QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices

Kiršanskas, Gediminas; Pedersen, Jonas Nyvold; Karlström, Olov; Leijnse, Martin Christian; Wacker, Andreas

Published in:
Computer Physics Communications

Link to article, DOI:
10.1016/j.cpc.2017.07.024

Publication date:
2017

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Accepted Manuscript

QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices

Gediminas Kiršanskas, Jonas Nyvold Pedersen, Olov Karlström, Martin Leijnse, Andreas Wacker

PII: S0010-4655(17)30251-5
DOI: http://dx.doi.org/10.1016/j.cpc.2017.07.024
Reference: COMPHY 6291

To appear in: Computer Physics Communications

Received date: 14 March 2017
Revised date: 15 June 2017
Accepted date: 28 July 2017

Please cite this article as: G. Kiršanskas, J.N. Pedersen, O. Karlström, M. Leijnse, A. Wacker, QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices, Computer Physics Communications (2017), http://dx.doi.org/10.1016/j.cpc.2017.07.024

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
QmeQ 1.0: An open-source Python package for calculations of transport through quantum dot devices

Gediminas Kiršanskas\textsuperscript{a}, Jonas Nyvold Pedersen\textsuperscript{b}, Olov Karlström\textsuperscript{a}, Martin Leijnse\textsuperscript{b}, Andreas Wacker\textsuperscript{a}

\textsuperscript{a}Mathematical Physics and NanoLund, Lund University, Box 118, 221 00 Lund, Sweden
\textsuperscript{b}Solid State Physics and NanoLund, Lund University, Box 118, 221 00 Lund, Sweden
\textsuperscript{c}Department of Micro- and Nanotechnology, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

Abstract

QmeQ is an open-source Python package for numerical modeling of transport through quantum dot devices with strong electron-electron interactions using various approximate master equation approaches. The package provides a framework for calculating stationary particle or energy currents driven by differences in chemical potentials or temperatures between the leads which are tunnel coupled to the quantum dots. The electronic structures of the quantum dots are described by their single-particle states and the Coulomb matrix elements between the states. When transport is treated perturbatively to lowest order in the tunneling couplings, the possible approaches are Pauli (classical), first-order Redfield, and first-order von Neumann master equations, and a particular form of the Lindblad equation. When all processes involving two-particle excitations in the leads are of interest, the second-order von Neumann approach can be applied. All these approaches are implemented in QmeQ. We here give an overview of the basic structure of the package, give examples of transport calculations, and outline the range of applicability of the different approximate approaches.

Keywords: Open quantum systems, Quantum dots, Anderson-type model, Coulomb blockade, Python

Program summary

Program Title: QmeQ
Program Files doi: http://dx.doi.org/10.17632/8687mrhgg9.1
Licensing provisions: BSD 2-Clause
Programming language: Python
External libraries: NumPy, SciPy, Cython
Nature of problem: Calculation of stationary state currents through quantum dots tunnel coupled to leads.
Solution method: Exact diagonalisation of the quantum dot Hamiltonian for a given set of single particle states and Coulomb matrix elements. Numerical solution of the stationary-state master equation for a given approximate approach.
Restrictions: Depending on the approximate approach the temperature needs to be sufficiently large compared to the coupling strength for the approach to be valid.

1. Introduction

Quantum dot devices are usually made of a nanostructure or a molecule coupled to leads [1, 2, 3]. A vast amount of experiments have been performed using quantum dots, where with bias spectroscopy [4] or thermoelectric measurements [5], it is possible to obtain the energy level structure of the quantum dot, which is important for understanding and predicting the device behavior. Such a device is also a useful tool for studying the fundamental physics of open-quantum systems in non-equilibrium, where the leads play the role of an environment [6]. Modeling transport through such nanoscale systems is a complicated task, especially when strong Coulomb interaction is present in the quantum dots, as it can lead to phenomena like Coulomb blockade [7, 8], Kondo effect [9, 10, 11], or Pauli blockade [12], just to mention a few. Quantum transport can be calculated using different theoretical methods like scattering-
states numerical-renormalization group [13], non-equilibrium Green’s functions [14, 15], and master equation based approaches [6, 16, 17, 18, 19, 20, 21, 22].

We consider here approximate master equation approaches, which were also used to interpret transport experimental data in the regime of weak coupling to the environment [23, 24, 25]. In particular, we address the so-called Pauli master equation [6, 26, 27], the first-order Redfield approach [6, 28, 29], the first/second-order von Neumann approaches [19, 30], and a particular form of the Lindblad equation [31, 32], and apply these methods for tunneling models, where the quantum dots can have arbitrary Coulomb interactions [33]. Even though there is a lot of literature on such methods, there is no publicly-available, well-documented package, which is easy to apply for quantum dot model systems. One reason is that depending on parameter regime these methods can fail and a good knowledge of the method is required to know whether to trust the result or not. For example, these methods can violate positivity [6] of the reduced density matrix and lead to large currents flowing against the bias [34]. Nevertheless, we think it is important to have a package where a user can duplicate existing calculations, check the applicability of different methods, or simply discover new kind of physics using different approximate master equations.

In this paper, we describe an open-source package QmeQ designed for simulating stationary-state transport through quantum dots (available at http://github.com/gedaskir/qmeq). The name QmeQ deciphers as Quantum master equation for Quantum dot transport calculations. It is written in the programming languages Python [35] and Cython [36, 37]. In recent years, a large number of Python-based scientific computation packages have appeared [38] because of the development of NumPy [39] and SciPy [40]. These have modules for linear algebra, optimisation, and special functions, which also QmeQ relies on. Thus it is convenient to use the same open-source platform based on Python, which provides an easy integration between different kind of packages. Additionally, Python is a high-level interpreted language, which allows development of an easy-to-read code and easy usage. For some parts of the code we use Cython which lets us to compile bottleneck parts of the package to achieve better performance.1 Moreover, in our examples for data visualization we use the matplotlib package [41], the tutorials are provided using Jupyter notebooks [42], and detailed documentation of the code is generated using Sphinx [43].

There are already a variety of well-developed open-source Python packages dedicated to model quantum mechanical phenomena and transport. One such package is ASE (Atomic Simulation Environment), which performs atomistic simulations with the possibility to make transport calculations in molecular junctions using density-functional theory (DFT) and non-equilibrium Green’s functions (NEGF) [44]. For numerical calculations on tight-binding models with a focus on quantum transport the Kwant package can be used [45]. The simulation of the dynamics of open quantum systems using master equations in Lindblad, Floquet–Markov, or Bloch–Redfield form can be performed using the QuTiP (Quantum Toolbox in Python) framework [46, 47]. In contrast to QuTiP, we focus on a particular model for the quantum dot device, where Coulomb interaction between particles can be dominating, and allow for a direct comparison between different approximate quantum master equations, including the second-order von Neumann approach. For an overview of other transport packages including commercial ones also see Section 4 of Ref. [45]. Lastly, the cotunneling calculations using a T-matrix based approach [27] can be performed using the humo package.2

The paper is organized as follows. In Section 2 we introduce the model Hamiltonian for the quantum dot device and in Section 3 we describe how such a Hamiltonian is set up in QmeQ. In Sections 4, 5, and 6 we give examples of transport calculation through various quantum dot devices, using different approaches (examples scripts are available at http://github.com/gedaskir/qmeq-examples). The derivation of the approximate master equations used in QmeQ is relegated to Appendices. Throughout the paper our units are such that $\hbar = 1, k_B = 1, |e| = 1$.

2. The model

The system under consideration consists of a number of quantum dots attached to a number of metallic leads. Such a quantum dot device can be modelled by the following Hamiltonian:

$$H = H_{\text{leads}} + H_{\text{tunneling}} + H_{\text{dot}}.$$  \hspace{1cm} (1)

1In short, Cython converts Python-like code with static type definitions into C code.

2Available at http://github.com/georglind/humo. This package is currently rather undocumented but the used methods are well described in Ref. [48]. The calculations can be performed on models of the type described by Eq. (1).
where $H_{\text{leads}}$ describes electrons in the leads, $H_{\text{dot}}$ describes electrons in the dot, and $H_{\text{tunneling}}$ corresponds to tunneling between the leads and the dot. The leads are described as non-interacting electrons

$$H_{\text{leads}} = \sum_{\alpha k} e_{\alpha k} c_{\alpha k}^{\dagger} c_{\alpha k},$$

where $c_{\alpha k}^{\dagger}$ creates an electron in the lead channel $\alpha$ with $k$ representing a continuous quantum number, which refers to a continuum energy. This means that $k$-sums can be performed by introducing the density of states $\nu(E)$ as $\sum_{k} f_{k} \rightarrow \int dE\nu(E)f(E)$. The lead channel $\alpha$ can encompass, for example, an actual source/drain lead label, spin of an electron, etc., depending on actual physical setup under consideration.

For the quantum dot the following rather general many-body Hamiltonian is used

$$H_{\text{dot}} = H_{\text{single}} + H_{\text{Coulomb}},$$

$$H_{\text{single}} = \sum_{i} \epsilon_{i} d_{i}^{\dagger} d_{i} + \sum_{i \neq j} \Omega_{ij} d_{i}^{\dagger} d_{j},$$

$$H_{\text{Coulomb}} = \sum_{mnkl} U_{mnkl} d_{m}^{\dagger} d_{n}^{\dagger} d_{k} d_{l}, \quad \text{with } m < n,$$

where $d_{i}^{\dagger}$ creates an electron in single-particle orbital $i$, $\epsilon_{i}$ is the energy of that orbital, and $\Omega_{ij}$ gives the hybridization between single-particle orbitals. Here also the Coulomb interaction $U_{mnkl}$ can be present. We note that the Coulomb interaction takes the form of Eq. (5) (without a factor of $\frac{1}{2}$) for state labels $m < n$. Lastly, the tunneling Hamiltonian is

$$H_{\text{tunneling}} = \sum_{\alpha k,i} t_{\alpha k,i} c_{\alpha k}^{\dagger} c_{\alpha k} + \text{H.c.},$$

where H.c. denotes Hermitian conjugate of the first term and $t_{\alpha k,i}$ is the tunneling amplitude between the leads and the dot. An important energy scale in the calculations is the tunneling rate defined as

$$\Gamma_{\alpha i}(E) = 2\pi \sum_{k} |t_{\alpha k,i}|^{2} \delta(E - \epsilon_{ak}).$$

In the case of single spinful orbital with on-site interaction $U$ Hamiltonian Eq. (1) is referred to as Anderson-type model [33, 49, 50].

2.1. Approximations

For the calculations in QmeQ we make the following important assumption for the model Eq. (1):

1. The leads are thermalized according to a Fermi-Dirac occupation function $f_{\alpha}(E) = \left[ e^{(E-\mu_{\alpha})/T_{\alpha}} + 1 \right]^{-1}$ with respective temperatures $T_{\alpha}$ and chemical potentials $\mu_{\alpha}$.

Additionally, we use the so-called wide-band limit for the leads:

2. The leads have constant density of states $\nu(E) \approx \nu(E_{F}) = \nu_{F}$, where the subscript $F$ corresponds to the Fermi level. Then the $k$-sums are performed as $\sum_{k} \rightarrow \nu_{F} \int_{D} dE$, where $D$ denotes bandwidth of the leads. Also the tunneling amplitudes are energy (or $k$) independent, i.e., $t_{\alpha k,i} \approx t_{\alpha i}$. In this case the tunneling rates become $\Gamma_{\alpha i} = 2\pi \nu_{F} |t_{\alpha i}|^{2}$.

Approximation 2. is made for convenience to simplify the integrals and not to clutter the QmeQ code with the specification of energy-dependent functions. The wide-band limit can be easily relaxed. On the other hand, this is often a good approximation for metallic leads.
2.2. Many-body eigenstates and the reduced density matrix

In QmeQ the quantum dot Hamiltonian (3) is constructed in a Fock basis and diagonalized exactly to obtain the many-body eigenstates \(|a\rangle\),

\[ H_{\text{dot}} = \sum_a E_a |a\rangle \langle a| \]  

(8)

We note that a fermionic many-body Hamiltonian is constructed efficiently using Lin tables (for more details see Refs. [48, 51]). The tunneling Hamiltonian is expressed in this many-body eigenbasis as

\[ H_{\text{tunneling}} = \sum_{ab,ak} T_{ba,ak} |b\rangle \langle a| c_{ak} + \text{H.c.}, \quad T_{ba,ak} = \sum_{i} t_{i}(b|d_{i}\rangle |a\rangle, \]  

(9)

where \( T_{ba,ak} \) represents many-particle tunneling amplitudes. Here we used the letter convention: if more than one state enters an equation, then the position of the letter in the alphabet follows the particle number (for example \( N_b = N_a + 1 \), \( N_c = N_a + 2 \), \( N_d = N_a \)). In such a way the sum \( \sum_{i} \) restricts to those combinations, where \( N_c = N_b + 1 \). The approximate quantum master equations are constructed in the many-body eigenbasis of the quantum dot and are discussed in Appendices.

The central quantity in the approximate master equations is the reduced density matrix, which is defined as:

\[ \Phi_{bb}^{[0]} = \sum_g \langle bg|\rho|bg\rangle, \]  

(10)

with \( \rho \) being the full density matrix operator of the system. Here \( g \) represents the many-body eigenstates of the lead Hamiltonian \( H_{\text{leads}} \), (2) with arbitrary occupations of the lead states \( \kappa \). Then a basis of the combined lead and dot system is given by the product states \(|bg\rangle = |b\rangle \otimes |g\rangle\).

2.3. Calculation of currents

In QmeQ the current emanating from a particular lead channel \( \alpha \) is calculated for a stationary state. The particle current leaving the lead channel \( \alpha \) is defined as

\[ I_{\alpha} = \frac{\partial}{\partial t} (N_{\alpha}) = -i[\{H, N_{\alpha} \}], \quad \text{with} \quad N_{\alpha} = \sum_k c_{ak}^{\dagger} c_{ak}, \]  

(11)

and the energy current as

\[ \dot{E}_{\alpha} = \frac{\partial}{\partial t} (H_{\alpha}) = -i[\{H, H_{\alpha} \}], \quad \text{with} \quad H_{\alpha} = \sum_k E_{ak} c_{ak}^{\dagger} c_{ak}. \]  

(12)

We introduce the following energy-resolved particle-current amplitudes:

\[ \Phi_{ba,ak}^{[1]} = \sum_{\gamma} \langle eg - \gamma k|\rho|bg\rangle (-1)^{N_{\gamma}} \otimes |b\rangle \otimes |c_{ak}g\rangle, \]  

(13)

where \( N_{\gamma} \) is the number of particles in the many-body state \(|b\rangle\). We can express the particle and energy currents as

\[ I_{\alpha} = -2 \sum_{k,\gamma} \text{Im}[T_{ba,ak} \Phi_{ba,ak}^{[1]}], \]  

(14)

\[ \dot{E}_{\alpha} = -2 \sum_{k,\gamma} E_{ak} \text{Im}[T_{ba,ak} \Phi_{ba,ak}^{[1]}]. \]  

(15)

Lastly, the heat current emanating from the lead channel \( \alpha \) can be defined as

\[ Q_{\alpha} = \dot{E}_{\alpha} - \mu_{\alpha} I_{\alpha}. \]  

(16)

The above definition of the heat current corresponds to the rate of change of the entropy \( \dot{S}_{\alpha} = \frac{1}{T_{\alpha}} (\dot{E}_{\alpha} - \mu_{\alpha} N_{\alpha}) \).
Figure 1: Spinful single orbital coupled to spinful metallic leads, where $\Gamma_L$ and $\Gamma_R$ are the tunneling rates to the source ($L$) and the drain ($R$) leads, respectively. Numbers represent the labeling of states in the example. Energies of the spin-up and spin-down levels are given by $\varepsilon_0$ and $\varepsilon_1$, respectively. $\Omega$ is the hybridisation between the states, which can appear due to presence of spin-orbit coupling (sets a concrete spin quantization axis) and a magnetic field perpendicular to the spin orientation.

3. The QmeQ package

The QmeQ package diagonalises the quantum dot Hamiltonian (3) exactly and performs various approximate master equation calculations on the system Hamiltonian (1), where tunneling (6) is treated as a perturbation. Installation instructions for QmeQ are provided in the INSTALL.md file coming with the source code of the package. We start with a short script giving a minimal example of a quantum dot containing one spinful orbital and on-site charging energy $U$ coupled to source ($L$) and drain ($R$) leads as shown in Figure 1:

$$H_{\text{one}} = \sum_{k,l=L,R} \sum_{\sigma=\uparrow,\downarrow} \varepsilon_{lk} c_{l\sigma,k}^\dagger c_{l\sigma,k} + \sum_{l;k} (tr d_{\sigma,l} c_{l\sigma,k} + H.c.) + \sum_{\sigma} \varepsilon_{\sigma} d_{\sigma}^\dagger d_{\sigma} + (\Omega d_{\uparrow}^\dagger d_{\uparrow} + H.c.) + Ud_{\uparrow}^\dagger d_{\uparrow}. \quad (17)$$

The script calculating the particle current and the energy current in a single point in the parameter space reads

```python
# Prerequisites
import qmeq
from numpy import sqrt, pi, power

# Parameters used (numbers are in arbitrary energy scale, e.g., meV)
e0, e1, omega, U = 0.0, 0.0, 0.0, 20.0
# dband is bandwidth $D$ of the leads
tempL, tempR, muL, muR, dband= 1.0, 1.0, 0.2, -0.2, 60.0
gammaL, gammaR = 0.5, 0.7
tL, tR = sqrt(gammaL/(2*pi)), sqrt(gammaR/(2*pi))

## Hamiltonian, which requires definition
# nsingle is the number of single-particle states
# Here level 0 is spin up; level 1 is spin down
nsingle = 2
hsingle = {(0,0): e0, (1,1): e1, (0,1): omega}
# (0,1,1,0) represents an operator $d_0^+d_1^+d_1^-d_0^-$
coulomb = {(0,1,1,0): U}

## Lead and tunneling properties, which requires definition
```
Here we have calculated the particle current and the energy current from the four lead channels $L \uparrow$, $R \uparrow$, $L \downarrow$, $R \downarrow$ into the dot using Pauli master equation and second-order von Neumann approach (2vN) and we get the following output:

<table>
<thead>
<tr>
<th>Output is currents in four lead channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>[L-up R-up L-down R-down]</td>
</tr>
<tr>
<td>Pauli, particle current (in units of energy_scale/hbar):</td>
</tr>
<tr>
<td>[0.01948779 -0.01948779 0.01948779 -0.01948779]</td>
</tr>
<tr>
<td>Pauli, energy current (in units of energy_scale^2/hbar):</td>
</tr>
<tr>
<td>[1.59585694e-09 -1.59585694e-09 1.59585694e-09 -1.59585694e-09]</td>
</tr>
<tr>
<td>2vN, particle current (in units of energy_scale/hbar):</td>
</tr>
<tr>
<td>[0.01431084 -0.01431084 0.01431084 -0.01431084]</td>
</tr>
<tr>
<td>2vN, energy current (in units of energy_scale^2/hbar):</td>
</tr>
<tr>
<td>[0.00617130 -0.00617129 0.00617130 -0.00617129]</td>
</tr>
</tbody>
</table>
We elaborate more on this single spinful orbital example in Section 4. Next we discuss what different parts of the script mean.

From the above example we see that the system is defined using the Builder class. For Builder to construct a system object, we need to specify at least the following variables:

```python
system = qmeq.Builder(nsingle, # Number of single-particle states
                        hsingle, # Single-particle Hamiltonian
                        coulomb, # Coulomb matrix elements
                        nleads, # Number of lead channels
                        tleads, # Single-particle tunneling amplitudes
                        mulst, # Chemical potentials of leads
                        tlst, # Temperatures of leads
                        dband) # Bandwidth of leads
```

Additionally, there is a variety of optional arguments, which can be viewed by typing `help(qmeq.Builder)` in the Python interpreter. These optional arguments can be set by specifying `Builder(..., argument=value)`.

The variables `nsingle`, `hsingle`, and `coulomb` describe the quantum dot Eq. (3). `nsingle` is the number of single-particle states. Single-particle states are labeled by integers from 0 to `nsingle − 1`, and the single-particle Hamiltonian (4) `hsingle` is specified using the Python dictionary:

```python
hsingle = {(0,0): ε₀, (0,1): Ω₀|, ..., (0,nsingle−1): Ω₀(nsingle−1),
            ...,
            (i,i): εᵢ, (i,i+1): Ωᵢ|, ..., (i,nsingle−1): Ωᵢ(nsingle−1),
            ...
            (nsingle−1,nsingle−1): Ω(nsingle−1,nsingle−1)}
```

In a dictionary it is enough to specify one element like $\Omega_{ij} d_i^\dagger d_j$, because the element $\Omega_{si} d_i^\dagger d_s$ is determined by complex conjugation $\Omega_{ij} = \Omega_{ji}^*$ and is included automatically in order to get a Hermitian quantum dot Hamiltonian. If an element like $\langle j,i \rangle$ is given, it will be added to the Hamiltonian. So specifying $\{(i,j): \Omega_{ij}, \langle j,i \rangle: \Omega_{ij} \}$ will simply double count $\Omega_{ij}$. Default values of $\epsilon_i$ and $\Omega_{ij}$ are zero.

The interaction Hamiltonian (5) `coulomb` can be specified using a dictionary as

```python
coulomb = {\langle m,n,k,l \rangle: U_{mnlk},
            ...
            \langle i,j,p,r \rangle: U_{ijpr}}
```

where $U_{mnlk}d_m^\dagger d_l^\dagger d_i d_j$, and $U_{ijpr}d_i^\dagger d_j^\dagger d_p d_r$, represent some non-zero Coulomb interaction matrix elements.

The variable `nleads` is an integer and sets the number of lead channels (we will simply call it leads). The properties of leads can be given in dictionaries `mulst` (chemical potential) and `tlst` (temperature), which have `nleads` entries. Also the leads are labeled by integers from 0 to `nleads − 1`. So the lead properties can be set by using

```python
mulst = [0: μ₀, 1: μ₁, ..., nleads−1: μₙleads−1]
tlst = [0: T₀, 1: T₁, ..., nleads−1: Tₙleads−1]
dband = D
```

We note that `dband` is a floating point number representing the bandwidth $D$ of the leads.

Lastly, the single-particle tunneling amplitudes `tlleads` can be specified using a dictionary as

```python
tleads = {\langle α,i \rangle: t_{αi},
           ...
           \langle β,j \rangle: t_{βj}}
```

where first index in $\langle α,i \rangle$ is the lead label and second index is the single-particle state label. Here $t_{αi} = \sqrt{V_{lαi}}$.

---

3For the 2eV approach it is necessary to set `kpnt`, which is the number of the equidistant energy grid points for the lead electrons.
Once the system object was constructed using the Builder class it can be solved by simply using

```
        system.solve()
```

The `system.solve()` performs the following steps

1. **qdq**, obtains many-body eigenstates and eigenenergies of the quantum dot Hamiltonian \((3)\) (sets `system.qd.Ea`).
2. **rotateq**, expresses the tunnelling Hamiltonian \((6)\) in many-body eigenbasis (sets `system.lead.Tba`).
3. **masterq**, solves approximate master equation and obtains reduced density matrix elements \(\Phi^{0}\) (sets `system.phi0`).
4. **currentq**, calculates the current amplitudes and currents (sets `system.phi1`, `system.current`, etc.).

Unnecessary steps can be skipped. For example,

```
        system.solve(qdq=True, rotateq=True, masterq=False, currentq=False)
```

will perform step 1. (`qdq=True`) and 2. (`rotateq=True`), however, the master equation will not be solved (`masterq=False`) and the current will not be calculated (`currentq=False`).

How step 3. is performed depends on the master equation used. For first-order approaches we solve the following linear equations

\[
    \mathcal{L}\Phi^{0} = 0, \\
    \text{Tr}[\Phi^{0}] = 1,
\]

where \(\mathcal{L}\) is the kernel (or Liouvillian) of the approach (see Appendices for more details). The solution procedure for the 2vN approach is more complicated and is described in Appendix B.

After the calculation different properties of the system can be accessed through

```
        system.current  # Particle current
        system.energy_current  # Energy current
        system.heat_current  # Heat current
        system.phi0       # Reduced density matrix elements
        system.phi1       # Energy integrated current amplitudes
        system.Ea        # Eigenenergies of the quantum dot
        system.Tba       # Many-body tunneling amplitudes
```

For example, `system.current` is a one-dimensional NumPy array of length nleads.

### 3.1. Attributes and functions of the Builder class

Here we concisely describe some of the useful attributes and functions of the Builder class. The variable

```
        system.indexing  # Possible values: 'Lin', 'charge', 'sz', 'ssq'
```

determines the labeling of states and symmetries, which can be used to simplify the numerical calculations. We note that 'sz' stands for classification using total spin projection \(S_z\) and 'ssq' stands for \(S^2\). For 'Lin' and 'charge' only the charge is considered as a good quantum number. For now the usage of symmetries to optimize the calculations are available only for the first-order approaches in QmeQ. The central idea of labeling of many-body states is so-called Lin tables [48, 51]. Here Fock states are labeled by a decimal number and an actual Fock state is obtained by converting the decimal label to a binary number. For example, let us say we have `nsingle=6`, which leads to `nmany=64`. Then the labels ‘33’ and ‘44’ would represent different Fock states for different indexing:

```
        'Lin'  'charge'  'sz' or 'ssq'
        33 :  [1,0,0,0,0,1], [0,1,1,0,1,1], [0,1,0,1,1,1]
        44 :  [1,0,1,1,0,0], [0,1,1,0,1,1], [1,0,0,1,1,1]
```

\(^4\)For the 2vN approach it is also mandatory to specify the number of iterations, `system.solve(niter=integer)`.

\(^5\)For the 2vN approach, if `masterq=True` then the current is always calculated.
Decimal numbers 33 and 44 are the binary numbers 100001 and 101100. For 'charge' indexing the Lin labels are sorted by increasing charge of the states and for 'sz' and 'ssq' the sorting goes first by the charge and then by the $S_z$ value of states.

The quantum dot Hamiltonian has $n\text{many}=2^{n\text{single}}$ many-body eigenstates $|b\rangle$, which in Eq. (8) are labeled by integers $b$ from 0 to $n\text{many}-1$. To get information about the many-body eigenstate $|b\rangle$ use the function

```python
system.print_state(b)
```

and to get information about the all many-body eigenstates use

```python
system.print_all_states(filename)
```

where `filename` is a string giving the file destination where this information will be printed. After diagonalisation depending on indexing, the many-body eigenstates have a particular labeling convention. For example, with indexing='ssq', which has the highest symmetry, the states are sorted in the following order of parameters: charge, $S_z$, $S^2$, and energy $E_b$. To change the sorting order and relabel the states use the function

```python
system.sort_eigenstates(srt=[l0, l1, l2, l3])
```

Here $l_0 \in \{0, 1, 2, 3\}$, where different numbers represent such properties: 0 – $E_b$, 1 – charge, 2 – $S_z$, 3 – $S^2$. So the default sorting for 'ssq' is srt=[1, 2, 3, 0]. For example, in order to sort by the energy and then by the charge use srt=[0, 1]. However, we note that sort_eigenstates affects the labels when printing the states and the default labels are used in actual calculations. For instance, the order of energies in system.Ea or tunneling amplitudes system.Tba will not be changed by using sort_eigenstates.

In some master equation calculations not all many-body states may be relevant and contribute to the transport, because they are very far above the ground state in energy. To neglect the states with energy higher than $\Delta E$ above the ground state use

```python
system.remove_states(\Delta E)
```

and to reset the usage of all states call

```python
system.use_all_states()
```

To access particular matrix elements of the reduced density matrix $\Phi^{[0]}_{b'b'}$ and current amplitudes $\Phi^{[1]}_{c'b,\alpha}$ the following function can be used

```python
system.get_phi0(b, b')
system.get_phi1(\alpha, b, b')
```

Values of $\Phi^{[0]}_{b'b'}$ and $\Phi^{[1]}_{c'b,\alpha}$ are stored in the arrays

```python
system.phi0
system.phi1
```

We note that there is a difference how the reduced density matrix $\Phi^{[0]}$ is stored in phi0 for the first-order approaches and the 2vN approach. For the 2vN approach phi0 is a one-dimensional array of complex numbers of length $n\text{dm0}$, where $n\text{dm0}$ denotes the total number of matrix elements in $\Phi^{[0]}$. For the first-order approaches we exploit the Hermiticity $\Phi^{[0]}_{b'b} = \Phi^{[0]}_{b'b}$, and store just the elements with $b \leq b'$, where $n\text{dm0}$ now denotes the number of complex elements in the upper triangle of $\Phi^{[0]}$. In this case phi0 is a one-dimensional array of float numbers and is of length $2 \times n\text{dm0} - n\text{pauli}$, where $n\text{pauli}$ is the number of diagonal elements. The diagonal elements $\Phi^{[0]}_{bb}$ can be accessed by phi0[npauli:npauli]. The entries phi0[npauli:ndm0] and phi0[ndm0:] contain the real and imaginary parts of $\Phi^{[0]}_{bb}$ with $b < b'$. The variables npauli and ndm0 are contained in

```python
system.si.npauli
system.si.ndm0
```

The Liouvillian $L$ corresponding to the reduced density matrix $\Phi^{[0]}$ can be accessed through

```python
system.kern
```
Lastly, when doing calculations there is a need to change the parameters of the system. This can be achieved by using the functions

```python
system.add(hsingle, coulomb, tleads, mulst, tlst)
system.change(hsingle, coulomb, tleads, mulst, tlst)
```

where as the names suggest `system.add` and `system.change` adds and changes the value of a parameter. For example, to change the temperature of the lead 0 to $T_0$ one can use the call: `system.change(tlst={0: T_0})`. To modify `dband` to value $D$ simply use `system.dband=D`.

### 3.2. Important classes

*QmeQ* Builder interacts with different classes to set up the system and perform calculations using different approaches as shown in Figure 2. Here we concisely describe these classes. StateIndexingDM is a class describing the indexing of many-body states and density matrix elements. An object of this class is accessed in previously constructed Builder object `system` through

```
system.si
```

The class StateIndexingDMc is used for the 2vN approach. Both StateIndexingDM and StateIndexingDMc are derived from StateIndexing. The QuantumDot is a class used internally in the Builder to construct and diagonalise the many-body Hamiltonian (3). An object of this class is accessed through

```
system.qd
```

The LeadsTunneling is a class used to represent properties of the leads, construct many-body tunneling amplitudes, and express it the eigenbasis of the Hamiltonian. An object of this class is accessed through

```
system.leads
```

The FunctionProperties is a class containing miscellaneous variables used by the Approach class. An object of this class is accessed through

```
system.funcp
```

Finally, the classes which deal with implementation of different approaches are derived from Approach class, and its object is

```
system.appr
```
The appr object is created using

```
appr = Approach(sys)
```

where sys is any object, which has attributes qd, leads, si, and funcp. For more details and structure see the source code of Approach class contained in qmeq.aprclass module.

For a custom approach the user can define a custom class, which needs to be derived from Approach. For example, let us take the Python implementation of the Lindblad approach contained in qmeq.approach.lindblad module and define a custom class from it:

```python
from qmeq.aprclass import Approach
# Makes factors used for generating Lindblad master equation kernel
from qmeq.approach.lindblad import generate_tLba
# Generates master equation kernel
from qmeq.approach.lindblad import generate_kern_lindblad
# Calculates currents
from qmeq.approach.lindblad import generate_current_lindblad
# Acts on given reduced density matrix with kernel
from qmeq.approach.lindblad import generate_vec_lindblad

class ApproachCustom(Approach):
    kerntype = 'custom_pyLindblad'
    generate_fct = staticmethod(generate_tLba)
    generate_kern = staticmethod(generate_kern_lindblad)
    generate_current = staticmethod(generate_current_lindblad)
    generate_vec = staticmethod(generate_vec_lindblad)
```

We note that here we use static methods in order to have the same behavior between Python and Cython implementations. Then this custom approach can be used by setting Builder(..., kerntype=ApproachCustom). Here we used already existing implementation of the Lindblad approach, but the user can redefine, for example, the function generate_tLba, which generates particular matrix elements of the jump operators.

### 3.3. Parallelization

In the present version QmeQ 1.0 the modules, which generate the Hamiltonian, the Liouvillian, and calculate the current, natively do not support parallelization. However, on computers with many cores the QmeQ code still can exploit parallelization through NumPy, when calculating matrix products, the eigenvalues, or matrix inverse, which is the most time consuming for larger systems. For NumPy to have the best multithreading capabilities it should be linked to so-called ATLAS/OpenBLAS/MKL libraries. The usage of many threads can be achieved by setting an environment variable OMP_NUM_THREADS. In Python this is done in the following way (here we use 8 threads):

```python
import os
os.environ['OMP_NUM_THREADS'] = '8'
```

The above method may not always work in all operating systems. It is also possible to change OMP_NUM_THREADS using the terminal of a particular operating system. For example, on Windows terminal use set OMP_NUM_THREADS=8 and on Unix/Linux/MacOS terminal use export OMP_NUM_THREADS=8. Lastly, when the calculations are made for large set of parameters it is always possible to use very simple poor-man’s parallelization, where for every parameter value a new Python instance is run. This works when every instance is not too much memory consuming.

### 4. Example 1: Spinful single orbital

In this section we elaborate more on the spinful single orbital example (see Eq. (17) and Figure 1) and show how to calculate a stability diagram for such a quantum dot. Here we parameterize the spin-up and spin-down energies as

\[
e_{\uparrow} = V_g + \frac{B}{2}, \quad e_{\downarrow} = V_g - \frac{B}{2},
\]

where \(V_g\) is the gate voltage and \(B\) is the magnetic field (representing anomalous Zeeman splitting of spinful orbital). First, we import all the relevant prerequisites
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq

and we choose such values for parameters:

# Quantum dot parameters
vgate, bfield, omega, U = 0.0, 0.0, 0.0, 20.0
# Lead parameters
vbias, temp, dband = 0.5, 1.0, 60.0
# Tunneling amplitudes
gam = 0.5
t0 = np.sqrt(gam / (2*np.pi))

Here the variable \( \Gamma = 2\pi|t_0|^2 \) with \( t_0 = \sqrt{\nu_F}t_0 \) (see Section 8). In \( H_{\text{occ}} \) we have two single particle states \(|\uparrow\rangle\) and \(|\downarrow\rangle\), which we label by 0 and 1, respectively. We also have four energy integrated lead channels \( L \uparrow, R \uparrow, L \downarrow, R \downarrow \), which we label 0, 1, 2, 3. This system is constructed and solved with the Pauli master equation in the following way:

nsingle = 2
# 0 is up, 1 is down
hsingle = { (0,0): vgate+bfield/2, 
            (1,1): vgate-bfield/2, 
            (0,1): omega }

coulomb = { (0,1,1,0): U }

tleads = { (0,0): t0, # L, up <--- up
           (1,0): t0, # R, up <--- up
           (2,1): t0, # L, down <--- down
           (3,1): t0 } # R, down <--- down
           # lead label, lead spin <--- level spin

nleads = 4  
# L,up R,up L,down R,down
mulst = { 0: vbias/2, 1: -vbias/2, 2: vbias/2, 3: -vbias/2 }
tlst = { 0: temp, 1: temp, 2: temp, 3: temp }

system = qmeq.Builder(nsingle, hsingle, coulomb, 
                        nleads, tleads, mulst, tlst, dband, 
                        kerntype='Pauli')

system.solve()

We can check the obtained current and energy current using:

print('Pauli current: ')
print(system.current)
print(system.energy_current)

Pauli current:
[ 0.0207255 -0.0207255 0.0207255 -0.0207255]
[ 1.73557597e-09 -1.73557597e-09 1.73557597e-09 -1.73557597e-09]

The four entries correspond to current in the four lead channels \( L \uparrow, R \uparrow, L \downarrow, R \downarrow \). Currents through the left lead and the right lead channels are conserved up to numerical errors.
Current continuity:
2.77555756156e-17

If we want to change the approximate approach we could redefine the system with qmeq.Builder by specifying the new kerntype. It is also possible just to change the value of system.kerntype:

```python
kernels = ['Redfield', '1vN', 'Lindblad', 'Pauli']
for kerntype in kernels:
    system.kerntype = kerntype
    system.solve()
    print(kerntype, ' current: ', system.current)
```

Redfield current:
[ 0.02072555 -0.02072555 0.02072555 -0.02072555]
1vN current:
[ 0.02072555 -0.02072555 0.02072555 -0.02072555]
Lindblad current:
[ 0.02072555 -0.02072555 0.02072555 -0.02072555]
Pauli current:
[ 0.02072555 -0.02072555 0.02072555 -0.02072555]

For the considered system, the Pauli, Redfield, 1vN, and Lindblad methods all yield the same results as for given parameter values spin is a good quantum number and no coherences between \(|\uparrow\rangle\) and \(|\downarrow\rangle\) are developed.

In order to use the 2vN approach, we will rebuild the system, because the solution method is rather different. We also have to specify an equidistant energy grid on which the 2vN calculations are performed (iterative solution of an integral equation):

```python
kpnt = np.power(2,12)
system2vN = qmeq.Builder(nsingle, hsingle, coulomb, nleads, tleads, mulst, tlst, dband, kerntype='2vN', kpnt=kpnt)
```

Let us solve the 2vN integral equations using 7 iterations:

```python
system2vN.solve(niter=7)
print('Particle current: ')
print(system2vN.current)
print('Energy current: ')
print(system2vN.energy_current)
```

Particle current:
[ 0.01595739 -0.01595739 0.01595739 -0.01595739]
Energy current:
[ 0.00599524 -0.00599523 0.00599524 -0.00599523]

We see that the particle current is reduced and the energy current is increased considerably compared to the Pauli result. This is related to the inclusion of broadening in the 2vN approach. We can check the current for every iteration to see if it has converged:

---

6 The solution procedure is implemented by a Approach2vN object and not a Approach object, which is used for the first-order approaches.
for i in range(system2vN.niter+1):
    print(i, system2vN.iters[i].current)

0   [ 0.01505524 -0.01505524  0.01505524 -0.01505524]
1   [ 0.01589457 -0.01589457  0.01589457 -0.01589457]
2   [ 0.01595427 -0.01595427  0.01595427 -0.01595427]
3   [ 0.01595720 -0.01595720  0.01595720 -0.01595720]
4   [ 0.01595738 -0.01595738  0.01595738 -0.01595738]
5   [ 0.01595739 -0.01595739  0.01595739 -0.01595739]
6   [ 0.01595739 -0.01595739  0.01595739 -0.01595739]

In experiments usually the bias \( V \) and gate voltage \( V_g \) are controlled. A contour plot of current or conductance in the \((V, V_g)\) plane is called a stability diagram. Let us produce such stability diagram for our spinful single orbital quantum dot using the Pauli master equation. Thus we define a function for calculation

```python
def stab_calc(system, bfield, vlst, vglst, dV=0.0001):
    vpnt, vgpnt = vlst.shape[0], vglst.shape[0]
    stab = np.zeros((vpnt, vgpnt))
    stab_cond = np.zeros((vpnt, vgpnt))
    #
    for j1 in range(vgpnt):
        system.change(hsingle={(0,0): vglst[j1]+bfield/2, (1,1):vglst[j1]-bfield/2})
        system.solve(masterq=False)
        for j2 in range(vpnt):
            system.change(mulst={0: vlst[j2]/2, 1: -vlst[j2]/2, 2: vlst[j2]/2, 3: -vlst[j2]/2})
            system.solve(qdq=False)
            stab[j1, j2] = (system.current[0] + system.current[2])
            #
            system.add(mulst={0: dV/2, 1: -dV/2, 2: dV/2, 3: -dV/2})
            system.solve(qdq=False)
            stab_cond[j1, j2] = (system.current[0] + system.current[2] - stab[j1, j2])/dV
    return stab, stab_cond
```

and a function for plotting

```python
def stab_plot(stab_cond, vlst, vglst, U, gam, title):
    (xmin, xmax, ymin, ymax) = np.array([vglst[0], vglst[-1], vlst[0], vlst[-1]])/U
    fig = plt.figure(figsize=(6,4.2))
    p = fig.subplot(1, 1, 1)
    p.set_xlabel('\( V_g/U \)', fontsize=20)
    p.set_ylabel('\( V/U \)', fontsize=20)
    p.set_title(title, fontsize=20)
    p.im = plt.imshow(stab_cond.T, extent=[xmin, xmax, ymin, ymax],
                     aspect='auto',
                     origin='lower',
                     cmap=plt.get_cmap('Spectral'))
    cbar = plt.colorbar(p.im)
```

2.5
2.0
1.5
1.0
0.5
0.0
0.5
1.0
1.5
2.0
Vg/U
V/U
Pauli, B = 0
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Conductance dI/dV

Figure 3: Stability diagrams calculated using the Pauli approach for a spinful single orbital quantum dot without ($B = 0$) and with ($B = 0.375$) magnetic field. Other parameters are $\Gamma_L = \Gamma_R = T/2, U = 20T, D = 60T$.

In stab_calc we changed the single-particle Hamiltonian by calling the function system.change and specifying which matrix elements to change. The function system.add adds a value to a specified parameter. Also the option masterq=False in system.solve indicates just to diagonalise the quantum dot Hamiltonian, but not to solve the master equation. Similarly, the option qdq=False means that the quantum dot Hamiltonian is not diagonalized (it was already diagonalized previously) and just master equation is solved. Let us now produce the stability diagram

```python
system.kerntype = 'Pauli'
vpnt, vgpnt = 201, 201
vlst = np.linspace(-2*U, 2*U, vpnt)
vglst = np.linspace(-2.5*U, 1.5*U, vgpnt)
stab, stab_cond = stab_calc(system, bfield, vlst, vglst)
stab_plot(stab_cond, vlst, vglst, U, gam, 'Pauli, B = 0')
```

The result is shown in the left plot of Figure 3.

If a Zeeman splitting of the orbital is included, we obtain a stability diagram where the spin-split excited states can be seen:

```python
stab_b, stab_cond_b = stab_calc(system, 7.5, vlst, vglst)
stab_plot(stab_cond_b, vlst, vglst, U, gam, 'Pauli, B = 0.375U')
```

which is shown in the right plot of Figure 3.

Now we demonstrate the difference between the 2vN and the Pauli approaches by considering the bias dependence of the conductance $dI/dV$ at the particle-hole symmetry point $V_g = -U/2$ for $B = 0.375U$:

```python
def trace_vbias(system, vlst, vglate, bfield, dV=0.01, niter=7):
    vpnt = vlst.shape[0]
    trace = np.zeros(vpnt)
    #
    system.change(hsingle=((0, 0): vglate+bfield/2, (1, 1): vglate-bfield/2))
    system.solve(masterq=False)
    #
    for j1 in range(vpnt):
        system.change(mulst=[0: vlst[j1]/2, 1: -vlst[j1]/2,
```
Figure 4: Left plot: bias trace at the particle-hole symmetric point $V_g = -U/2$ for $B = 0.375U$. Compared to the first-order Pauli master equation the 2vN approach additionally yields a cotunneling conductance step appearing around $V = B$. Right plot: stability diagram calculated using the 2vN approach. Other parameters are $\Gamma_L = \Gamma_R = T/2$, $U = 20T$, $D = 60T$.

```python
2: vlst[j1]/2, 3: -vlst[j1]/2)
system.solve(qdq=False, niter=niter)
trace[j1] = (system.current[0] + system.current[2])
# system.add(mult={0: dV/2, 1: -dV/2,
2: dV/2, 3: -dV/2})
system.solve(qdq=False, niter=niter)
trace[j1] = (system.current[0] + system.current[2] - trace[j1])/dV
return trace

vpnt = 101
vlst = np.linspace(0, 2*U, vpnt)
trace_Pauli = trace_vbias(system, vlst, -U/2, 7.5)
trace_2vN = trace_vbias(system2vN, vlst, -U/2, 7.5)

fig = plt.figure()
p = plt.subplot(1, 1, 1)
p.set_xlabel('$V/U$', fontsize=20)
p.set_ylabel('$dI/dV$', fontsize=20)
plt.plot(vlst/U, trace_Pauli, label='Pauli',
color='blue',
lw=2)
plt.plot(vlst/U, trace_2vN, label='2vN',
color='black',
lw=2,
linestyle='--')
plt.legend(loc=2, fontsize=20)
plt.show()
The outcome is shown in the left plot of Figure 4. We see that the $2vN$ approach captures the effect of cotunneling (conductance step appearing around $V = B$), while the first-order Pauli master equation does not describe this effect.

Finally, we calculate the stability diagram for finite magnetic field using $2vN$ approach:

```
vpnt, vgpnt = 201, 201
vlst = np.linspace(-2*U, 2*U, vpnt)
vglst = np.linspace(-2.5*U, 1.5*U, vgpnt)
stab_b_2vN, stab_cond_b_2vN = stab_calc(system2vN, 7.5, vlst, vglst)
stab_plot(stab_cond_b_2vN, vlst, vglst, U, gam, '2vN, B = 0.375U')
```

which is shown in the right plot of Figure 4.

5. Example 2: Spinless double quantum dot

Our second example is spinless serial double quantum dot described by the Hamiltonian (see Figure 5):

$$H_{\text{two}} = \sum_{k,l;\ell} \epsilon_{\ell k} c_{\ell k}^\dagger c_{\ell k} + \sum_k (t_{dL}^d c_{Lk}^\dagger c_{Rk} + t_{dR}^d c_{Rk}^\dagger c_{Lk} + \text{H.c.}) + V_g (d_{L1}^d d_{L2}^d + d_{R1}^d d_{R2}^d + \text{H.c.}) + \Omega (d_{L1}^d d_{R1}^d + \text{H.c.}) + U d_{L1}^d d_{R1}^d.$$ (20)

As before we setup the system using Builder:

```
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq

# Quantum dot parameters
vgate, omega, U = 0.0, 2.0, 5.0
# Lead parameters
vbias, temp, dband = 0.5, 2.0, 60.0
# Tunneling amplitudes
gam = 1.0
t0 = np.sqrt(gam / (2*np.pi))
nsingle = 2

hsingle = {(0,0): vgate, (1,1): vgate, (0,1): omega}
coulomb = {(0,1,1,0): U}
tleads = {(0,0): t0, # L ---< L
           (1,1): t0} # R ---< R
nleads = 2
mulst = {0: vbias/2, 1: -vbias/2}
tlst = {0: temp, 1: temp}

system = qmeq.Builder(nsingle, hsingle, coulomb, nleads, tleads, mulst, tlst, dband)
```

We are interested in how the current depends the gate voltage $V_g$ and the hybridisation between the dots $\Omega$, which we determine using the function:
Occupation
0 1
0
1
Figure 5: Spinless double quantum dot structure. The energies of the dot states are shifted by a gate voltage $V_g$. Both dots are coupled to each other ($\Omega$) and to one lead each ($\Gamma$). The two leads have an applied bias voltage $V$, which gives a particle current $I$.

```python
def omega_vg(system, olst, vglst):
opnt, vgpnt = olst.shape[0], olst.shape[0]
mtr = np.zeros((opnt, vgpnt), dtype=float)
for j1 in range(opnt):
    system.change(hsingle=((0, 1): olst[j1]))
    for j2 in range(vgpnt):
        system.change(hsingle=((0, 0): vglst[j2],
                              (1, 1): vglst[j2]))
    system.solve()
mtr[j1, j2] = system.current[0]
return mtr
```

The plotting will be performed using:

```python
def plot_omega_vg(mtr, olst, vglst, U, gam, kerntype, itype, num):
    (xmin, xmax, ymin, ymax) = (vglst[0]/U, vglst[-1]/U,
                              olst[0]/gam, olst[-1]/gam)
p = plt.subplot(2, 2, num)
p.set_xlabel('\(V_g/\Gamma\)', fontsize=20)
p.set_ylabel('\(\Omega/\Gamma\)', fontsize=20)
p.set_title(kerntype+'$, itype='+str(itype), fontsize=20)
p.im = plt.imshow(mtr/gam, extent=[xmin, xmax, ymin, ymax],
                 vmin=0, vmax=0.023,
                 aspect='auto',
                 origin='lower',
                 cmap=plt.get_cmap('Spectral'))
cbar = plt.colorbar(p.im)
cbar.set_label('Current [\(\Gamma\)]', fontsize=20)
cbar.set_ticks(np.linspace(0.0, 0.03, 4))
plt.tight_layout()
```

Here, for the calculations we use the *Pauli* approach and the *1vN* approach, with three different values 0, 1, and 2 for Builder’s optional argument *itype*. The value of *itype* determines how the integral of Eq. (C.5) is treated. In QmeQ for *itype=2*, the principal part $P$ integrals are neglected, for *itype=1* the integrals are approximated using a digamma function [52], and for *itype=0* a full calculation is performed using the SciPy implementation of the QUADPACK routine DQAWC. Now we calculate the current dependence on $V_g$ and $\Omega$ using the *Pauli* and the *1vN* methods:

```python
opnt, vgpnt = 201, 201
olst = np.linspace(0., 5., opnt)
```
Figure 6: The dependence of the current on $V_g$ and $\Omega$ for a spinless double quantum dot calculated using the Pauli and the 1vN approaches. The values of the parameters are $\Gamma = T/2$, $U = 5T/2$, $D = 30T$.

The outcome of the calculation is shown in Figure 6. We see that the Pauli master equation gives the wrong current for $\Omega < \Gamma$, where the role of coherences is relevant and is correctly captured in 1vN approach (the Redfield, Lindblad, and 2vN approaches give similar results for the considered parameter values).
6. Example 3: Spinful serial triple-dot structure

As a last example, we consider a system with many-degrees of freedom in the quantum dot region: the spinful serial triple-dot structure shown in Figure 7. The parameters for the model are summarized in Table 1. In Ref. [34] the effect of different kinds of Coulomb matrix elements were considered in such a system. For the purposes of an example we here restrict ourselves to the dipole-dipole scattering of electrons between different quantum dots ($U_{sc}$). This term is responsible for the Auger process and is crucial for the current flow in the system.

Now we setup the system:

```python
# Prerequisites
import matplotlib.pyplot as plt
import numpy as np
import qmeq

# Lead and tunneling parameters
mul, mur = 50.0, 10.0
gam, temp, dband = 0.1, 1.0, np.power(10.0, 4)

tl, tr = np.sqrt(gam / (2 * np.pi)), np.sqrt(gam / (2 * np.pi))
tleads = {0: +tl, 1: +tl, 2: -tr, 3: +tl, 4: +tl}

nleads = 4
mulst = {0: mul, 1: mur, 2: mul, 3: mur}
tlst = {0: temp, 1: temp, 2: temp, 3: temp}

# Quantum dot single-particle Hamiltonian
nsingle = 10
e0, e1, e2, e3, e4 = 60, 40, 38, 20, 20
o02, o03, o12, o13, o24, o34 = 0.2, 0.1, 0.1, -0.05, 0.2, 0.1

# Spin up Hamiltonian
hsingle0 = np.array([[e0, 0, o02, o03, 0],
                      [0, e1, o12, o13, 0],
                      [o02, o12, e2, 0, o24],
                      [o03, o13, 0, e3, o34],
                      [0, 0, o24, o34, e4]])

# Augment the Hamiltonian to have spin up and down
hsingle = np.kron(np.eye(2), hsingle0)

# Coulomb matrix elements
usc = -0.2
coulomb = {(0,2,3,1):usc, (0,3,2,1):usc, (0,7,8,1):usc, (0,8,7,1):usc, (1,2,3,0):usc, (1,3,2,0):usc, (1,7,8,0):usc, (1,8,7,0):usc, (2,5,6,3):usc, (2,6,5,3):usc, (3,5,6,2):usc, (3,6,5,2):usc, (5,7,8,6):usc, (5,8,7,6):usc, (6,7,8,5):usc, (6,8,7,5):usc}

system = qmeq.Builder(nsingle, hsingle, coulomb, nleads, tleads, mulst, tlst, dband, indexing='ssq', kerntype='Pauli', itype=2)
```

Here we use the spin symmetry of the problem by setting `indexing='ssq'`, which is very relevant for reducing the Liouvillian size and memory consumption in the 1vN approach calculations. Also we neglect the principal part...
Figure 7: A spinful serial triple-dot structure studied in Ref. [34]. It has 10 single-particle states and 1024 many-body states in total.

| $E_0 = E_5$ | $U_{02} = \Omega_{57} = 0.2$ |
| $E_1 = E_6$ | $\Gamma = 0.1$  $\Omega_{13} = \Omega_{68} = 0.1$ |
| $E_2 = E_7$ | $\mu_L = 50$  $\Omega_{12} = \Omega_{67} = 0.1$ |
| $E_3 = E_8$ | $\mu_R = 10$  $\Omega_{13} = \Omega_{68} = -0.05$ |
| $E_4 = E_9$ | $T = 1$ $\Omega_{23} = \Omega_{79} = 0.2$ |
| $D = 10^4$ | $\Omega_{34} = \Omega_{89} = 0.1$ |

Table 1: Parameters used in the calculations of transport through a triple dot.

integrals by setting `itype=2`. Additionally, we have set up the single-particle Hamiltonian using a NumPy array.\(^7\)

Assume we are interested in how the current depends on the level position $E_3 = E_8$, which we calculate using:

```python
def trace_e3(system, e3lst, removeq=False, dE=150.0):
    e3pnt = e3lst.shape[0]
    trace = np.zeros(e3pnt)
    system.use_all_states()
    for j1 in range(e3pnt):
        system.change({(3, 3): e3lst[j1], (8, 8): e3lst[j1]})
        system.solve(masterq=False)
        if removeq:
            system.remove_states(dE)
        system.solve(qdq=False)
        trace[j1] = sum(system.current[np.ix_([0, 2])])
    return trace
```

```python
e3pnt = 201
e3lst = np.linspace(15., 25., e3pnt)

system.kerntype = 'Pauli'
tr_e3_Pauli = trace_e3(system, e3lst)
tr_e3_Pauli_rm = trace_e3(system, e3lst, removeq=True)

system.kerntype = '1vN'
tr_e3_1vN = trace_e3(system, e3lst)
tr_e3_1vN_rm = trace_e3(system, e3lst, removeq=True)
```

\(^7\)For more information on allowed input types in QmeQ see the 00 types.ipynb notebook.
Figure 8: The dependence of the current on the level position $E_3 = E_8$ for the serial triple-dot structure. The thin curves “Pauli (reduced)” and “1vN (reduced)” correspond to the calculation when the many-body states with energy $\Delta E = 150T$ above the ground state are removed. For the $1vN$ calculation this removal considerably reduces the Liouvillian size and still gives reasonable results.

The result is shown in Figure 8. We see that the current is considerably reduced in the $1vN$ approach compared to the first-order Pauli master equation. This reduction is the effect of the coherences developing between the many-body states which have energy differences smaller than the tunneling rate $\Gamma$, similarly as discussed in the simple double-dot example in Section 5.

An important thing to note is how we used the spin symmetry (indexing='ssq'). In order to be able to use the spin symmetry in the problem the single particle Hamiltonian, tunneling amplitudes, and the lead parameters need to be set up in a particular way. Let us say we have $n = 2m$ quantum dot single particle states counting with spin. Then we use the convention that the states with spin up have the labels $0 \ldots m-1$ and the states with spin down have the labels $m \ldots n-1$. Also there should be no coupling between the up and down states to obtain correct results. Additionally, when the spin degeneracy is present and we want to have a number $m_{\alpha}$ of physical leads, then we need to specify parameters for $n_{\alpha} = 2m_{\alpha}$ number of leads. For example, if there are source and drain leads with chemical potentials $\mu_L$ and $\mu_R$, then we need to specify the following dictionary for chemical potentials:

```python
mulst = {0: $\mu_L$, 1: $\mu_R$, # spin up
        2: $\mu_L$, 3: $\mu_R$} # spin down
```

For more information on usage of symmetries in QmeQ see the 01_symmetries.ipynb notebook.
Now we shortly discuss how the number of $\Phi^{[0]}$ matrix elements is reduced using the spin symmetry. The many-body eigenstates $|b\rangle = |N,S,M,i\rangle$ are classified by the number of particles $N$, the total spin value $S$, and the spin-projection value $M$, and $i$ denotes some other quantum numbers. In the stationary state, due to selection rules the only non-vanishing reduced density matrix $\Phi^{[0]}_i$ elements are between the states $|b\rangle = |N,S,M,i\rangle$ and $|b'\rangle = |N,S,M,i'\rangle$ for arbitrary $i$ and $i'$. Additionally, all the matrix elements are equal when the states have the same $N$, $S$, $i$, and $i'$ quantum numbers, e.g., $\Phi^{[0]}_i|N,S,M_0,0\rangle\langle N,S,M_0,0| = \Phi^{[0]}_{i'}|N,S,M_0,0\rangle\langle N,S,M_0,0|$

Lastly, to manually specify all the matrix elements related by spin symmetry can be tedious and error prone. For this reason it is possible to set up the system by giving only the values for spin-up direction and using optional argument `symmetry='spin'` in the Builder. This will automatically augment the parameters to be consistent with the spin symmetry and will use `indexing='ssq'` if it is supported by the implementation of the approach. Then for the serial triple dot considered above we can write:

```python
tleads = {(0,0): +tl, (0,1): +tl, (1,4): -tr}
mulst = {0: mul, 1: mur}
tlst = {0: temp, 1: temp}
coulomb = {{0,2,3,1}:usc, (0,3,2,1):usc, (1,3,2,0):usc, (1,2,3,0):usc}
system = qmeq.Builder(nsingle , hsingle0 , coulomb ,
    leads, tleads, mulst, tlst, dlist,
    kerntype='Pauli', itype=2,
    symmetry='spin')
```

We also note that if a direct interaction element like $Ud_0^i d_0^i = Ud_0^i d_0^i d_0^i d_0^i$ is used, then the user needs to specify $Ud_0^i d_0^i d_0^i d_0^i$ in `coulomb`

7. Computational cost

In this section we discuss the memory requirements for different approaches and computation time for examples considered in this paper. For $n$ single-particle states we have

* $N = 2^n$ many-body states,
* $N_0 = \frac{2^{2n}}{2n!}$ reduced density matrix $\Phi^{[0]}$ elements,
* $N_1 = \frac{n}{n+1} N_0$ current amplitude $\Phi^{[1]}$ elements.

Here the estimate of $\Phi^{[0]}$ is done assuming that there are no coherences between the many-body states containing different number of particles, for example, $\Phi^{[0]}_{00} = 0$, $\Phi^{[0]}_{01} = 0$, and etc. When the Hamiltonian conserves the total number of particles such off-diagonal elements decay rapidly due to superselection rules and are not present in the stationary state [53]. Additionally, in the considered approaches when the reduced density matrix is diagonal in some conserved quantity (e.g., charge) at some point in time (e.g., in the infinite past), it will remain diagonal for all times. The estimate of $\Phi^{[1]}$ is done by counting only the matrix elements of the form $\Phi^{[1]}_{ii'}$.

To store the kernel for the first-order approaches or the solution (with related quantities) for the $2vN$ approach we need:

* $64 \times N^2$ bits for Pauli,
* $64 \times (2N_0 - N)^2$ bits for $1vN$, Redfield, Lindblad,
* $4 \times 128 \times N_0 N_0 N_1$ bits for $2vN$,

where we assumed 64-bit floating point arithmetics. Here $N_0$ is the number of lead channels and $N_1$ is the number of energy grid points. If all many-body states are kept the memory requirement depending on the number of single-particle states $n$ is shown in Figure 9 (for the $2vN$ approach we used $N_0 = 2$, $N_1 = 2^{12}$). For example, if 8GB of
memory is available then it is possible to solve a system with the following number of single-particle states: 15 for Pauli, 8 for 1vN, 4 for 2vN.\(^9\)

The spinful serial triple-dot structure (see Section 6) with \(n = 10\) single-particle states is an example, which requires a lot of memory. With all the many-body states present the spin-symmetry of the problem allows to reduce the memory requirement from 1TB to 3GB for the 1vN approach. The calculation for a single \(E_3\) point (see Figure 8) takes 170 seconds on our computer based on 3.50GHz processor (Intel Xeon E3-1270v3). For the calculation, where the many-body states with energy \(\Delta E = 150T\) above the ground state are removed, the memory requirement is reduced to 0.1GB and computation time is reduced to 4 seconds.

In the spinful single orbital example (see Section 6) the 2vN approach takes 24 seconds to calculate a single point of current on our computer with 3.50GHz processor. So it takes around 1.5 hour to produce the left plot of Figure 4 with 2 \(\times\) 101 points of calculation and around 540 hours to produce the right plot of Figure 4 with 2 \(\times\) 80802 points of calculation. We note that because of slow speed and large memory requirement for larger systems (see Figure 9) the 2vN approach is useful only for describing systems with small number of many-body states.

### 8. Remark on units

The inputted tunneling amplitudes \(t_{\alpha i}\) in \(t1\)eads correspond to tunneling amplitudes \(t_{\alpha i}\) weighted by the density of states \(\nu_F\) in the following way:

\[
t_{\alpha i} = \sqrt{\nu_F t_{\alpha i}}.\tag{21}
\]

Then the tunneling rates simply become \(\Gamma_{\alpha i} = 2\pi|t_{\alpha i}|^2\). When different lead channels can have different density of states \(\nu_{\alpha i}\), any difference can be absorbed into the tunneling amplitudes. Thus it was not necessary to specify \(\nu_F\) in the given examples.

\(^9\)We note that for the first-order approaches it is possible to use matrix-free methods for solution by specifying an optional argument \(\text{Builder(..., mfreeq=True)}\). Then the memory requirement is much lower and is proportional to \(N\) or \(N_0\) for Pauli or 1vN, Redfield, Lindblad approaches, respectively. However, the computation time increases considerably.
Throughout the calculations we have used $\hbar = 1$ and considered particle currents instead of electrical currents. If we are interested in the case $\hbar/\neq 1$, $|e|/\neq 1$ and in electrical currents with carriers having a charge $e$ (which can be $e < 0$) we have to make the following changes in Figures 3–8:

\begin{align*}
V &\to eV, \\
V_g &\to eV_g, \\
I[\Gamma] &\to I[e\Gamma/\hbar], \\
dI/dV[1] &\to dI/dV[e^2/\hbar].
\end{align*}

To get the conductance in units of $G_0 = e^2/\hbar$, we should multiply the $dI/dV$ plots of Figures 3 and 4 by $2\pi$.

9. Conclusion

We have presented an open-source Python package QmeQ for numerical modeling of stationary state transport through quantum dot devices, which can be described by the tunneling model (1) with quantum dots having arbitrary Coulomb interactions. The main objective of the package is to provide numerical results based on various approximate master equation approaches. We have introduced and explained the features of the package in an example driven way. The version of the package described in this paper is QmeQ 1.0.

Acknowledgements

We thank K. M. Seja, F. A. Damtie, B. Goldozian, and K. G. L. Pedersen for useful discussions. Financial support from the Swedish Research Council (VR) and NanoLund is gratefully acknowledged.

Appendix A. Approximate master equations

In the Appendices we present the derivation of the approximate master equations implemented in QmeQ and discuss their properties. These are the second-order von Neumann (Appendix B), the first-order von Neumann (Appendix C), the first-order Redfield (Appendix D), and the Pauli (Appendix E) approaches. We also shortly describe a particular form of the Lindblad equation in Appendix F, which takes first-order processes into consideration. It is similar to the first-order Redfield or first-order von Neumann methods, but additionally preserves positivity of the reduced density matrix [32]. In Appendix G we give suggestions in which cases which approach to use.

We are interested in solving the von Neumann equation approximately. This equation describes the evolution of the density matrix $\rho$ of the whole system:

\[ i\hbar \frac{\partial \rho}{\partial t} = [H, \rho]. \]  

Our derivation of the approaches is based on the hierarchical method, where we classify the density matrix elements by the number of electron/hole excitations in the leads (see Figure A.10). Such a derivation was originally presented.
in Refs. [19, 30], but most of this Appendix is based on Refs. [34, 54]. Here we derive the necessary equations to define the second-order von Neumann approach. Lower-order approaches will be deductively obtained in succeeding sections by making additional approximations to these equations. Density matrix elements are defined and classified as

\[ \rho^{(n)}_{bg} = (ag|bg'), \]  

(A.2)

where \(|bg\rangle = |b\rangle \otimes |g\rangle\), with \(|b\rangle\) denoting the eigenstate of the dot Hamiltonian \(H_{\text{dot}}\) (3) and \(|g\rangle\) denoting the eigenstate of the lead Hamiltonian \(H_{\text{lead}}\) (2). Here the label \(n\) provides the number of electron or hole excitations needed to transform \(|g\rangle\) into \(|g'\rangle\). For example, we consider the matrix elements of the type

\[\begin{align*}
\rho_{bg,bg}^{[0]} &= \langle bg|bg'\rangle, \\
\rho_{bg,bg}^{[1]} &= \langle dg - \kappa|bg\rangle, \\
\rho_{bg,bg}^{[2]} &= \langle bg - \kappa'|bg'\rangle.
\end{align*}\]  

(A.3)

Here we have introduced a composite index

\[ \kappa \equiv k, \alpha; \]  

(A.4)

and the following notation

\[ \langle bg + \kappa| = |b\rangle \otimes |c^+\rangle, \quad \langle dg - \kappa'| = |d\rangle \otimes |c^+\rangle, \]  

\[ \langle bg - \kappa| = |b\rangle \otimes |c^-\rangle, \quad \langle bg - \kappa + \kappa'| = |b\rangle \otimes |c^+\rangle. \]  

(A.5)

By neglecting all the density matrix elements with more than two electron or hole excitations \(n > 2\) from Eq. (A.1) we obtain the equations

\[\begin{align*}
i \frac{\partial}{\partial t} \rho_{bg,bg}^{[0]} &= \langle E_b - E_g - E_d\rangle \rho_{bg,bg}^{[0]} + T_{b_1,a_1}^{[1]} \rho_{b_1\gamma + e_1\delta - b_1\gamma}^{[1]} \langle -1 \rangle \rho_{b_1\gamma - e_1\delta + b_1\gamma} - \rho_{b_1\gamma}^{[1]} \rho_{b_1\gamma + e_1\delta - b_1\gamma}^{[1]} \langle -1 \rangle \rho_{b_1\gamma} T_{c_1\beta',c_1\beta}, \\
i \frac{\partial}{\partial t} \rho_{bg,bg}^{[1]} &= \langle E_c - E_b + E_d\rangle \rho_{bg,bg}^{[1]} + T_{c_1\beta,\beta'}^{[2]} \rho_{b_1\gamma + e_1\delta}^{[1]} \langle -1 \rangle \rho_{b_1\gamma - e_1\delta + b_1\gamma} - \rho_{b_1\gamma}^{[2]} \rho_{b_1\gamma + e_1\delta}^{[1]} \langle -1 \rangle \rho_{b_1\gamma} T_{c_1\beta',c_1\beta}, \\
i \frac{\partial}{\partial t} \rho_{bg,bg}^{[2]} &= \langle E_d - E_c - E_b\rangle \rho_{bg,bg}^{[2]} + T_{c_1\beta,\beta'}^{[2]} \rho_{b_1\gamma + e_1\delta}^{[1]} \langle -1 \rangle \rho_{b_1\gamma - e_1\delta + b_1\gamma} - \rho_{b_1\gamma}^{[2]} \rho_{b_1\gamma + e_1\delta}^{[1]} \langle -1 \rangle \rho_{b_1\gamma} T_{c_1\beta',c_1\beta}.
\end{align*}\]  

(A.6)  

(A.7)  

(A.8)

Note our convention, that we sum over all indices with subscript 1, e.g., \(a_1, c_1, \kappa_1\). Additionally, phase factors like \((-1)^{N_b}\) appear due to order exchange of the lead operators with the dot operators, i.e., \(c_1\langle (b) \otimes |g\rangle = \langle -1 \rangle \langle b \otimes c^+\rangle\).

Summing Eqs. (A.6) and (A.7) over all the lead states \(|g\rangle\) we get

\[\begin{align*}
i \frac{\partial}{\partial t} \Phi_{bg}^{[0]} &= \langle E_b - E_g\rangle \Phi_{bg}^{[0]} + T_{b_1,a_1} \Phi_{a_1\beta',\beta_1}^{[1]} + T_{b_1,\kappa_1} \Phi_{e_1\kappa,\beta'_1}^{[1]} - \Phi_{b_1,\kappa_1} T_{c_1\beta',\beta_1} - \Phi_{b_1,\kappa_1} T_{a_1\beta',\beta_1}, \\
i \frac{\partial}{\partial t} \Phi_{bg}^{[1]} &= \langle E_c - E_b\rangle \Phi_{bg}^{[1]} + T_{c_1\beta,\beta'} \Phi_{b_1\gamma + e_1\delta}^{[1]} - \Phi_{c_1\beta,\beta'} T_{c_1\beta',c_1\beta} - \Phi_{c_1\beta,\beta'} T_{a_1\beta',\beta_1}.
\end{align*}\]  

(A.9)  

(A.10)  

(A.11)
\[
\frac{\partial}{\partial t} \Phi^{[2]}_{bb',-\kappa+\kappa'} \approx (E_b - \epsilon_k + \epsilon_{k'} - E_{b'}) \Phi^{[1]}_{bb',-\kappa+\kappa'} f_k + T_{bc',\kappa} \Phi^{[1]}_{c_b',\kappa} f_{-c_k},
\]
(A.12)

\[
\frac{\partial}{\partial t} \Phi^{[2]}_{db,-\kappa'} \approx (E_d - \epsilon_k - \epsilon_{k'} - E_b) \Phi^{[2]}_{db,-\kappa'} - T_{ba1,\kappa} \Phi^{[1]}_{ba1,\kappa} f_{\kappa},
\]
(A.13)

where we introduced the following notation

\[
\Phi^{[0]}_{\alpha\beta} = \sum_g \rho^{[0]}_{\alpha\beta g}, \quad \Phi^{[1]}_{\alpha\beta} = \sum_g \rho^{[1]}_{\alpha\beta g} (-1)^{N_g}, \quad \Phi^{[1]}_{\alpha\beta} = [\Phi^{[1]}_{\alpha\beta}]',
\]
(A.14)

\[f_k \equiv f_{ba} = (\exp((\epsilon_k - \mu_a)/T_a) + 1)^{-1}, \quad f_{-k} \equiv f_{bc} = 1 - f_{ba}.
\]

Going from Eq. (A.7) to Eq. (A.11) we also used

\[
\rho^{[2]}_{bb',-\kappa+\kappa'} = \rho^{[0]}_{bb'} + (1 - \delta_{kk'}) \rho^{[2]}_{bb',\kappa} + \sum_{\kappa''} \rho^{[2]}_{bb',-\kappa+\kappa''},
\]
(A.15)

We note that \(\Phi^{[0]}_{bb'}\) represents the reduced density matrix of the quantum dot. Here we have also assumed that the electrons in the leads are thermally distributed according to the Fermi-Dirac distribution \(f\) and that this distribution is not affected by the coupling to the quantum dots. This assumption leads to the following relations for Eq. (A.7)

\[
\sum_g \rho^{[0]}_{bb'g,\kappa} = f_{bb}, \quad \sum_g \rho^{[0]}_{bb',\kappa+\kappa-g} = f_{-\kappa} \Phi^{[0]}_{cc},
\]
(A.16)

for Eq. (A.8)

\[
\sum_g \rho^{[1]}_{a_b\kappa+\kappa'+\kappa'-g} (-1)^{N_g} \approx -f_{-\kappa} \Phi^{[1]}_{cc}, \quad \sum_g \rho^{[1]}_{c_b\kappa,\kappa'} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}',
\]

\[
\sum_g \rho^{[1]}_{b_g\kappa'-\kappa,\kappa'-g} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}, \quad \sum_g \rho^{[1]}_{b_g\kappa,\kappa'} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}'.
\]

and for Eq. (A.9)

\[
\sum_g \rho^{[1]}_{c_b\kappa+\kappa'-\kappa'-g} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}, \quad \sum_g \rho^{[1]}_{c_b\kappa,\kappa'} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}',
\]

\[
\sum_g \rho^{[1]}_{d_g\kappa'-\kappa,\kappa'-g} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}, \quad \sum_g \rho^{[1]}_{d_g\kappa,\kappa'} (-1)^{N_g} \approx f_{-\kappa} \Phi^{[1]}_{cc}'.
\]
(A.17)

Equations (A.10)-(A.13) are the central equations for deriving approximate master equation implemented in Q-sim.

**Appendix B. Solution of the second-order von Neumann approach equations**

In the second-order von Neumann (2\(vN\)) approach we use the approximations:

1. Only terms involving up to two excitations \(n = 2\) are kept (\(\Phi^{[3]} \rightarrow 0\)).
2. A Markov approximation is made to \(\Phi^{[2]}\).
3. Electrons in the leads are thermally distributed according to the Fermi-Dirac distribution \(f\).
Pros (+) and cons (-) of this approach:

+ Can describe sequential tunneling, cotunneling, pair-tunneling, and broadening effects [19].
+ Yields exact currents for non-interacting systems with $H_{\text{Coulomb}} = 0$ [18, 55].
- With interactions ($H_{\text{Coulomb}} \neq 0$) the results can only be trusted up to moderate coupling strengths $\Gamma \lesssim T$. Also the temperature $T$ needs to be larger than any Kondo temperature $T_K$ in the system [11, 19, 56].
- Can violate the positivity of the reduced density matrix $\Phi^{[0]}$ and does not satisfy the Onsager relations [57], when $H_{\text{Coulomb}} \neq 0$. However, negative occupations are rare compared to the first-order approaches addressed below and occur only for strong couplings. Similarly, the deviations from Onsager’s theorem are of third order in the rates ($\Gamma^3$) as shown in [57]. Thus these features may be used to monitor the validity of results with increasing tunnel coupling.
- Has long calculation times and large memory consumption compared to the first-order approaches .

After solving the linear inhomogeneous differential equations (A.12) and (A.13) we obtain:

$$
\Phi_{bb',-\kappa \pm \kappa'}^{[2]}(t) = \frac{1}{i} \int_{-\infty}^t dt' e^{i \epsilon_n - \epsilon_{n'} - E_n + E_{n'} + i\eta(t-t')} \left( -T_{ba_1,\kappa} \Phi_{a_1b_1,\kappa'}^{[1]}(t') f_{n'} + T_{bc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]}(t') f_{n'} - \Phi_{bc_1,\kappa'}^{[1]}(t') T_{c_1b_1,\kappa} f_{n'} + \Phi_{bb_1,\kappa}^{[1]}(t') T_{b_1b_1,\kappa} f_{n'} \right) .
$$

(B.1)

$$
\Phi_{db',-\kappa \pm \kappa'}^{[2]}(t) = \frac{1}{i} \int_{-\infty}^t dt' e^{i \epsilon_n - \epsilon_{n'} - E_n + E_{n'} + i\eta(t-t')} \left( -T_{dc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]}(t') f_{n'} + T_{dc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]}(t') f_{n'} - \Phi_{dc_1,\kappa'}^{[1]}(t') T_{c_1b_1,\kappa} f_{n'} + \Phi_{dd_1,\kappa}^{[1]}(t') T_{d_1d_1,\kappa} f_{n'} \right) .
$$

(B.2)

Here we have added a positive infinitesimal $\eta = +0$ to ensure a proper decay of initial conditions. Here we neglect the memory by replacing $t'$ by $t$ in the $\Phi^{[1]}$ functions of the integrals (Markov approximation). Then the integration of the exponential factor provides the final result for $\Phi^{[2]}$ functions:

$$
\Phi_{bb',-\kappa \pm \kappa'}^{[2]} = -T_{ba_1,\kappa} \Phi_{a_1b_1,\kappa'}^{[1]} f_{n'} + T_{bc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]} f_{n'} - \Phi_{bc_1,\kappa'}^{[1]} T_{c_1b_1,\kappa} f_{n'} + \Phi_{bb_1,\kappa}^{[1]} T_{b_1b_1,\kappa} f_{n'},
$$

(B.3)

$$
\Phi_{db',-\kappa \pm \kappa'}^{[2]} = -T_{dc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]} f_{n'} + T_{dc_1,\kappa} \Phi_{c_1b_1,\kappa'}^{[1]} f_{n'} - \Phi_{dc_1,\kappa'}^{[1]} T_{c_1b_1,\kappa} f_{n'} + \Phi_{dd_1,\kappa}^{[1]} T_{d_1d_1,\kappa} f_{n'} .
$$

We note that the Markov approximation can be made quicker by this simple substitution:

$$
i \left( \frac{\partial}{\partial t} + \eta \right) \Phi_{bb',-\kappa \pm \kappa'}^{[2]} = 0, \quad i \left( \frac{\partial}{\partial t} + \eta \right) \Phi_{db',-\kappa \pm \kappa'}^{[2]} = 0.
$$

(B.4)

The above relations are also expected to hold in the stationary state. After inserting the above expressions into Eq. (A.11), we obtain the integral equations of the 2vN method (colors are explained below)
\[ \frac{\partial}{\partial t} \Phi_{bb}[0] = (E_b - E_b') \Phi_{bb}[0] + T_{ba,1} \Phi_{a1}[1] - \Phi_{bb}[0] T_{c1b,1} - \Phi_{ba,1}^* T_{ba,1}, \]  
(B.6)

Additionally, we impose the normalisation condition for the diagonal reduced density matrix elements:

\[ \sum_b \Phi_{bb}[0] = 1. \]  
(B.7)

In \( Q_{\neq Q} \) we perform steady-state transport calculations and thus have the additional conditions

\[ i \frac{\partial}{\partial t} \Phi_{bb}[0] = 0, \quad \text{and} \quad i \frac{\partial}{\partial t} \Phi_{cb,1}[1] = 0. \]  
(B.8)

Now we describe the numerical procedure implemented in \( Q_{\neq Q} \) for solving the integral equations (B.5) and (B.6) in the stationary state. This numerical procedure works well for moderate coupling strengths \( \Gamma \lesssim T \) and a large bandwidth \( D \gg T, \Gamma \). Using Eq. (B.8), the integral equation (B.5) can be cast in the following form

\[ \Phi_{\epsilon}[1] = F_{\epsilon} + \sum_{\kappa_1} K_{\epsilon \kappa_1} \Phi_{\kappa_1}[1]. \]  
(B.9)

The function \( F_{\epsilon} \) corresponds to a solution of Eq. (B.5) with a local approximation, i.e., terms of the form \( \Phi_{ba,1}[1] \), which have integrated energy label \( \kappa_1 \) are neglected (colored in red). We solve Eq. (B.9) iteratively with the zeroth iteration given by \( \Phi_{\epsilon}[1][0] = F_{\epsilon} \). Then the first correction is determined as \( \delta \Phi_{\epsilon}[1] = \sum_{\kappa_1} K_{\epsilon \kappa_1} F_{\kappa_1} \). The higher order corrections are given by \( \delta \Phi_{\epsilon}[1] = \sum_{\kappa_1} K_{\epsilon \kappa_1} \delta \Phi_{\kappa_1}[1] \) and the solution is expressed as \( \Phi_{\epsilon}[1] = \Phi_{\epsilon}[1][0] + \sum_{\kappa_1} \delta \Phi_{\epsilon}[1] \). In these iterations we have to evaluate Hilbert transforms of the form:

\[ H(\Phi_{\epsilon}[1]) = \frac{1}{\pi} \int_{-D}^{D} \frac{\Phi_{\epsilon}[1][\epsilon'] d\epsilon'}{\epsilon - \epsilon'} = \frac{1}{\pi} \int_{-D}^{D} \frac{\Phi_{\epsilon}[1][\epsilon'] d\epsilon'}{\epsilon - \epsilon'} \pm i\eta, \]  
(B.10)

The principal value integrals are efficiently evaluated on equidistant grid with \( N \) points with a fast Fourier transform, which has complexity \( O(N \log N) \) [58, 59].

**Appendix C. First-order von Neumann approach**

The first-order von Neumann (1vN) approach is obtained with the following approximations to Eqs. (A.10)-(A.13):

1. Only terms involving up to a single excitation \( n = 1 \) are kept (\( \Phi_{[2]} \to 0 \)).
2. A Markov approximation is made to \( \Phi_{[1]} \).

The properties of the approach are:

+ Can describe sequential tunneling in the presence of coherences.
- Coupling strength has to be considerably smaller than the temperature \( \Gamma \ll T \).
- Can violate the positivity of the reduced density matrix \( \Phi_{[0]} \) [6] and does not satisfy the Onsager relations [57, 60]. Here the same consideration hold as for the 2vN approach, but problems occur at lower coupling strengths. In particular, the deviation from the Onsager theorem is of second order in the rates \( (\Gamma)^2 \).

We formally integrate Eq. (A.11) (with \( \Phi_{[2]} \) neglected) and obtain

\[ \Phi_{cb,1}[1](t) = \frac{1}{T} \int_{-\infty}^{\infty} dt' \delta(\epsilon - E + \epsilon + i\eta) \left( T_{cb,1} \Phi_{bb}[0](t') f_{cb,1}(t') T_{cb,1}^* f_{cb,1} \right). \]  
(C.1)

Here we have added a positive infinitesimal \( \eta = +0 \) to ensure a proper decay of initial conditions. After performing a Markov approximation in the above integral, \( \Phi_{bb}[0](t') \approx \Phi_{bb}[0](t), \) and setting \( t \to +\infty \) we get:

\[ \Phi_{cb,1}[1] = \frac{T_{cb,1} \Phi_{bb}[0] f_{cb,1} - \Phi_{cb,1}[0] T_{cb,1}^* f_{cb,1}}{\epsilon - E + \epsilon + i\eta}. \]  
(C.2)
The same expression (C.2) is obtained if we use $i\partial_t \Phi^{[1]} = 0$. Combining Eqs. (A.10) and (C.2) we get the $1vN$ approach equations:

$$i\frac{\partial}{\partial t} \Phi^{[0]}_{bb'} = (E_b - E_{b'})\Phi^{[0]}_{bb'} + \sum_{b''a'} \Phi^{[0]}_{b'b''} \left[ \sum_c \Gamma^{a}_{b''a, c, b'''} \bar{F}^{+}_{ka'} - \sum_c \Gamma^{a}_{b''a, c, b''} \bar{F}^{+}_{ca'} \right]$$

$$+ \sum_{b''a'} \Phi^{[0]}_{b'b''} \left[ \sum_c \Gamma^{a}_{c, b'' a, b'''} \bar{F}^{+}_{ca''} - \sum_c \Gamma^{a}_{c, b'' a, b''} \bar{F}^{+}_{ca''} \right]$$

$$+ \sum_{a''c} \Phi^{[0]}_{a'a''} \Gamma^{a}_{b''a, c' b'''} \left[ \bar{F}^{+}_{ca''} - \bar{F}^{+}_{ca'} \right] + \sum_{a''c} \Phi^{[0]}_{a'a''} \Gamma^{a}_{c, b'' a, b'''} \left[ \bar{F}^{+}_{ca''} - \bar{F}^{+}_{ca'} \right].$$

(C.3)

with the normalisation condition $\sum_a \Phi^{[0]}_{aa} = 1$. In Eq. (C.3) the tunneling rate matrix $\Gamma$ is defined as

$$\Gamma^{a}_{ba, cb} = 2\pi v_f T_{ba} T_{cb'},$$

(C.4)

and the following integral was introduced

$$2\pi F_{cb} = \int_{-D}^{D} dE f\left( \frac{E - E_{cb}}{\hbar^2} \right) = -\int_{-D}^{D} dE f\left( \frac{E - E_{cb}}{\hbar^2} \right) \ln \left( \frac{D}{2\pi T_a} \right).$$

(C.5)

which appears after performing the $k$-sums using a flat density of states approximation, i.e., $\sum_k \rightarrow v_f \int_{-D}^{D} dE$, with $v_f$ denoting the density of states at the Fermi level and $2D$ denoting the bandwidth of the leads. For very large bandwidth $D \rightarrow \infty$ compared to the other energy scales the principal part integral in (C.5) can be approximated using a digamma function $\Psi$ [52] in the following way

$$-\int_{-D}^{D} dE f\left( \frac{E - E_{cb}}{\hbar^2} \right) \ln \left( \frac{D}{2\pi T_a} \right).$$

(C.6)

This is the expression used for $\text{type}=1$.

For the steady state we have $i\partial_t \Phi^{[0]}_{bb'} = 0$ and thus we can determine $\Phi^{[0]}_{bb'}$ from Eq. (C.3) and the normalisation condition $\sum_a \Phi^{[0]}_{aa} = 1$. Lastly, the particle and energy currents can be expressed as

$$I_a = -2 \sum_{cb} \text{Im} \left[ \sum_{b'} \Gamma^{a}_{b', c b} \bar{F}^{+}_{cb'} \Phi^{[0]}_{b'b} - \sum_{c'} \Gamma^{a}_{b c, c' b} \bar{F}^{+}_{c'b} \Phi^{[0]}_{c'b} \right],$$

(C.8)

$$\dot{E}_a = -2 \sum_{cb} \text{Im} \left[ \sum_{b'} \Gamma^{a}_{b', c b} \bar{F}^{+}_{cb'} \Phi^{[0]}_{b'b} - \sum_{c'} \Gamma^{a}_{b c, c' b} \bar{F}^{+}_{c'b} \Phi^{[0]}_{c'b} \right],$$

where for the energy currents we introduced the following integral

$$\bar{F}^{+}_{cb} = \frac{1}{2\pi} \int_{-D}^{D} dE f\left( \frac{E - E_{cb}}{\hbar^2} \right) = E_{cb} F^{+}_{cb} + \frac{1}{2\pi} \int_{-D}^{D} dE f\left( \frac{E - \mu_a}{\hbar^2} \right).$$

(C.10)

We emphasize that there are many ways to derive the equations referred to here as the $1vN$ approach. For example, identical equations result from the real-time diagrammatic technique [61, 62, 63] with a perturbation expansion truncated at leading order in the tunnel couplings. In contrast, including next-to-leading order contributions in the real-time diagrammatic technique does not give the same result as the $2vN$ approach. Actually, the $2vN$ approach includes some terms up to infinite order in the tunnel couplings, which is equivalent to the resonant tunneling approximation [18] in the diagrammatic approach [55].

30
Appendix D. First-order Redfield approach

The first-order Redfield approach is based on the same approximations as the $1vN$ approach to Eqs. (A.10)-(A.13). Also the Redfield approach has the same pros and cons as the $1vN$ approach. The difference is in the Markov approximation, where we under the integral in Eq. (C.1) set:

$$\Phi^{[0]}_{bb'}(t') \approx e^{i(E_b-E_{b'})t'-i\eta}\Phi^{[0]}_{bb}(t).$$

This gives:

$$\Phi^{[1]}_{cb,a} = \frac{T_{cb,a}\Phi^{[0]}_{bb,f_x}E_x - E_{b_f} + i\eta - \Phi^{[0]}_{cc,f_x}T_{cc,b}f_x}{E_x - E_{c_f} + E_{b_f} + i\eta},$$

(E.1)

Also note that this corresponds to expressing $\Phi^{[0]}$ in the interaction picture under the integral in Eq. (C.1) and then performing the Markov approximation $[\Phi^{[1]}_{cb,a}(t') \approx \Phi^{[1]}_{cb,a}(t)]$. Using Eqs. (A.10) and (D.2) we obtain the Redfield approach equations:

$$\frac{\partial}{\partial t} \Phi^{[0]}_{bb} = (E_b - E_{b'})\Phi^{[0]}_{bb} + \sum_{b'} \Phi^{[0]}_{bb'} \left[ \sum_a \Gamma^{\alpha}_{bb'a} \Phi^{[0]}_{bb'a} - \sum_a \Gamma^{\alpha}_{ba,cbb'} \Phi^{[0]}_{ba,cbb'} \right]$$

$$+ \sum_{b'} \Phi^{[0]}_{bb'c} \left[ \sum_c \Gamma^{\alpha}_{bcab} \Phi^{[0]}_{bcab} - \sum_a \Gamma^{\alpha}_{ba,bb'} \Phi^{[0]}_{ba,bb'} \right]$$

$$+ \sum_{a'c} \Phi^{[0]}_{ab'a} \Gamma^{\alpha}_{ab'a} \left[ \Phi^{[0]}_{bb'} - \Phi^{[0]}_{bb} \right] + \sum_{cc'} \Phi^{[0]}_{cc'} \Gamma^{\alpha}_{cc'} \left[ \Phi^{[0]}_{bb'} - \Phi^{[0]}_{bb} \right].$$

(D.3)

The definitions of $\Gamma$ and $I$ are the same as in Eqs. (C.4) and (C.5). As a result of the different Markov approximation, both the equation of motion and the currents are different from the $1vN$ approach. In QmQ Eq. (D.3) is solved for stationary state $i\partial_t \Phi^{[0]}_{bb} = 0$ together with normalisation condition $\sum_b \Phi^{[0]}_{bb} = 1$.

Using Eq. (D.2) the particle and energy currents become

$$I_a = -2 \sum_{cb} \text{Im} \left[ \sum_{b'} \Gamma^{\alpha}_{bcab} \Phi^{[0]}_{bb'} - \sum_{c'} \Gamma^{\alpha}_{bc,cab} \Phi^{[0]}_{bb'} \right],$$

(D.4)

$$E_a = -2 \sum_{cb} \text{Im} \left[ \sum_{b'} \Gamma^{\alpha}_{bcab} \Phi^{[0]}_{bb'} - \sum_{c'} \Gamma^{\alpha}_{bc,cab} \Phi^{[0]}_{bb'} \right].$$

(D.5)

Appendix E. Pauli master equation

The Pauli master equation is obtained with the following approximations to Eqs. (A.10)-(A.13):

1. Only the terms involving up to single excitation $n = 1$ are kept ($\Phi^{[2]} \rightarrow 0$).
2. The Markov approximation is made to $\Phi^{[1]}$.
3. The coherences of the reduced density matrix $\Phi^{[0]}$ of the quantum dot are neglected ($\Phi^{[0]}_{bb'} = 0$ for $b \neq b'$).

The properties of the approach are:

- Can describe sequential tunneling.
- Preserves the positivity of the reduced density matrix $\Phi^{[0]}$ [6] and satisfies the Onsager relations [64].
- The coupling strength has to be considerably smaller than the temperature $\Gamma \ll T$.
- The energy level splitting $\Delta E$ between the states with the same charge has to be considerably smaller than the coupling strength $\Gamma$ [65, 34].

The Pauli master equation can be obtained from the $1vN$ or the Redfield approaches by neglecting the coherences $\Phi^{[0]}_{bb'}, b \neq b'$. From Eq. (C.3) or (D.3) for the populations $P_b = \Phi^{[0]}_{bb}$ we obtain the equations:

$$\frac{\partial}{\partial t} P_b = \sum_{a_i} \left[ P_{a_i} \Gamma^{\alpha}_{a_i,bb} f(x_{a_i}^{bb}) - P_b \Gamma^{\alpha}_{b,bb} f(-x_{b}^{bb}) \right]$$

$$+ \sum_{c_i} \left[ P_{c_i} \Gamma^{\alpha}_{c_i,bb} f(-x_{c_i}^{bb}) - P_b \Gamma^{\alpha}_{b,bb} f(x_{b}^{bb}) \right],$$

(E.1)
where we have denoted \( \Gamma_{a \to b} = \Gamma_{ab}^{\alpha} = \Gamma_{ba}^{\alpha} = \Gamma_{ba}^{\alpha} \). In Qme\( Q \) Eq. (E.1) is solved for stationary state \( \partial_t P_b = 0 \) together with the normalisation condition \( \sum_b P_b = 1 \). Using the populations \( P_b \) the particle and energy currents are expressed as

\[
I_a = \sum_{ab} \left[ P_b \Gamma_{a \to b} \Phi_{a \to b}(+x_{ab}^a) - P_b \Gamma_{b \to a} \Phi_{b \to a}(-x_{ba}^b) \right], \tag{E.2}
\]

\[
\dot{E}_a = \sum_{bc} \left[ P_b \Gamma_{b \to c} \Phi_{b \to c}(+x_{cb}^a) - P_b \Gamma_{c \to b} \Phi_{c \to b}(-x_{bc}^b) \right], \tag{E.3}
\]

where \( E_{cb} \) and \( x_{cb}^a \) are defined in (C.6).

**Appendix F. Lindblad equation**

The Lindblad equation [31], which we are using in Qme\( Q \), has the following form:

\[
i \partial_t \Phi_{br}^{[0]} = (E_b - E_{br}) \Phi_{br}^{[0]} - \frac{1}{2} \sum_{b'=a} \Phi_{b'r}^{[0]} \left[ \sum_a L_{abr}^a L_{adr}^a + \sum_c L_{cb}^a L_{cbr}^a \right] - \frac{1}{2} \sum_{b'=a} \Phi_{b'r}^{[0]} \left[ \sum_c L_{cb}^a L_{cbr}^a + \sum_b L_{ab} L_{adr}^a \right] + \sum_{a'a} \Phi_{a'a}^{[0]} L_{a'a}^a + \sum_{c'c} \Phi_{c'c}^{[0]} L_{c'c}^a L_{bc}^a, \tag{F.1}\]

where the matrix elements of the jump operators are defined as [32]

\[
L_{cb,a} = \sqrt{2\pi v_f (-x_{cb}^a)} T_{bc,a}, \quad L_{bc,a} = \sqrt{2\pi v_f (+x_{bc}^a)} T_{bc,a}. \tag{F.2}\]

As before we solve Eq. (F.1) for the stationary state \( i \partial_t \Phi_{br}^{[0]} = 0 \) supplemented by the normalisation condition \( \sum_b \Phi_{br}^{[0]} = 1 \). Some of the properties of the approach are:

\begin{itemize}
  \item Equation (F.1) is of the first-order type in the rates \( \Gamma \) and can describe the sequential tunneling in the presence of coherences.
  \item Preserves the positivity of the reduced density matrix \( \Phi^{[0]} \) [31].
  \begin{itemize}
    \item The coupling strength has to be considerably smaller than the temperature \( \Gamma \ll T \).
    \item Does not satisfy the Onsager relations [32]. Here, the same considerations hold as for the 2vN approach, but the deviation from the theorem is of second order in the rates (\( \Gamma^2 \)).
    \item Effects, which can arise due to principal part integrals [see Eq. (C.5)] are not included.
  \end{itemize}
\end{itemize}

**Appendix G. Which approach to use?**

Having different approaches the question “Which approach to use?” can arise. From the pros (+) and cons (-) of different approaches it is clear that if a system is small and cotunneling or pair-tunneling is under interest then the 2vN approach should be used. On the other hand, the Pauli master equation is a reliable choice with low computational cost if only sequential tunneling is of relevance and no coherences between different states develop.

If coherences are important, while tunneling is dominated by sequential events, it is possible to use the IvN, Redfield, or Lindblad approaches. We recommend to use the Lindblad approach in the case, when the effects of principal part integrals [see Eq. (C.5)] are not important, because this approach preserves positivity. At the same time we note that the principal parts are required to catch important physics such as renormalization of energy levels in some systems [66, 67, 68, 69]. By simulating different systems we also observed that with neglected principal parts the IvN and the Redfield approaches give the same results. In Ref. [57] the deviation from the Onsager’s theorem was found to be slightly larger in the IvN approach compared to the Redfield approach (with the principal parts included). However, for the stationary state, the density matrix is constant in time within the Schrödinger picture. Thus for stationary states it appears to be better to do the Markov limit in the Schrödinger picture (the IvN approach). So we cannot give a clear-cut answer for what case the Redfield or IvN approach is better. The clear thing is that for sufficiently weak coupling both approaches do not show substantial differences.