1** (*float*) – First QH angle (in degrees).

- **theta\_2** (*float*) – Second QH angle (in degrees).
- **params** (*dict*) – System’s parameters.

**make\_system**(*onsite, hopping*)

Make the (unfinalized) system.

**Parameters**

- **onsite** (*fun*) – Onsite energy function.
- **hopping** (*fun*) – Hopping energy function.

**Returns** Unfinalized Kwant system.

**class** `modules.system.DeviceInfinite`(*params*)

Construct a an infinite QH-SC interface (lead).

**Parameters** **params** (*dict*) – System’s parameters.

**make\_system**(*onsite, hopping*)

Make the (unfinalized) lead.

**Parameters**

- **onsite** (*fun*) – Onsite energy function.
- **hopping** (*fun*) – Hopping energy function.

**Returns** Unfinalized Kwant lead.

**class** `modules.system.DeviceSingleCorner`(*theta\_qh, theta\_sc, params, small=False*)

Construct a semi-infinite junction with a single corner.

**Parameters**

- **theta\_qh** (*float*) – QH angle (in degrees).
- **theta\_sc** (*str*) – SC angle (in degrees).
- **params** (*dict*) – System’s parameters.
- **small** (*bool*) – Must be set to True when small dimensions are used, default to False.

**make\_system**(*onsite, onsite\_qh, onsite\_sc, hopping, hopping\_qh, hopping\_sc*)

Make the (unfinalized) system.

**Parameters**

- **onsite** (*fun*) – Onsite energy function.
- **onsite\_qh** (*fun*) – Onsite energy function in QH region.
- **onsite\_sc** (*fun*) – Onsite energy function in SC region.
- **hopping** (*fun*) – Hopping energy function.
- **hopping\_qh** (*fun*) – Hopping energy function in QH region.
- **hopping\_sc** (*fun*) – Hopping energy function in SC region.

**Returns** Unfinalized Kwant system.

### 3.3 utils.py

Definitions of the functions used in *calculations.py*.

`modules.utils.compute_corner_transmissions(device, energy=0.0)`

Compute the corner transmission amplitudes with Kwant.

**Parameters**

- **device** (*object*) – The QH-SC corner.
- **energy** (*float*) – Value of the energy, default to 0.

**Returns** The normal and Andreev amplitudes : `t_ee`, `t_he`.

**Return type** list

`modules.utils.compute_delta_b_and_phi_b(device, L_b, v_b, mu_b)`

Compute the effective barrier parameters `delta_b` and `phi_b`.

Here have to give `L_b`, `v_b`, and `mu_b` as inputs.

**Parameters**

- **device** (*object*) – The QH-SC junction.
- **L\_b** (*str*) – Length of the barrier.
- **v\_b** (*str*) – Velocity in the barrier.
- **mu\_b** (*str*) – Chemical potential in the barrier.

**Returns** The values of `delta_b` and `phi_b`.

**Return type** array

`modules.utils.compute_downstream_conductance_TB(device)`

Compute the (zero-temperature) downstream conductance with Kwant.

**Parameters** **device** (*object*) – The QH-SC junction.

**Returns** The value of the downstream conductance.

**Return type** float

`modules.utils.compute_momentum_difference(params, E)`

Compute the momentum difference `dk`.

**Parameters**

- **params** (*dict*) – The system's parameters.
- **E** (*float*) – The energy value.

**Returns** The value of `dk`.

**Return type** float

`modules.utils.compute_nu_crit(params, nu_min=1.0, nu_max=3.0, tol=1e-06)`

Compute the value of `nu_crit`.

**Parameters**

- **params** (*dict*) – The system's parameters.
- **nu\_min** (*float*) – Lower bound, defaults to 1.

- **nu\_max** (*float*) – Higher bound, defaults to 3.
- **tol** (*float*) – Precision of the returned value, defaults to 1E-6.

**Returns** The value of `nu_crit`.

**Return type** `float`

`modules.utils.plot_density(device, energy=0.0, fig_name=False)`

Plot the probability density  $|u|^2 - |v|^2$  of the incoming electron.

**Parameters**

- **device** (*object*) – The device.
- **energy** (*float*) – Value of the energy (relative to the CAES Fermi level), default to 0.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_device(device)`

Plot the Kwant system.

Be carefull, the colors are badly defined for negative angles.

**Parameters** **device** (*object*) – The device.

`modules.utils.plot_downstream_conductance_comparison(Ls, device, device_1, device_2, L_b, v_b, mu_b, fig_name=False, from_data=True)`

Comparison between analytical and tight-binding conductance.

The analytical formula is shifted in order to recover the tight-binding simulation at large L. We we need to give `L_b`, `v_b`, and `mu_b` as inputs.

The shifted data and the plot are saved in the directory ‘files/downstream\_conductance/comparison’.

**Parameters**

- **Ls** (*list*) – The values of the interface’s length.
- **device** (*object*) – The QH-SC junction.
- **device\_1** (*object*) – The first QH-SC corner.
- **device\_2** (*object*) – The second QH-SC corner.
- **L\_b** (*str*) – Length of the barrier.
- **v\_b** (*str*) – Velocity in the barrier.
- **mu\_b** (*str*) – Chemical potential in the barrier.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **from\_data** (*bool*) – If True the spectrum is plotted using existing data. If False the data are computed even if they exist.

`modules.utils.plot_fh_p_vs_Z_various_fillings(nus, Zs, params, fig_name=False)`

Plot the hole content  $f_h^+$  vs Z for various values of the filling factor.

The plot is saved in the directory ‘files/andreev\_and\_hole\_prob/hole\_probability/varying\_Z/plots’.

**Parameters**

- **nus** (*list*) – The values of the filling factor.
- **Zs** (*list*) – The values of the barrier strength.
- **params** (*dict*) – The system’s parameters.

- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_finite_T_conductance_TB_vs_L_various_temps(device, kTs, Ls, from_data=True, fig_name=False)`

Plot the finite-temperature downstream conductance versus the energy for various temperatures.

Here we use a full tight-binding calculation and we compare with the zero-temperature result.

The data and the plot are saved in the directory ‘files/finite\_temperature/downstream\_conductance/varying\_L/tight\_binding’.

#### Parameters

- **device** (*object*) – The QH-SC junction.
- **kTs** (*list*) – The values of  $k_B T$ .
- **Ls** (*list*) – The values of the interface’s length.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_k0_vs_Z_various_fillings(nus, Zs, params, fig_name=False)`

Plot the momentum  $k_0$  versus  $Z$  for various values of the filling factor.

The plot is saved in the directory ‘files/momentum\_k0/varying\_Z/plots’.

#### Parameters

- **nus** (*list*) – The values of the filling factor.
- **Zs** (*list*) – The values of the barrier strength.
- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_k0_vs_nu(nus, params, fig_name=False)`

Plot the momentum at the Fermi level  $k_0$  versus  $\nu$ .

The plot is saved in the directory ‘files/momentum\_k0/varying\_nu/plots’.

#### Parameters

- **nus** (*list*) – The values of the filling factor.
- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_momentum_difference_vs_energy(params, fig_name=False)`

Plot momentum difference *v.s.* energy.

The resulting plot is saved in the ‘files/finite\_temperature/momentum\_difference/varying\_energy’ directory.

#### Parameters

- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_nu_crit_limit_vs_Z(params, from_data=True, fig_name=False)`

Plot the asymptotic value of  $\nu_{\text{crit}}$  *v.s.*  $Z$ .

The resulting plot is saved in the ‘files/track\_states/varying\_Z’ directory.

#### Parameters

- **params** (*dict*) – The system’s parameters.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist. Defaults to True.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_nu_crit_limit_vs_mismatch(params, from_data=True, fig_name=False)`

Plot the asymptotic value of  $\nu_{\text{crit}}$  v.s.  $\mu_{\text{sc}}/\mu_{\text{qh}}$ .

The resulting plot is saved in the ‘files/track\_states/varying\_mismatch’ directory.

#### Parameters

- **params** (*dict*) – The system’s parameters.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist. Defaults to True.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_nu_crit_vs_mu_qh_delta(params, from_data=True, fig_name=False)`

Plot  $\nu_{\text{crit}}$  v.s.  $\mu_{\text{qh}}/\delta$ .

The resulting plot is saved in the ‘files/track\_states/varying\_mu\_qh\_delta’ directory.

#### Parameters

- **params** (*dict*) – The system’s parameters.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist. Defaults to True.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

`modules.utils.plot_spectrum_TB(device, from_data=True)`

Plot tight-binding spectrum.

The data and plots are saved in the directory ‘files/energy\_spectrum/tight\_binding’.

#### Parameters

- **device** (*object*) – Infinite QH-SC interface.
- **from\_data** (*bool*) – If true the spectrum is plotted using the existing data. If False the data are computed even if they exist.

`modules.utils.plot_spectrum_comparison(device, params, fig_name=False, from_data=True)`

Comparison between microscopic and tight-binding spectrums.

The plots are saved in the directory ‘files/energy\_spectrum/comparison/plots’.

#### Parameters

- **device** (*object*) – Infinite QH-SC interface.
- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **from\_data** (*bool*) – If true the spectrum is plotted using the existing data. If False the data are computed even if they exist.

```
modules.utils.plot_spectrum_micro(params, from_data=True, fig_name=False, show_k0=False,
                                   qp_labels=False)
```

Plot microscopic spectrum.

The data and plots are saved in the directory 'files/energy\_spectrum/microscopic'.

#### Parameters

- **params** (*dict*) – The system's parameters.
- **from\_data** (*bool*) – If true the spectrum is plotted using the existing data. If False the data are computed even if they exist.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **show\_k0** (*str*) – Show the position of  $k_0$ , default to False.
- **qp\_labels** (*str*) – Show the quasiparticles labels, default to False.

```
modules.utils.plot_tau_vs_energy(device, fig_name=False, from_data=True, tau_label=None)
```

Plot the conversion probability tau vs energy.

The data and the plot are saved in the directory 'files/finite\_temperature/scattering\_probabilities/transmissions/'.

#### Parameters

- **device** (*object*) – The QH-SC junction.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist. Defaults to True.
- **tau\_label** (*int*) – None, 1 or 2. Defaults to None.

```
modules.utils.plot_tau_vs_mu_qh_delta_various_fillings(nus, deltas, theta_qh, theta_sc, params,
                                                         from_data=True, fig_name=False)
```

Plot the corner's Andreev transmission versus  $\mu_{\{QH\}}/\Delta$  for various values of the filling factor.

The data and plot are saved in the directory 'files/chapter3/andreev\_transmission/varying\_mu\_qh\_delta'.

#### Parameters

- **nus** (*list*) – The values of the filling factor.
- **deltas** (*list*) – The values of the SC gap.
- **theta\_qh** (*float*) – The QH angle.
- **theta\_sc** (*float*) – The SC angle.
- **params** (*dict*) – The system's parameters.
- **from\_data** (*bool*) – If True the spectrum is plotted using the existing data. If False the data are computed even if they exist.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.

```
modules.utils.plot_tau_vs_theta_qh_various_fillings(nus, thetas, device, fig_name=False,
                                                     from_data=True,
                                                     show_only_commensurate=False)
```

Plot the corner's Andreev transmission versus the QH angle for various values of the filling factor.

The data and the plot are saved in the directory 'files/andreev\_and\_hole\_prob/andreev\_transmission/varying\_theta'.

#### Parameters



- **nus** (*list*) – The values of the filling factor.
- **thetas** (*list*) – The values of the QH angle.
- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **from\_data** (*bool*) – If True the spectrum is plotted using the stored data. If False the data are computed even if they exist.
- **show\_only\_commensurate** (*bool*) – If True only the commensurate angles are shown. Defaults to False.

```
modules.utils.plot_tau_vs_theta_sc_various_fillings(nus, thetas, device, fig_name=False,
                                                    from_data=True,
                                                    show_only_commensurate=False)
```

Plot the corner’s Andreev transmission versus the SC angle for various values of the filling factor.

The data and the plot are saved in the directory ‘files/andreev\_and\_hole\_prob/andreev\_transmission/varying\_theta\_sc’.

#### Parameters

- **nus** (*list*) – The values of the filling factor.
- **thetas** (*list*) – The values of the SC angle.
- **params** (*dict*) – The system’s parameters.
- **fig\_name** (*str*) – The name of the plot used for the manuscript, optional.
- **from\_data** (*bool*) – If True the spectrum is plotted using the stored data. If False the data are computed even if they exist.
- **show\_only\_commensurate** (*bool*) – If True only the commensurate angles are shown. Defaults to False.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## LICENSE

See the [license terms](#) for license rights and limitations (MIT).



## PYTHON MODULE INDEX

### C

`calculations`, 7

### m

`manuscript_figures`, 8

`modules.functions`, 9

`modules.system`, 14

`modules.utils`, 16





## C

calculations  
module, 7

chi\_m() (in module modules.functions), 10

chi\_p() (in module modules.functions), 10

compute\_corner\_transmissions() (in module modules.utils), 16

compute\_delta\_b\_and\_phi\_b() (in module modules.utils), 16

compute\_downstream\_conductance\_TB() (in module modules.utils), 16

compute\_momentum\_difference() (in module modules.utils), 16

compute\_nu\_crit() (in module modules.utils), 16

## D

Device (class in modules.system), 14

DeviceInfinite (class in modules.system), 15

DeviceSingleCorner (class in modules.system), 15

## E

effective\_tau() (in module modules.functions), 10

## F

fermi\_momenta() (in module modules.functions), 11

## H

hole\_probability() (in module modules.functions), 11

hopping() (in module modules.functions), 11

hopping\_qh() (in module modules.functions), 12

hopping\_sc() (in module modules.functions), 12

## K

kronecker\_delta() (in module modules.functions), 12

## M

make\_system() (modules.system.Device method), 15

make\_system() (modules.system.DeviceInfinite method), 15

make\_system() (modules.system.DeviceSingleCorner method), 15

manuscript\_figures  
module, 8

module

calculations, 7

manuscript\_figures, 8

modules.functions, 9

modules.system, 14

modules.utils, 16

modules.functions

module, 9

modules.system

module, 14

modules.utils

module, 16

## O

onsite() (in module modules.functions), 12

onsite\_qh() (in module modules.functions), 12

onsite\_sc() (in module modules.functions), 13

## P

phi\_m() (in module modules.functions), 13

phi\_p() (in module modules.functions), 13

plot\_density() (in module modules.utils), 17

plot\_device() (in module modules.utils), 17

plot\_downstream\_conductance\_comparison() (in module modules.utils), 17

plot\_fh\_p\_vs\_Z\_various\_fillings() (in module modules.utils), 17

plot\_finite\_T\_conductance\_TB\_vs\_L\_various\_temps() (in module modules.utils), 18

plot\_k0\_vs\_nu() (in module modules.utils), 18

plot\_k0\_vs\_Z\_various\_fillings() (in module modules.utils), 18

plot\_momentum\_difference\_vs\_energy() (in module modules.utils), 18

plot\_nu\_crit\_limit\_vs\_mismatch() (in module modules.utils), 19

plot\_nu\_crit\_limit\_vs\_Z() (in module modules.utils), 18

`plot_nu_crit_vs_mu_qh_delta()` (in module `modules.utils`), 19  
`plot_spectrum_comparison()` (in module `modules.utils`), 19  
`plot_spectrum_micro()` (in module `modules.utils`), 19  
`plot_spectrum_TB()` (in module `modules.utils`), 19  
`plot_tau_vs_energy()` (in module `modules.utils`), 20  
`plot_tau_vs_mu_qh_delta_various_fillings()`  
(in module `modules.utils`), 20  
`plot_tau_vs_theta_qh_various_fillings()` (in  
module `modules.utils`), 20  
`plot_tau_vs_theta_sc_various_fillings()` (in  
module `modules.utils`), 21

## S

`secular_equation()` (in module `modules.functions`),  
14

## U

`U()` (in module `modules.functions`), 9  
`U_m()` (in module `modules.functions`), 9  
`U_p()` (in module `modules.functions`), 9

## V

`velocity()` (in module `modules.functions`), 14