

VOTCA-XTP

EXCITON TRANSPORT SIMULATIONS

USER MANUAL



compiled from: 1.5-dev (gitid: 9f8c877)

January 29, 2018

www.votca.org

Disclaimer

This manual is not complete. The best way to start using the software is to look at provided tutorials. The reference section is generated automatically from the source code, so please make sure that your software and manual versions match.

Citations

Development of this software depends on academic research grants. If you are using the package, please cite the following papers

[1] Microscopic simulations of charge transport in disordered organic semiconductors, Victor Rühle, Alexander Lukyanov, Falk May, Manuel Schrader, Thorsten Vehoff, James Kirkpatrick, Björn Baumeier and Denis Andrienko
J. Chem. Theor. Comp. **7**, 3335, 2011

[2] Versatile Object-oriented Toolkit for Coarse-graining Applications
Victor Rühle, Christoph Junghans, Alexander Lukyanov, Kurt Kremer and Denis Andrienko
J. Chem. Theor. Comp. **5**, 3211, 2009

Development

The core development is currently taking place at the Max Planck Institute for Polymer Research, Mainz, Germany and TU/e Eindhoven.

Copyright

VOTCA-XTP is free software. The entire package is available under the Apache License. For details, check the LICENSE file in the source code. The VOTCA-XTP source code is available on our homepage, www.votca.org.

Contents

1	Introduction	1
2	Theoretical background	3
2.1	Workflow	3
2.2	Material morphology	3
2.3	Conjugated segments and rigid fragments	4
2.4	Neighbor list	6
2.5	Reorganization energy	6
2.5.1	Intramolecular reorganization energy	6
2.5.2	Outersphere reorganization energy	7
2.6	Site energies	8
2.6.1	Externally applied electric field	8
2.6.2	Internal energy	9
2.6.3	Electrostatic interaction energy	9
2.6.4	Induction energy - the Thole model	10
2.7	Transfer integrals	12
2.7.1	Projection of monomer orbitals on dimer orbitals (DIPRO)	13
2.7.2	DFT-based transfer integrals using DIPRO	14
2.7.3	ZINDO-based transfer integrals using MOO	17
2.8	Charge transfer rate	17
2.8.1	Classical charge transfer rate	17
2.8.2	Semi-classical bimolecular rate	18
2.8.3	Semi-classical rate	18
2.9	Master equation	19
2.9.1	Extrapolation to nondispersive mobilities	19
2.10	Stochastic Networks	20
2.10.1	Coarse-grained morphology	20
2.10.2	Charge transport network	23
2.11	Macroscopic observables	26
2.11.1	Charge density	26
2.11.2	Current	27
2.11.3	Mobility and diffusion constant	27
2.11.4	Spatial correlations of energetic disorder	28
2.12	Random Facts	28
2.12.1	xqmm	28
2.12.2	EWALD	28
2.12.3	GW-BSE	29

3	Input and output files	33
3.1	Atomistic topology	33
3.2	Mapping file	35
3.3	Molecular orbitals	36
3.4	Monomer calculations for DFT transfer integrals	37
3.5	Pair calculations for DFT transfer integrals	39
3.6	DFT transfer integrals	41
3.7	State file	42
4	Reference	45
4.1	Programs	45
4.1.1	xtp_map	45
4.1.2	xtp_run	45
4.1.3	xtp_tools	45
4.1.4	xtp_parallel	46
4.1.5	xtp_dump	46
4.1.6	xtp_testsuite	46
4.1.7	xtp_update	47
4.1.8	xtp_update_exciton	47
4.1.9	xtp_basisset	47
4.1.10	xtp_makeauxbasis	47
4.2	Calculators	47
4.3	Common options	48
	Bibliography	49

Chapter 1

Introduction

sec:introduction

Charge carrier dynamics in an organic semiconductor can often be described in terms of charge hopping between localized states. The hopping rates depend on **electronic coupling elements**, **reorganization energies**, and **site energies**, which vary as a function of position and orientation of the molecules. The purpose of the VOTCA-XTP package [1] is to simplify the workflow for charge transport simulations, provide a uniform error-control for the methods, flexible platform for their development, and eventually allow *in silico* prescreening of organic semiconductors for specific applications.

The toolkit is implemented using modular concepts introduced earlier in the Versatile Object-oriented Toolkit for Coarse-graining Applications (VOTCA) [2]. It contains different **programs**, which execute specific tasks implemented in **calculators** representing an individual step in the workflow. Figure 1.1 summarizes a typical chain of commands to perform a charge transport simulation: First, the VOTCA code structures are adapted to reading atomistic trajectories, mapping them onto conjugated segments and rigid fragments, and substituting (if needed) rigid fragments with the optimized copies (**xtp_map**). The programs **xtp_run** and **xtp_parallel** (for heavy-duty tasks) are then used to calculate all bimolecular charge hopping rates (via precalculation of all required ingredients). **Site energies (or energetic disorder)** can be determined as a combination of internal (ionization potentials/electron affinities of single molecules) as well as electrostatic and polarization contributions within the molecular environment. The calculation of **electronic coupling elements** between conjugated segments from the corresponding molecular orbitals can be performed using a **dimer-projection** technique based on **density-functional** theory (DFT). This requires explicit calculations using quantum-chemistry software for which we provide interfaces to Gaussian, Turbomole, and NWChem. Alternatively, the **molecular orbital overlap** module calculates electronic coupling elements relying on the semi-empirical INDO Hamiltonian and molecular orbitals in the format provided by the Gaussian package.

The **kinetic Monte Carlo** module reads in the neighbor list, site coordinates, and hopping rates and performs charge dynamics simulations using either periodic boundary conditions or charge sources and sinks.

The toolkit is written as a combination of modular C++ code and scripts. The data transfer between programs is implemented via a state file (sql database), which is also used to restart simulations. Analysis functions and most of the calculation routines are encapsulated by using the observer pattern [3] which allows the implementation of new functions as individual modules.

In the following chapter 2, we summarize the **theoretical background** of the workflow of charge transport simulations and in particular its individual steps. Chapter 3 describes the structure and content of input and output files, while a full reference of **programs** and **calculators** is available in chapter 4. For a hands-on tutorial, the reader is referred to the VOTCA-XTP project page at <http://code.google.com/p/votca-xtp/>.

Input files:

conf.gro
GROMACS trajectory
topol.tpr
GROMACS topology
map.xml
mapping and energies
options.xml
options for calculators

Output files:

state.sql
sqlite3 database file for
data transfer between
modules



Get list of available calculators: xtp_run/xtp_parallel/xtp_kmc_run -l

Get help and list of options for a calculator: xtp_run/xtp_parallel/xtp_kmc_run -d neighborlist

Figure 1.1: A practical workflow of charge transport simulations using VOTCA-XTP. The **theoretical background** of the individual steps is given in chapter 2. Chapter 3 describes the content of input and output files, while a full reference of **programs** and **calculators** is available in chapter 4.

fig:summary

Chapter 2

Theoretical background

2.1 Workflow

A typical workflow of charge transport simulations is depicted in figure 2.1. The first step is the simulation of an atomistic morphology, which is then partitioned on hopping sites. The coordinates of the hopping sites are used to construct a list of pairs of molecules, or neighbor list.



Figure 2.1: Workflow for microscopic simulations of charge transport.

For each pair an electronic coupling element, a reorganization energy, a driving force, and eventually the hopping rate are evaluated. The neighbor list and hopping rates define a directed graph. The corresponding master equation is solved using the kinetic Monte Carlo method, which allows to explicitly monitor the charge dynamics in the system as well as to calculate time or ensemble averages of occupation probabilities, charge fluxes, correlation functions, and field-dependent mobilities.

2.2 Material morphology

There is no generic recipe on how to predict a large-scale atomistically-resolved morphology of an organic semiconductor. The required methods are system-specific: for ultra-pure crystals, for



Figure 2.2: The concept of conjugated segments and rigid fragments. Dashed lines indicate conjugated segments while colors denote rigid fragments. (a) Hexabenzocoronene: the π -conjugated system is both a rigid fragment and a conjugated segment. (b) Alq_3 : the Al atom and each ligand are rigid fragments while the whole molecule is a conjugated segment. (c) Polythiophene: each repeat unit is a rigid fragment. A conjugated segment consists of one or more rigid fragments. One molecule can have several conjugated segments.

fig:segment

example, density-functional methods can be used provided the crystal structure is known from experiment. For partially disordered organic semiconductors, however, system sizes much larger than a unit cell are required. Classical molecular dynamics or Monte Carlo techniques are then the methods of choice.

In molecular dynamics, atoms are represented by point masses which interact via empirical potentials prescribed by a force-field. Force-fields are parametrized for a limited set of compounds and their refinement is often required for new molecules. In particular, special attention shall be paid to torsion potentials between successive repeat units of conjugated polymers or between functional groups and the π -conjugated system. First-principles methods can be used to characterize the missing terms of the potential energy function.

Self-assembling materials, such as soluble oligomers, discotic liquid crystals, block copolymers, partially crystalline polymers, etc., are the most complicated to study. The morphology of such systems often has several characteristic length scales and can be kinetically arrested in a thermodynamically non-equilibrium state. For such systems, the time- and length-scales of atomistic simulations might be insufficient to equilibrate or sample desired morphologies. In this case, systematic coarse-graining can be used to enhance sampling [2]. Note that the coarse-grained representation must reflect the structure of the atomistic system and allow for back-mapping to the atomistic resolution.

Here we assume that the morphology is already known, that is we know how the topology and the coordinates of all atoms in the systems at a given time. VOTCA-XTP can read standard GROMACS topology files. Custom definitions of **atomistic topology** via XML files are also possible. Since the description of the atomistic topology is the first step in the charge transport simulations, it is important to follow simple conventions on how the system is partitioned on molecules, residues, and how atoms are named in the topology. Required input files are described in section **atomistic topology**.

2.3 Conjugated segments and rigid fragments

sec:segments

With the morphology at hand, the next step is partitioning the system on hopping sites, or conjugated segments, and calculating charge transfer rates between them. Physically intuitive arguments can be used for the partitioning, which reflects the localization of the wave function of a charge. For most organic semiconductors, the molecular architecture includes relatively rigid, planar π -conjugated systems, which we will refer to as rigid fragments. A conjugated segment can contain one or more of such rigid fragments, which are linked by bonded degrees of freedom.

The dynamics of these degrees of freedom evolves on timescales much slower than the frequency of the internal promoting mode. In some cases, e.g. glasses, it can be ‘frozen’ due to non-bonded interactions with the surrounding molecules.

To illustrate the concept of conjugated segments and rigid fragments, three representative molecular architectures are shown in figure 2.2. The first one is a typical discotic liquid crystal, hexabenzocoronene. It consists of a conjugated core to which side chains are attached to aid self-assembly and solution processing. In this case the orbitals localized on side chains do not participate in charge transport and the conjugated π -system is both, a rigid fragment and a conjugated segment. In Alq_3 , a metal-coordinated compound, a charge carrier is delocalized over all three ligands. Hence, the whole molecule is one conjugated segment. Individual ligands are relatively rigid, while energies of the order of $k_B T$ are sufficient to reorient them with respect to each other. Thus the Al atom and the three ligands are rigid fragments. In the case of a conjugated polymer, one molecule can consist of several conjugated segments, while each backbone repeat unit is a rigid fragment. Since the conjugation along the backbone can be broken due to large out-of-plane twists between two repeat units, an empirical criterion, based on the dihedral angle, can be used to partition the backbone on conjugated segments [4]. However, such intuitive partitioning is, to some extent, arbitrary and shall be validated by other methods [5–7].

After partitioning, an additional step is often required to remove bond length fluctuations introduced by molecular dynamics simulations, since they are already integrated out in the derivation of the rate expression. This is achieved by substituting respective molecular fragments with rigid, planar π -systems optimized using first-principles methods. Centers of mass and gyration tensors are used to align rigid fragments, though a custom definition of local axes is also possible. Such a procedure also minimizes discrepancies between the force-field and first-principles-based ground state geometries of conjugated segments, which might be important for calculations of electronic couplings, reorganization energies, and intramolecular driving forces.

To partition the system on hopping sites and substitute rigid fragments with the corresponding ground-state geometries `xtp_map` program is used:

 **Mapping the GROMACS trajectory**
`| xtp_map -t topol.tpr -c traj.xtc -s map.xml -f state.sql`

It reads in the GROMACS topology (`topol.tpr`) and trajectory (`traj.xtc`) files, definitions of conjugated segments and rigid fragments (`map.xml`) and outputs coordinates of conjugated segments (hopping sites) and rigid fragments (as provided in the MD trajectory and after rigidification) to the state file (`state.sql`). In order to do this, a mapping file `map.xml` has to be provided, which specifies the corresponding atoms in the different representations. After this step, all information (frame number, dimensions of the simulation box, etc) are stored in the state file and only this file is used for further calculations.



Be careful!

VOTCA-XTP requires a wrapped trajectory for mapping the segments and fragments, so all molecules should be whole in the frame.

In order to visually check the mapping one can use either the `tdump` calculator or the program `xtp_dump` with the calculator `trajectory2pdb`.

 **Writing a mapped trajectory with `xtp_dump`**
`| xtp_dump -f state.sql -e trajectory2pdb`

It reads in the state file created by `xtp_map` and outputs two trajectory files corresponding to the original and rigidified atom coordinates. To check the mapping, it is useful to superimpose the three outputs (original atomistic, atomistic stored in the state file, and rigidified according to ground state geometries), e.g., with VMD.

 **Writing a mapped trajectory with `tdump`**
`| xtp_run -f state.sql -o options.xml -e tdump`

It also reads in the state file but appends the coordinates to a `pdb` file. So make sure to delete old `QM.pdb` and `MD.pdb` if you want to create a new imagef

2.4 Neighbor list

A list of neighboring conjugated segments, or neighbor list, contains all pairs of conjugated segments for which **coupling elements**, **reorganization energies**, **site energy differences**, and **rates** are evaluated.

Two segments are added to this list if the distance between centers of mass of any of their rigid fragments is below a certain cutoff. This allows neighbors to be selected on a criterion of minimum distance of approach rather than center of mass distance, which is useful for molecules with anisotropic shapes.

The neighbor list can be generated from the atomistic trajectory by using the `neighborlist calculator`. This calculator requires a cutoff, which can be specified in the `options.xml` file. The list is saved to the `state.sql` file:

 **Generating a neighbor list**
`| xtp_run -o options.xml -f state.sql -e neighborlist`

2.5 Reorganization energy

The reorganization energy λ_{ij} takes into account the change in nuclear (and dielectric) degrees of freedom as the charge moves from donor i to acceptor j . It has two contributions: intramolecular, $\lambda_{ij}^{\text{int}}$, which is due to reorganization of nuclear coordinates of the two molecules forming the charge transfer complex, and intermolecular (outersphere), $\lambda_{ij}^{\text{out}}$, which is due to the relaxation of the nuclear coordinates of the environment. In what follows we discuss how these contributions can be calculated.

2.5.1 Intramolecular reorganization energy

If intramolecular vibrational modes of the two molecules are treated classically, the rearrangement of their nuclear coordinates after charge transfer results in the dissipation of the internal reorganization energy, $\lambda_{ij}^{\text{int}}$. It can be computed from four points on the potential energy surfaces (PES) of both molecules in neutral and charged states, as indicated in figure 2.3.

Adding the contributions due to discharging of molecule i and charging of molecule j yields [8]

$$\lambda_{ij}^{\text{int}} = \lambda_i^{cn} + \lambda_j^{nc} = U_i^{nC} - U_i^{nN} + U_j^{cN} - U_j^{cC}. \quad (2.1)$$

Here U_i^{nC} is the internal energy of the neutral molecule i in the geometry of its charged state (small n denotes the state and capital C the geometry). Similarly, U_j^{cN} is the energy of the charged molecule j in the geometry of its neutral state. Note that the PES of the donor and acceptor are not identical for chemically different compounds or for conformers of the same molecule. In this



Figure 2.3: Potential energy surfaces of (a) donor and (b) acceptor in charged and neutral states. After the change of the charge state both molecules relax their nuclear coordinates. If all vibrational modes are treated classically, the total internal reorganization energy and the internal energy difference of the electron transfer reaction are $\lambda_{ij}^{\text{int}} = \lambda_i^{cn} + \lambda_j^{nc}$ and $\Delta E_{ij}^{\text{int}} = \Delta U_i - \Delta U_j$, respectively.

fig:parabolas

case $\lambda_i^{cn} \neq \lambda_j^{cn}$ and $\lambda_i^{nc} \neq \lambda_j^{nc}$. Thus $\lambda_{ij}^{\text{int}}$ is a property of the charge transfer complex, and not of a single molecule.

Intramolecular reorganization energies for discharging (λ^{cn}) and charging (λ^{nc}) of a molecule need to be determined using quantum-chemistry and given in `map.xml`. The values are written to the `state.sql` using the calculator `einternal` (see also internal energy):

Intramolecular reorganization energies

```
| xtp_run -o options.xml -f state.sql -e einternal
```

2.5.2 Outersphere reorganization energy

161

sec:outersphere

During the charge transfer reaction, also the molecules outside the charge transfer complex reorient and polarize in order to adjust for changes in electric potential, resulting in the outersphere contribution to the reorganization energy. $\lambda_{ij}^{\text{out}}$ is particularly important if charge transfer occurs in a polarizable environment. Assuming that charge transfer is much slower than electronic polarization but much faster than nuclear rearrangement of the environment, $\lambda_{ij}^{\text{out}}$ can be calculated from the electric displacement fields created by the charge transfer complex [9]

$$\lambda_{ij}^{\text{out}} = \frac{c_p}{2\epsilon_0} \int_{V^{\text{out}}} dV \left[\vec{D}_I(\vec{r}) - \vec{D}_F(\vec{r}) \right]^2, \quad (2.2)$$

equ:lambda_outer1

where ϵ_0 is the permittivity of free space, $\vec{D}_{I,F}(\vec{r})$ are the electric displacement fields created by the charge transfer complex in the initial (charge on molecule i) and final (charge transferred to molecule j) states, V^{out} is the volume outside the complex, and $c_p = \frac{1}{\epsilon_{\text{opt}}} - \frac{1}{\epsilon_s}$ is the Pekar factor, which is determined by the low (ϵ_s) and high (ϵ_{opt}) frequency dielectric permittivities.

Eq. (2.2) can be simplified by assuming spherically symmetric charge distributions on molecules i and j with total charge e . Integration over the volume V^{out} outside of the two spheres of radii R_i and R_j centered on molecules i and j leads to the classical Marcus expression for the outersphere reorganization energy

$$\lambda_{ij}^{\text{out}} = \frac{c_p e^2}{4\pi\epsilon_0} \left(\frac{1}{2R_i} + \frac{1}{2R_j} - \frac{1}{r_{ij}} \right), \quad (2.3)$$

equ:lambda_outer2

where r_{ij} is the molecular separation. While eq. (2.3) captures the main physics, e.g. predicts smaller outer-sphere reorganization energies (higher rates) for molecules at smaller separations,

it often cannot provide quantitative estimates, since charge distributions are rarely spherically symmetric.

Alternatively, the displacement fields can be constructed using the atomic partial charges. The difference of the displacement fields at the position of an atom b_k outside the charge transfer complex (molecule $k \neq i, j$) can be expressed as

$$\vec{D}_I(\vec{r}_{b_k}) - \vec{D}_F(\vec{r}_{b_k}) = \sum_{a_i} \frac{q_{a_i}^c - q_{a_i}^n}{4\pi} \frac{(\vec{r}_{b_k} - \vec{r}_{a_i})}{|\vec{r}_{b_k} - \vec{r}_{a_i}|^3} + \sum_{a_j} \frac{q_{a_j}^n - q_{a_j}^c}{4\pi} \frac{(\vec{r}_{b_k} - \vec{r}_{a_j})}{|\vec{r}_{b_k} - \vec{r}_{a_j}|^3}, \quad (2.4)$$

where $q_{a_i}^n$ ($q_{a_i}^c$) is the partial charge of atom a of the neutral (charged) molecule i in vacuum. The partial charges of neutral and charged molecules are obtained by fitting their values to reproduce the electrostatic potential of a single molecule (charged or neutral) in vacuum. Assuming a uniform density of atoms, the integration in eq. (2.2) can be rewritten as a density-weighted sum over all atoms excluding those of the charge transfer complex.

The remaining unknown needed to calculate $\lambda_{ij}^{\text{out}}$ is the Pekar factor, c_p . In polar solvents $\epsilon_s \gg \epsilon_{\text{opt}} \sim 1$ and c_p is of the order of 1. In most organic semiconductors, however, molecular orientations are fixed and therefore the low frequency dielectric permittivity is of the same order of magnitude as ϵ_{opt} . Hence, c_p is small and its value is very sensitive to differences in the permittivities.

Outersphere reorganization energies for all pairs of molecules in the `neighbor list` can be computed from the atomistic trajectory by using the `eoutersphere calculator`.

Two methods can be used to compute $\lambda_{ij}^{\text{out}}$. The first method uses the atomistic partial charges of neutral and charged molecules from files specified in `map.xml` and eq. (2.2). The Pekar factor c_p and a cutoff radius based on molecular centers of mass have to be specified in the `options.xml` file.

If this method is computationally prohibitive, $\lambda_{ij}^{\text{out}}$ can be computed using eq. (2.3), which assumes spherical charge distributions on the molecules. In this case the radii of these spheres are specified in `segments.xml`, while the Pekar factor c_p is given in the `options.xml` file and no cutoff radius is needed.

The outer sphere reorganization energies are saved to the `state.sql` file:

```

 Outersphere reorganization energy
| xtp_run -o options.xml -f state.sql -e outersphere

```

2.6 Site energies

A charge transfer reaction between molecules i and j is driven by the site energy difference, $\Delta E_{ij} = E_i - E_j$. Since the transfer rate, ω_{ij} , depends exponentially on ΔE_{ij} (see eq. (2.31)) it is important to compute its distribution as accurately as possible. The total site energy difference has contributions due to **externally applied electric field**, electrostatic interactions, polarization effects, and **internal energy** differences. In what follows we discuss how to estimate these contributions by making use of first-principles calculations and polarizable force-fields.

2.6.1 Externally applied electric field

The contribution to the total site energy difference due to an external electric field \vec{F} is given by $\Delta E_{ij}^{\text{ext}} = q\vec{F} \cdot \vec{r}_{ij}$, where $q = \pm e$ is the charge and $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ is a vector connecting molecules i and j . For typical distances between small molecules, which are of the order of 1 nm, and moderate fields of $F < 10^8$ V/m this term is always smaller than 0.1 eV.

2.6.2 Internal energy

The contribution to the site energy difference due to different internal energies (see figure 2.3) can be written as

$$\Delta E_{ij}^{\text{int}} = \Delta U_i - \Delta U_j = (U_i^{cC} - U_i^{nN}) - (U_j^{cC} - U_j^{nN}), \quad (2.5)$$

where $U_i^{cC(nN)}$ is the total energy of molecule i in the charged (neutral) state and geometry. ΔU_i corresponds to the adiabatic ionization potential (or electron affinity) of molecule i , as shown in figure 2.3. For one-component systems and negligible conformational changes $\Delta E_{ij}^{\text{int}} = 0$, while it is significant for donor-acceptor systems.

Internal energies determined using quantum-chemistry need to be specified in `map.xml`. The values are written to the `state.sql` using the calculator `einternal` (see also `intramolecular reorganization energy`):

```
Internal energies
| xtp_run -o options.xml -f state.sql -e einternal
```

2.6.3 Electrostatic interaction energy

We represent the molecular charge density by choosing multiple expansion sites (“polar sites”) per molecule in such a way as to accurately reproduce the molecular electrostatic potential (ESP), with a set of suitably chosen multipole moments $\{Q_{lk}^a\}$ (in spherical-tensor notation) allocated to each site. The expression for the electrostatic interaction energy between two molecules A and B in the multi-point expansion includes an implicit sum over expansion sites $a \in A$ and $b \in B$,

$$U_{AB} = \sum_{a \in A} \sum_{b \in B} \hat{Q}_{l_1 k_1}^a T_{l_1 k_1 l_2 k_2}^{a,b} \hat{Q}_{l_2 k_2}^b \equiv \hat{Q}_{l_1 k_1}^a T_{l_1 k_1 l_2 k_2}^{a,b} \hat{Q}_{l_2 k_2}^b, \quad (2.6)$$

where we have used the Einstein sum convention for the site indices a and b on the right-hand side of the equation, in addition to the sum convention that is in place for the multipole-moment components $t \equiv l_1 k_1$ and $u \equiv l_2 k_2$. The $T_{l_1 k_1 l_2 k_2}^{a,b}$ are tensors that mediate the interaction between a multipole component $l_1 k_1$ on site a with the moment $l_2 k_2$ on site b . If we include the molecular environment into a perturbative term W to enter in the single-molecule Hamiltonian, the above expression is exactly the first-order correction to the energy where the quantum-mechanical detail has been absorbed in classical multipole moments.

There are a number of strategies how to arrive at such a collection of *distributed multipoles*. They can be classified according to whether the multipoles are derived (a) from the electrostatic potential generated by the SCF charge density or (b) from a decomposition of the wavefunction itself. Here, we will only draft two of those approaches, CHELPG [10] from category (a) and DMA [11] from category (b).

The CHELPG (CHarges from ELEctrostatic Potentials, Grid-based) method relies on performing a least-squares fit of atom-placed charges to reproduce the electrostatic potential as evaluated from the SCF density on a regularly spaced grid [10]. The fitted charges result from minimizing the Lagrangian function [12]

$$z(\{q_i\}) = \sum_{k=1}^M \left(\phi(\vec{r}_k) - \sum_{i=1}^N \frac{1}{4\pi\epsilon_0} \frac{q_i}{|\vec{r}_i - \vec{r}_k|} \right) + \lambda \left(q_{\text{mol}} - \sum_{i=1}^N q_i \right), \quad (2.7)$$

with M grid points, N atomic sites, the set of atomic partial charges $\{q_i\}$ and the SCF potential ϕ . The Lagrange multiplier λ constrains the sum of the fitted charges to the molecular charge q_{mol} . The main difference from other fitting schemes [13] is the algorithm that selects the positions

at which the potential is evaluated (we note that the choice of grid points can have substantial effects especially for bulky molecules). Clearly, the CHELPG method can be (and has been) extended to include higher atomic multipoles. It should be noted, however, how already the inclusion of atomic dipoles hardly improves the parametrization, and can in fact be harmful to its conformational stability.

The Distributed-Multipole-Analysis (DMA) approach [11, 14], developed by A. Stone, operates directly on the quantum-mechanical density matrix, expanded in terms of atom- and bond-centered Gaussian functions $\chi_\alpha = R_{LK}(\vec{x} - \vec{s}_\alpha) \exp[-\zeta(\vec{x} - \vec{s}_\alpha)^2]$,

$$\rho(\vec{x}) = \sum_{\alpha,\beta} \rho_{\alpha\beta} \chi_\alpha(\vec{x} - \vec{s}_\alpha) \chi_\beta(\vec{x} - \vec{s}_\beta). \quad (2.8)$$

The aim is to compute multipole moments according in a distributed fashion: If we use that the overlap product $\chi_\alpha \chi_\beta$ of two Gaussian basis functions yields itself a Gaussian centered at $\vec{P} = (\zeta_\alpha \vec{s}_\alpha + \zeta_\beta \vec{s}_\beta) / (\zeta_\alpha + \zeta_\beta)$, it is possible to proceed in two steps: First, we compute the multipole moments associated with a specific summand in the density matrix, referred to the overlap center \vec{P} :

$$Q_{LK}[\vec{P}] = - \int R_{LK}(\vec{x} - \vec{P}) \rho_{\alpha\beta} \chi_\alpha \chi_\beta d^3x. \quad (2.9)$$

Second, we transfer the resulting $Q_{lk}[\vec{P}]$ to the position \vec{S} of a polar site according to the rule [11]

$$Q_{nm}[\vec{S}] = \sum_{l=0}^L \sum_{k=-l}^l \left[\binom{n+m}{l+k} \binom{n-m}{l-k} \right]^{1/2} R_{n-l,m-k}(\vec{S} - \vec{P}) \cdot Q_{lk}[\vec{P}]. \quad (2.10)$$

Note how this requires a rule for the choice of the expansion site to which the multipole moment should be transferred. In the near past [14], the nearest-site algorithm, which allocates the multipole moments to the site closest to the overlap center, was replaced for diffuse functions by an algorithm based on a sxtptth weighting function in conjunction with grid-based integration methods in order to decrease the basis-set dependence of the resulting set of distributed multipoles. One important advantage of the DMA approach over fitting algorithms such as CHELPG or Merz-Kollman (MK) is that higher-order moments can also be derived without too large an ambiguity.

The ‘mps’ file format used by VOTCA for the definition of distributed multipoles (as well as point polarizabilities, see subsequent section) is based on the GDMA punch format of A. Stone’s GDMA program [14] (the punch output file can be immediately plugged into VOTCA without any conversions to be applied). Furthermore the log-file of different QM packages (currently Gaussian, Turbomole and NWChem) may be fed into the `log2mps` tool, which will subsequently generate the appropriate mps-file.



Read in ESP charges from a QM log file

```
| xtp_tools -o options.xml -e log2mps
```

2.6.4 Induction energy - the Thole model

If we in addition to the permanent set of multipole moments $\{Q_t^a\}$ allow for induced moments $\{\Delta Q_t^a\}$ and penalize their generation with a bilinear form (giving rise to a strictly positive contribution to the energy),

$$U_{\text{int}} = \frac{1}{2} \sum_A \Delta Q_t^a \eta_{tt'}^{aa'} \Delta Q_{t'}^{a'}, \quad (2.11)$$

it can be shown that the induction contribution to the site energy evaluates to an expression where all interactions between induced moments have cancelled out, and interactions between permanent and induced moments are scaled down by 1/2 [15]:

$$U_{pu} = \frac{1}{2} \sum_A \sum_{B>A} [\Delta Q_t^a T_{tu}^{ab} Q_u^b + \Delta Q_t^b T_{tu}^{ab} Q_u^a]. \quad (2.12) \quad \text{equ:u_pu}$$

This term can be viewed as the second-order (induction) correction to the molecular interaction energy. The sets of $\{Q_t^a\}$ are solved for self-consistently via

$$\Delta Q_t^a = - \sum_{B \neq A} \alpha_{tt'}^{aa'} T_{t'u}^{a'b} (Q_u^b + \Delta Q_u^b), \quad (2.13) \quad \text{equ:self_consistent_dQ}$$

where the polarizability tensors $\alpha_{tt'}^{aa'}$ are given by the inverse of $\eta_{tt'}^{aa'}$. With eqs. 2.13 and 2.12 we have at hand expressions that allow us to compute the induction energy contribution to site energies in an iterative manner based on a set of molecular distributed multipoles $\{Q_t^a\}$ and polarizabilities $\{\alpha_{tt'}^{aa'}\}$. We have drafted in the previous section how to obtain the former from a wavefunction decomposition or fitting scheme (GDMA, CHELPG). The $\{\alpha_{tt'}^{aa'}\}$ can be derived formally (or rather: read off) from a perturbative expansion of the molecular interaction. In this work we make use of the Thole model [16, 17] as a semi-empirical approach to obtain the sought-after point polarizabilities in the local dipole approximation, that is, $[\alpha_{tt'}^{aa'}] = \alpha_{tt'}^{aa'} \delta_{t\beta} \delta_{t'\beta} \delta_{aa'}$, where $\beta \in \{x, y, z\}$ references the dipole-moment component. The Thole model is based on a modified dipole-dipole interaction, which can be reformulated in terms of the interaction of smeared charge densities. This has been shown to be necessary due to the divergent head-to-tail dipole-dipole interaction that otherwise results at small interseparations on the Å scale [16–18]. Smearing out the charge distribution mimics the nature of the QM wavefunction, which effectively guards against this unphysical polarization catastrophe. Since the point dipoles however only react individually to the external field, any correlation effects as were still accounted for in the $\{\alpha_{tt'}^{aa'}\}$ are lost, except perhaps those correlations that are due to the mere classical field interaction.

The smearing of the nuclei-centered multipole moments is obtained via a fractional charge density $\rho_f(\vec{u})$ which should be normalized to unity and fall off rapidly as of a certain radius $\vec{u} = \vec{u}(\vec{R})$. The latter is related to the physical distance vector \vec{R} connecting two interacting sites via a linear scaling factor that takes into account the magnitude of the isotropic site polarizabilities α^a . This isotropic fractional charge density gives rise to a modified potential

$$\phi(u) = -\frac{1}{4\pi\epsilon_0} \int_0^u 4\pi u' \rho(u') du' \quad (2.14) \quad \text{equ:mod_potential}$$

We can relate the multipole interaction tensor $T_{ij\dots}$ (this time in Cartesian coordinates) to the fractional charge density in two steps: First, we rewrite the tensor in terms of the scaled distance vector \vec{u} ,

$$T_{ij\dots}(\vec{R}) = f(\alpha^a \alpha^b) t_{ij\dots}(\vec{u}(\vec{R}), \alpha^a \alpha^b), \quad (2.15)$$

where the specific form of $f(\alpha^a \alpha^b)$ results from the choice of $u(\vec{R}, \alpha^a \alpha^b)$. Second, we demand that the smeared interaction tensor $t_{ij\dots}$ is given as usual by the appropriate derivative of the potential in eq. 2.14,

$$t_{ij\dots}(\vec{u}) = -\partial_{u_i} \partial_{u_j} \dots \phi(\vec{u}). \quad (2.16)$$

It turns out that for a suitable choice of $\rho_f(\vec{u})$, the modified interaction tensors can be rewritten in such a way that powers n of the distance $R = |\vec{R}|$ are damped with a damping function $\lambda_n(\vec{u}(\vec{R}))$ [19].

There is a large number of fractional charge densities $\rho_f(\vec{u})$ that have been tested for the purpose of giving best results for the molecular polarizability as well as interaction energies. Note how a great advantage of the Thole model is the exceptional transferability of the atomic polarizabilities to compounds not used for the fitting procedure [17]. In fact, for most organic molecules, a fixed set of atomic polarizabilities ($\alpha_C = 1.334$, $\alpha_H = 0.496$, $\alpha_N = 1.073$, $\alpha_O = 0.873$, $\alpha_S = 2.926 \text{ \AA}^3$) based on atomic elements yields satisfactory results.

VOTCA implements the Thole model with an exponentially-decaying fractional charge density

$$\rho(u) = \frac{3a}{4\pi} \exp(-au^3), \quad (2.17)$$

where $\vec{u}(\vec{R}, \alpha^a \alpha^b) = \vec{R}/(\alpha^a \alpha^b)^{1/6}$ and the smearing exponent $a = 0.39$ (which can however be changed from the program options), as used in the AMOEBA force field [19].

Even though the Thole model performs very well for many organic compounds with only the above small set of element-based polarizabilities, conjugated molecules may require a more intricate parametrization. The simplest approach is to resort to scaled polarizabilities to match the effective molecular polarizable volume $V \sim \alpha_x \alpha_y \alpha_z$ as predicted by QM calculations (here $\alpha_x, \alpha_y, \alpha_z$ are the eigenvalues of the molecular polarizability tensor). The `molpol` tool assists with this task, it self-consistently calculates the Thole polarizability for an input `mps`-file and optimizes (if desired) the atomic polarizabilities in the above simple manner.

Generate Thole-type polarizabilities for a segment

```
| xtp_tools -o options.xml -e molpol
```

The electrostatic and induction contribution to the site energy is evaluated by the `emultipole` calculator. Atomistic partial charges for charged and neutral molecules are taken from `mps`-files (extended GDMA format) specified in `map.xml`. Note that, in order to speed up calculations for both methods, a cut-off radius (for the molecular centers of mass) can be given in `options.xml`. Threaded execution is advised.

Electrostatic and induction corrections

```
| xtp_run -o options.xml -f state.sql -e emultipole
```

Furthermore available are `zmultipole`, which extends `emultipole` to allow for an electrostatic buffer layer (loosely related to the z-buffer in OpenGL, hence the name) and anisotropic point polarizabilities. For the interaction energy of charged clusters of any user-defined composition (Frenkel states, CT states, ...), `xqmultipole` can be used.

Interaction energy of charged molecular clusters embedded in a molecular environment

```
| xtp_parallel -o options.xml -f state.sql -e xqmultipole
```

2.7 Transfer integrals

The electronic transfer integral element J_{ij} entering the Marcus rates in eq. (2.31) is defined as

$$J_{ij} = \langle \phi_i | \hat{H} | \phi_j \rangle, \quad (2.18) \quad \text{equ:TI}$$

where ϕ_i and ϕ_j are diabatic wavefunctions, localized on molecule i and j respectively, participating in the charge transfer, and \hat{H} is the Hamiltonian of the formed dimer. Within the frozen-core approximation, the usual choice for the diabatic wavefunctions ϕ_i is the highest occupied molecular orbital (HOMO) in case of hole transport, and the lowest unoccupied molecular orbital (LUMO) in the case of electron transfer, while \hat{H} is an effective single particle Hamiltonian,

e.g. Fock or Kohn-Sham operator of the dimer. As such, J_{ij} is a measure of the strength of the electronic coupling of the frontier orbitals of monomers mediated by the dimer interactions. Intrinsically, the transfer integral is very sensitive to the molecular arrangement, i.e. the distance and the mutual orientation of the molecules participating in charge transport. Since this arrangement can also be significantly influenced by static and/or dynamic disorder [20–24], it is essential to calculate J_{ij} explicitly for each hopping pair within a realistic morphology. Considering that the number of dimers for which eq. (2.18) has to be evaluated is proportional to the number of molecules times their coordination number, computationally efficient and at the same time quantitatively reliable schemes are required.

2.7.1 Projection of monomer orbitals on dimer orbitals (DIPRO)

sec:dipro

An approach for the determination of the transfer integral that can be used for any single-particle electronic structure method (Hartree-Fock, DFT, or semiempirical methods) is based on the projection of monomer orbitals on a manifold of explicitly calculated dimer orbitals. This dimer projection (DIPRO) technique including an assessment of computational parameters such as the basis set, exchange-correlation functionals, and convergence criteria is presented in detail in ref. [25]. A brief summary of the concept is given below.

We start from an effective Hamiltonian¹

$$\hat{H}^{\text{eff}} = \sum_i \epsilon_i \hat{a}_i^\dagger \hat{a}_i + \sum_{j \neq i} J_{ij} \hat{a}_i^\dagger \hat{a}_j + c.c. \quad (2.19) \quad \text{equ:dipro_eq1}$$

where \hat{a}_i^\dagger and \hat{a}_i are the creation and annihilation operators for a charge carrier located at the molecular site i . The electron site energy is given by ϵ_i , while J_{ij} is the transfer integral between two sites i and j . We label their frontier orbitals (HOMO for hole transfer, LUMO for electron transfer) ϕ_i and ϕ_j , respectively. Assuming that the frontier orbitals of a dimer (adiabatic energy surfaces) result exclusively from the interaction of the frontier orbitals of monomers, and consequently expand them in terms of ϕ_i and ϕ_j . The expansion coefficients, $\bar{\mathbf{C}}$, can be determined by solving the secular equation

$$(\mathbf{H} - E\mathbf{S})\bar{\mathbf{C}} = 0 \quad (2.20) \quad \text{equ:dipro_eq2}$$

where \mathbf{H} and \mathbf{S} are the Hamiltonian and overlap matrices of the system, respectively. These matrices can be written explicitly as

$$\mathbf{H} = \begin{pmatrix} e_i & H_{ij} \\ H_{ij}^* & e_j \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 1 & S_{ij} \\ S_{ij}^* & 1 \end{pmatrix} \quad (2.21) \quad \text{equ:dipro_eq3}$$

with

$$\begin{aligned} e_i &= \langle \phi_i | \hat{H} | \phi_i \rangle & H_{ij} &= \langle \phi_i | \hat{H} | \phi_j \rangle \\ e_j &= \langle \phi_j | \hat{H} | \phi_j \rangle & S_{ij} &= \langle \phi_j | \phi_j \rangle \end{aligned} \quad (2.22) \quad \text{equ:dipro_eq4}$$

The matrix elements $e_{i(j)}$, H_{ij} , and S_{ij} entering eq. (2.21) can be calculated via projections on the dimer orbitals (eigenfunctions of \hat{H}) $\{|\phi_n^D\rangle\}$ by inserting $\hat{1} = \sum_n |\phi_n^D\rangle \langle \phi_n^D|$ twice. We exemplify this explicitly for H_{ij} in the following

$$H_{ij} = \sum_{nm} \langle \phi_i | \phi_n^D \rangle \langle \phi_n^D | \hat{H} | \phi_m^D \rangle \langle \phi_m^D | \phi_j \rangle. \quad (2.23) \quad \text{equ:dipro_eq16}$$

The Hamiltonian is diagonal in its eigenfunctions, $\langle \phi_n^D | \hat{H} | \phi_m^D \rangle = E_n \delta_{nm}$. Collecting the projections of the frontier orbitals $|\phi_{i(j)}\rangle$ on the n -th dimer state $(\bar{\mathbf{V}}_{(i)})_n = \langle \phi_i | \phi_n^D \rangle$ and $(\bar{\mathbf{V}}_{(j)})_n = \langle \phi_j | \phi_n^D \rangle$ respectively, into vectors we obtain

$$H_{ij} = \bar{\mathbf{V}}_{(i)} \mathbf{E} \bar{\mathbf{V}}_{(j)}^\dagger. \quad (2.24) \quad \text{eq:dipro_eq17}$$

¹we use following notations: a - number, $\bar{\mathbf{a}}$ - vector, \mathbf{A} - matrix, \hat{A} - operator

What is left to do is determine these projections $\bar{\mathbf{V}}_{(k)}$. In all practical calculations the molecular orbitals are expanded in basis sets of either plane waves or of localized atomic orbitals $|\varphi_\alpha\rangle$. We will first consider the case that the calculations for the monomers are performed using a counterpoise basis set that is commonly used to deal with the basis set superposition error (BSSE). The basis set of atom-centered orbitals of a monomer is extended to the one of the dimer by adding the respective atomic orbitals at virtual coordinates of the second monomer. We can then write the respective expansions as

$$|\phi_k\rangle = \sum_{\alpha} \lambda_{\alpha}^{(k)} |\varphi_{\alpha}\rangle \quad \text{and} \quad |\phi_n^D\rangle = \sum_{\alpha} D_{\alpha}^{(n)} |\varphi_{\alpha}\rangle \quad (2.25) \quad \text{eq:dipro_eq18}$$

where $k = i, j$. The projections can then be determined within this common basis set as

$$(\bar{\mathbf{V}}_k)_n = \langle \phi_k | \phi_n^D \rangle = \sum_{\alpha} \lambda_{\alpha}^{(k)} \langle \alpha | \sum_{\beta} D_{\beta}^{(n)} |\beta\rangle = \bar{\lambda}_{(k)}^{\dagger} \mathcal{S} \bar{\mathbf{D}}_{(n)} \quad (2.26) \quad \text{eq:dipro_eq19}$$

where \mathcal{S} is the overlap matrix of the atomic basis functions. This allows us to finally write the elements of the Hamiltonian and overlap matrices in eq. (2.21) as:

$$\begin{aligned} H_{ij} &= \bar{\lambda}_{(i)}^{\dagger} \mathcal{S} \mathbf{D} \mathbf{E} \mathbf{D}^{\dagger} \mathcal{S}^{\dagger} \bar{\lambda}_{(j)} \\ S_{ij} &= \bar{\lambda}_{(i)}^{\dagger} \mathcal{S} \mathbf{D} \mathbf{D}^{\dagger} \mathcal{S}^{\dagger} \bar{\lambda}_{(j)} \end{aligned} \quad (2.27) \quad \text{eq:dipro_eq20}$$

Since the two monomer frontier orbitals that form the basis of this expansion are not orthogonal in general ($\mathbf{S} \neq \mathbf{1}$), it is necessary to transform eq. (2.20) into a standard eigenvalue problem of the form

$$\mathbf{H}^{\text{eff}} \bar{\mathbf{C}}^{\text{eff}} = E \bar{\mathbf{C}}^{\text{eff}} \quad (2.28) \quad \text{eq:dipro_eq7}$$

to make it correspond to eq. (2.19). According to Löwdin such a transformation can be achieved by

$$\mathbf{H}^{\text{eff}} = \mathbf{S}^{-1/2} \mathbf{H} \mathbf{S}^{-1/2}. \quad (2.29) \quad \text{eq:dipro_eq9}$$

This then yields an effective Hamiltonian matrix in an orthogonal basis, and its entries can directly be identified with the site energies ϵ_i and transfer integrals J_{ij} :

$$\mathbf{H}^{\text{eff}} = \begin{pmatrix} e_i^{\text{eff}} & H_{ij}^{\text{eff}} \\ H_{ij}^{*,\text{eff}} & e_j^{\text{eff}} \end{pmatrix} = \begin{pmatrix} \epsilon_i & J_{ij} \\ J_{ij}^* & \epsilon_j \end{pmatrix} \quad (2.30) \quad \text{eq:dipro_eq11}$$

2.7.2 DFT-based transfer integrals using DIPRO

sec:dft

The calculation of one electronic coupling element based on DFT using the DIPRO method requires the overlap matrix of atomic orbitals \mathcal{S} , the expansion coefficients for monomer $\bar{\lambda}_{(k)} = \{\lambda_{\alpha}^{(k)}\}$ and dimer orbitals $\bar{\mathbf{D}}_{(n)} = \{D_{\alpha}^{(n)}\}$, as well as the orbital energies E_n of the dimer are required as input. In practical situations, performing self-consistent quantum-chemical calculations for each individual monomer and one for the dimer to obtain this input data is extremely demanding. Several simplifications can be made to reduce the computational effort, such as using non-Counterpoise basis sets for the monomers (thereby decoupling the monomer calculations from the dimer run) and performing only a single SCF step in a dimer calculation starting from an initial guess formed from a superposition of monomer orbitals. This "noCP+noSCF" variant of DIPRO is shown in figure 2.4(a) and recommended for production runs. A detailed comparative study of the different variants can be found in [25].

The code currently contains supports evaluation of transfer integrals from quantum-chemical calculations performed with the Gaussian, Turbomole, and NWChem packages. The interfacing procedure consists of three main steps: generation of input files for monomers and dimers, performing the actual quantum-chemical calculations, and calculating the transfer integrals.



Figure 2.4: Schematics of the DIPRO method. (a) General workflow of the projection technique. (b) Strategy of the efficient noCP+noSCF implementation, in which the monomer calculations are performed independently from the dimer configurations (noCP), using the `edft calculator`. The dimer Hamiltonian is subsequently constructed based on an initial guess formed from monomer orbitals and only diagonalized once (noSCF) before the transfer integral is calculated by projection. This second step is performed by the `idft calculator`.

fig:dipro_scheme

Monomer calculations

First, `hopping sites` and a `neighbor list` need to be generated from the atomistic topology and trajectory and written to the `state.sql` file. Then the parallel `edft calculator` manages the calculation of the monomer properties required for the determination of electronic coupling elements. Specifically, the individual steps it performs are:

1. Creation of a job file containing the list of molecules to be calculated with DFT

```
Writing job file for edft
| xtp_parallel -o options.xml -f state.sql -e edft -jwrite
```

2. Running of all jobs in job file

```
Running all edft jobs
| xtp_parallel -o options.xml -f state.sql -e edft -jrun
```

which includes

- creating the input files for the DFT calculation (using the package specified in `options.xml`) in the directory

389 OR_FILES/package/frame_F/mol_M

390 where F is the index of the frame in the trajectory, M is the index of a molecule in this
391 frame,

- 392 • executing the DFT run, and
- 393 • after completion of this run, parsing the output (number of electrons, basis set, molec-
394 ular orbital expansion coefficients), and saving it in compressed form to

395 OR_FILES/molecules/frame_F/molecule_M.orb

396 Calculating the transfer integrals

sec:tdft

397 After the monomer calculations have been completed successfully, the respective runs for dimers
398 from the neighborlist can be performed using the parallel idft **calculator**, which manages the
399 DFT runs for the hopping pairs and determines the coupling element using DIPRO. Again, sev-
400 eral steps are required:

- 401 1. Creation of a job file containing the list of pairs to be calculated with DFT

 **Writing job file for idft**
| `xtp_parallel -o options.xml -f state.sql -e idft -j write`

- 402 2. Running of all jobs in job file

 **Running all idft jobs**
| `xtp_parallel -o options.xml -f state.sql -e idft -j run`

403 which includes

- 404 • creating the input files (including the merged guess for a noSCF calculation, if re-
405 quested) for the DFT calculation (using the package specified in `options.xml`) in the
406 directory

407 OR_FILES/package/frame_F/pair_M_N

408 where M and N are the indices of the molecules in this pair,

- 409 • executing the DFT run, and
- 410 • after completion of this run, parsing the output (number of electrons, basis set, molec-
411 ular orbital expansion coefficients and energies, atomic orbital overlap matrix), and
412 saving the pair information in compressed form to

413 OR_FILES/pairs/frame_F/pair_M_N.orb

- 414 • loading the monomer orbitals from the previously saved *.orb files.
- 415 • calculating the coupling elements and write them to the job file

- 416 3. Reading the coupling elements from the job file and saving them to the `state.sql` file

417

 **Saving idft results from job file to state.sql**
| `xtp_parallel -o options.xml -f state.sql -e idft -j read`

2.7.3 ZINDO-based transfer integrals using MOO

An approximate method based on Zerner's Intermediate Neglect of Differential Overlap (ZINDO) has been described in Ref. [26]. This semiempirical method is substantially faster than first-principles approaches, since it avoids the self-consistent calculations on each individual monomer and dimer. This allows to construct the matrix elements of the ZINDO Hamiltonian of the dimer from the weighted overlap of molecular orbitals of the two monomers. Together with the introduction of rigid segments, only a single self-consistent calculation on one isolated conjugated segment is required. All relevant molecular overlaps can then be constructed from the obtained molecular orbitals.

The main advantage of the molecular orbital overlap (MOO) library is *fast* evaluation of electronic coupling elements. Note that MOO is based on the ZINDO Hamiltonian which has limited applicability. The general advice is to first compare the accuracy of the MOO method to the DFT-based calculations.

MOO can be used both in a standalone mode and as an `izindo` calculator of VOTCA-XTP.

Since MOO constructs the Fock operator of a dimer from the molecular orbitals of monomers by translating and rotating the orbitals of *rigid fragments*, the optimized geometry of all *conjugated segments* and the coefficients of the molecular orbitals are required as its input in addition to the state file (`state.sql`) with the *neighbor list*. Coordinates are stored in `geometry.xyz` files with four columns, first being the atom type and the next three atom coordinates. This is a standard xyz format without a header. Note that the atom order in the `geometry.xyz` files can be different from that of the mapping files. The correspondence between the two is established in the `map.xml` file.



Be careful!

Izindo requires the specification of orbitals for hole and electron transport in `map.xml`. They are the HOMO and LUMO respectively and can be retrieved from the `log` file from which the `zindo.orb` file is generated. The number of alpha electrons is the HOMO, the LUMO is HOMO+1

The calculated transfer integrals are immediately saved to the `state.sql` file.



Transfer integrals from `izindo`

```
| xtp_run -o options.xml -f state.sql -e izindo
```

2.8 Charge transfer rate

Charge transfer rates can be postulated based on intuitive physical considerations, as it is done in the Gaussian disorder models [20, 27–29]. Alternatively, charge transfer theories can be used to evaluate rates from quantum chemical calculations [1, 8, 25, 30–32]. In spite of being significantly more computationally demanding, the latter approach allows to link the chemical and electronic structure, as well as the morphology, to charge dynamics.

2.8.1 Classical charge transfer rate

The high temperature limit of classical charge transfer theory [33, 34] is often used as a trade-off between theoretical rigor and computational complexity. It captures key parameters which influence charge transport while at the same time providing an analytical expression for the rate. Within this limit, the transfer rate for a charge to hop from a site i to a site j reads

$$\omega_{ij} = \frac{2\pi}{\hbar} \frac{J_{ij}^2}{\sqrt{4\pi\lambda_{ij}k_{\text{B}}T}} \exp \left[-\frac{(\Delta E_{ij} - \lambda_{ij})^2}{4\lambda_{ij}k_{\text{B}}T} \right], \quad (2.31) \quad \text{equ:marcus}$$

where T is the temperature, $\lambda_{ij} = \lambda_{ij}^{\text{int}} + \lambda_{ij}^{\text{out}}$ is the **reorganization energy**, which is a sum of intra- and inter-molecular (outersphere) contributions, ΔE_{ij} is the **site-energy difference**, or driving force, and J_{ij} is the **electronic coupling element**, or transfer integral.

2.8.2 Semi-classical bimolecular rate

The main assumptions in eq. (2.31) are non-adiabaticity (small electronic coupling and charge transfer between two diabatic, non-interacting states), and harmonic promoting modes, which are treated classically. At ambient conditions, however, the intramolecular promoting mode, which roughly corresponds to C-C bond stretching, has a vibrational energy of $\hbar\omega \approx 0.2 \text{ eV} \gg k_B T$ and should be treated quantum-mechanically. The outer-sphere (slow) mode has much lower vibrational energy than the intramolecular promoting mode, and therefore can be treated classically. The weak interaction between molecules also implies that each molecule has its own, practically independent, set of quantum mechanical degrees of freedom. A more general, quantum-classical expression for a bimolecular multi-channel rate is derived in the Supporting Information of ref. [1] and has the following form

$$\omega_{ij} = \frac{2\pi}{\hbar} \frac{|J_{ij}|^2}{\sqrt{4\pi\lambda_{ij}^{\text{out}}k_B T}} \sum_{l', m'=0}^{\infty} |\langle \chi_{i0}^c | \chi_{il'}^n \rangle|^2 |\langle \chi_{j0}^n | \chi_{jm'}^c \rangle|^2 \exp \left\{ -\frac{[\Delta E_{ij} - \hbar(l'\omega_i^n + m'\omega_j^c) - \lambda_{ij}^{\text{out}}]^2}{4\lambda_{ij}^{\text{out}}k_B T} \right\}. \quad (2.32)$$

If the curvatures of intramolecular PES of charged and neutral states of a molecule are different, that is $\omega_i^c \neq \omega_i^n$, the corresponding reorganization energies, $\lambda_i^{cn} = \frac{1}{2}[\omega_i^n(q_i^n - q_i^c)]^2$ and $\lambda_i^{nc} = \frac{1}{2}[\omega_i^c(q_i^n - q_i^c)]^2$, will also differ. In this case the Franck-Condon (FC) factors for discharging of molecule i read [35]

$$|\langle \chi_{i0}^c | \chi_{il'}^n \rangle|^2 = \frac{2}{2^{l'} l'! (\omega_i^c + \omega_i^n)} \exp(-|s_i|) \left[\sum_{\substack{k=0 \\ k \text{ even}}}^{l'} \binom{l'}{k} \left(\frac{2\omega_i^c}{\omega_i^c + \omega_i^n} \right)^{k/2} \frac{k!}{(k/2)!} H_{l'-k} \left(\frac{s_i}{\sqrt{2S_i^{cn}}} \right) \right]^2, \quad (2.33)$$

where $H_n(x)$ is a Hermite polynomial, $s_i = 2\sqrt{\lambda_i^{nc}\lambda_i^{cn}}/\hbar(\omega_i^c + \omega_i^n)$, and $S_i^{cn} = \lambda_i^{cn}/\hbar\omega_i^c$. The FC factors for charging of molecule j can be obtained by substituting $(s_i, S_i^{cn}, \omega_i^c)$ with $(-s_j, S_j^{nc}, \omega_j^n)$. In order to evaluate the FC factors, the **internal reorganization energy** λ_i^{cn} can be computed from the intramolecular PES.

2.8.3 Semi-classical rate

One can also use the quantum-classical rate with a common set of vibrational coordinates [9]

$$\omega_{ij} = \frac{2\pi}{\hbar} \frac{|J_{ij}|^2}{\sqrt{4\pi\lambda_{ij}^{\text{out}}k_B T}} \sum_{N=0}^{\infty} \frac{1}{N!} \left(\frac{\lambda_{ij}^{\text{int}}}{\hbar\omega^{\text{int}}} \right)^N \exp \left(-\frac{\lambda_{ij}^{\text{int}}}{\hbar\omega^{\text{int}}} \right) \exp \left\{ -\frac{[\Delta E_{ij} - \hbar N\omega^{\text{int}} - \lambda_{ij}^{\text{out}}]^2}{4\lambda_{ij}^{\text{out}}k_B T} \right\}. \quad (2.34)$$

Numerical estimates show that if $\lambda_{ij}^{\text{int}} \approx \lambda_{ij}^{\text{out}}$ and $|\Delta E_{ij}| \ll \lambda_{ij}^{\text{out}}$ the rates are similar to those of eq. (2.31). In general, there is no robust method to compute $\lambda_{ij}^{\text{out}}$ [36] and both reorganization energies are often assumed to be of the same order of magnitude. In this case the second condition also holds, unless there are large differences in electron affinities or ionization potentials of neighboring molecules, e.g. in donor-acceptor blends.

To calculate rates of the type specified in `options.xml` for all pairs in the `neighbor list` and to save them into the `state.sql` file, run the `rates calculator`. Note that all required ingredients (`reorganization energies`, `transfer integrals`, and `site energies` have to be calculated before).

 **Calculation of transfer rates**
`| xtp_run -o options.xml -f state.sql -e rates`

2.9 Master equation

sec:kmc Having determined the list of conjugated segments (hopping sites) and charge transfer rates between them, the next task is to solve the master equation which describes the time evolution of the system

$$\frac{\partial P_\alpha}{\partial t} = \sum_\beta P_\beta \Omega_{\beta\alpha} - \sum_\beta P_\alpha \Omega_{\alpha\beta}, \quad (2.35) \quad \text{equ:master}$$

where P_α is the probability of the system to be in a state α at time t and $\Omega_{\alpha\beta}$ is the transition rate from state α to state β . A state α is specified by a set of site occupations, $\{\alpha_i\}$, where $\alpha_i = 1(0)$ for an occupied (unoccupied) site i , and the matrix $\hat{\Omega}$ can be constructed from rates ω_{ij} .

The solution of eq. (2.35) is obtained by using kinetic Monte Carlo (KMC) methods. KMC explicitly simulates the dynamics of charge carriers by constructing a Markov chain in state space and can find both stationary and transient solutions of the master equation. The main advantage of KMC is that only states with a direct link to the current state need to be considered at each step. Since these can be constructed solely from current site occupations, extensions to multiple charge carriers (without the mean-field approximation), site-occupation dependent rates (needed for the explicit treatment of Coulomb interactions), and different types of interacting particles and processes, are straightforward. To optimize memory usage and efficiency, a combination of the variable step size method [37] and the first reaction method is implemented.

To obtain the dynamics of charges using KMC, the program `xtp_kmc_run` executes a specific `calculator` after reading its options (charge carrier type, runtime, number of carriers etc.) from `options.xml`.

 **KMC for a single carrier in periodic boundary conditions**
`| xtp_kmc_run -o options.xml -f state.sql -e kmcsingle`

 **KMC for multiple carriers of the same type in periodic boundary conditions**
`| xtp_kmc_run -o options.xml -f state.sql -e kmcmultiple`

2.9.1 Extrapolation to nondispersive mobilities

sec:nondispersive

Predictions of charge-carrier mobilities in partially disordered semiconductors rely on charge transport simulations in systems which are only several nanometers thick. As a result, simulated charge transport might be dispersive for materials with large energetic disorder [38, 39] and simulated mobilities are system-size dependent. In time-of-flight (TOF) experiments, however, a typical sample thickness is in the micrometer range and transport is often nondispersive. In order to link simulation and experiment, one needs to extract the nondispersive mobility from simulations of small systems, where charge transport is dispersive at room temperature.

Such extrapolation is possible if the temperature dependence of the nondispersive mobility is known in a wide temperature range. For example, one can use analytical results derived for one-dimensional models [40–42]. The mobility-temperature dependence can then be parametrized by

simulating charge transport at elevated temperatures, for which transport is nondispersive even for small system sizes. This dependence can then be used to extrapolate to the nondispersive mobility at room temperature [43].

For Alq₃, the charge carrier mobility of a periodic system of 512 molecules was shown to be more than three orders of magnitude higher than the nondispersive mobility of an infinitely large system [43]. Furthermore, it was shown that the transition between the dispersive and nondispersive transport has a logarithmic dependence on the number of hopping sites N . Hence, a brute-force increase of the system size cannot resolve the problem for compounds with large energetic disorder σ , since N increases exponentially with σ^2 .

2.10 Stochastic Networks

sec:stochastic

The VOTCA package contains the functionality of generating large, amorphous charge transport networks ($\sim 10^6$ molecules). This is done with a combined coarse-grained and stochastic approach. VOTCA::CSG is used to generate a coarse-grained morphology. The stochastic modeling of VOTCA::CTP allows to make a charge transfer network out of this morphology by reproducing the neighbor list (connectivity), transfer integrals, correlated site energies. An overview is given in Figure 2.5.

Throughout this section we will use two state files. One is the state file `state_ref.sql` of the smaller reference system that can be generated as explained in the previous sections. The second one is the state file `state_cg.sql` of the coarse-grained system, or the stochastic network, that can be parametrized as explained in this section.

When using the stochastic functionalities, please cite the corresponding work:

1. B. Baumeier, O. Stenzel, C. Poelking, D. Andrienko, and V. Schmidt: Stochastic modeling of molecular charge transport networks. *Phys. Rev. B* 86, 184202 (2012)
2. P. Kordt and D. Andrienko: Modeling of Spatially Correlated Energetic Disorder in Organic Semiconductors. *Journal of Chemical Theory and Computation* 12, 36–40 (2016)
3. P. Kordt, J. J. M. van der Holst, M. Al Helwi, W. Kowalsky, F. May, A. Badinski, C. Lennartz, and D. Andrienko: Modeling of Organic Light Emitting Diodes: From Molecular to Device Properties. *Advanced Functional Materials* 25, 1955–1971 (2015)

2.10.1 Coarse-grained morphology

The first step is to generate a coarse-grained morphology. In this example, it is done by mapping a DPBIC molecule (which consists of 103 atoms) to a single point, its center of mass and by using the iterative Boltzmann inversion (IBI) method. Starting point is a smaller reference morphology, generated with GROMACS. Using the command

```
g_rdf -f traj.xtc -s topol.tpr
```

you can extract the radial distribution function $g(r)$ of your reference topology, outputted into the file `rdf.xvg`. This file, together with `table.xvg`, `grompp.mdp`, `topol.top`, `index.ndx` and `confout.gro` form your reference data.

For VOTCA::CSG you need a `setting.xml` file:

```
<cg>
  <!-- example for a non-bonded interaction entry -->
  <non-bonded>
    <!-- name of the interaction -->
    <name>IR-IR</name>
    <!-- types involved in this interaction -->
    <type1>IR</type1>
    <type2>IR</type2>
```



```

557 <!-- dimension + grid spacing of tables for calculations -->
558 <min>0.5</min>
559 <max>5.0</max>
560 <step>0.01</step>
561 <inverse>
562   <!-- target distribution (rdf), just give gromacs rdf.xvg -->
563   <target>rdf.xvg</target>
564   <!-- update cycles -->
565   <do_potential>1</do_potential>
566   <!-- additional post processing of dU before added to potential -->
567   <post_update>scale smooth</post_update>
568   <post_update_options>
569     <scale>0.5</scale> <!--Scale the potential before updating it -->
570     <smooth>
571       <iterations>2</iterations>
572     </smooth>
573   </post_update_options>
574   <!-- additional post processing of U after dU added to potential -->
575   <post_add></post_add>
576   <!-- name of the table for gromacs run -->
577   <gromacs>
578     <table>table_IR_IR.xvg</table>
579   </gromacs>
580 </inverse>
581 </non-bonded>
582
583 <!-- general options for inverse script -->
584 <inverse>
585   <!-- 300*0.00831451 gromacs units -->
586   <kBT>2.49435300</kBT>
587   <initial_configuration>maindir</initial_configuration>
588   <!-- use gromacs as simulation program -->
589   <program>gromacs</program>
590   <!-- gromacs specific options -->
591   <gromacs>
592     <!-- trash so many frames at the beginning -->
593     <equi_time>500</equi_time>
594     <!-- grid for table*.xvg !-->
595     <table_bins>0.001</table_bins>
596     <!-- cut the potential at this value (gromacs bug) -->
597     <pot_max>1000000</pot_max>
598     <!-- extend the tables to this value -->
599     <table_end>6.0</table_end>
600   </gromacs>
601   <!-- these files are copied for each new run -->
602   <filelist>grompp.mdp topol.top index.ndx table.xvg</filelist>
603   <!-- do so many iterations -->
604   <iterations_max>500</iterations_max>
605   <!-- Try to clean a bit -->
606   <cleanlist>traj.xtc</cleanlist>
607   <!-- ibm: inverse boltzmann imc: inverse monte carlo -->
608   <method>ibi</method>
609   <!-- write log to this file -->
610   <log_file>inverse.log</log_file>
611   <!-- write restart step to this file -->
612   <restart_file>restart_points.log</restart_file>
613   <!-- imc specific stuff -->
614 </inverse>

```

```
</cg>
```

You run IBI using the command

```
csg_inverse -options settings.xml
```

IBI intends to find a potential $U(r)$ that reproduces your radial distribution function. It is stored in the file **table_IR_IR.xvg** in our example.

With the interaction potential at hand, a large topology can be generated using molecular dynamics simulations for the coarse grained model. Starting point is a box with equally distributed points, with each point representing one molecule and with the number of points chosen such that the density of the reference system is reproduced. A small python script can generate the conf.gro to start from, here shown to obtain a $50 \times 50 \times 120 \text{ nm}^3$ starting morphology.

```
from pylab import *
import numpy as np

lenX      = 50
lenY      = 50
lenZ      = 120
originalV = 4704.339
originalN = 4000
spacing   = (originalV/originalN)**(1./3.)

molecule = "DPBIC"
resname    = "IRI"
atomname   = "IR"

newV = lenX*lenY*lenZ
newN = int(newV/originalV*originalN)

nX = int(lenX/spacing)+1
nY = int(lenY/spacing)+1
nZ = int(lenZ/spacing)+1

print "max. molecules in X direction: "+str(nX)
print "max. molecules in Y direction: "+str(nY)
print "max. molecules in Z direction: "+str(nZ)
print "total number of molecules: "+str(newN)

file = open("box.gro", "w")
file.write(molecule+"\n")
file.write(str(newN)+"\n")

atomnumber = 1
for iX in range(nX):
    for iY in range(nY):
        for iZ in range(nZ):
            if(atomnumber > newN):
                break
            posX = spacing*iX
            posY = spacing*iY
            posZ = spacing*iZ
            print ">> file, \"%5d%-5s%5s%5d%8.3f%8.3f%8.3f%8.4f%8.4f%8.4f\" % \\"
                  (1,resname,atomname,1,posX,posY,posZ,0,0,0)
            atomnumber += 1

file.write(" "+str(lenX)+" "+str(lenY)+" "+str(lenZ))
```

```

673 file.close()
674
675 print "Note: for some obscure reason VMD will not be able to read this file\
676 properly unless you open it once in vi and save it."
677

```

678 Open the `box.gro` in `vi` and save it (`:wq`), afterwards you can have a look at it in VDM. Run your
 679 MD simulations using the `mdrun` command. In the end you can compare the radial distribution
 680 functions of your reference and coarse-grained system, as shown in figure 2.6(a) as an example.

681 2.10.2 Charge transport network

682 To generate a charge transport network you first need a reference system with neighbor list, site
 683 energies and transfer integrals calculated and stored in a **state.sql** state file. The procedure for all
 684 these three properties is always the same: first analyze the reference data, and second import the
 685 analyzation files and reproduce the properties.

686 Neighbor list

687 In the atomistic reference system molecules are connected if their two closest segments are below
 688 a certain cut-off radius. This finer picture of segments does not exist in the coarse-grained system,
 689 where each molecule is represented by a point. To mimick the neighbor list, the probability of
 690 two molecules to be connected is analyzed as a function of their center-of-mass distance. This
 691 can be done by using the *panalyze* calculator

 **Analyze the pair connectivity (neighborlist) in the reference system**
 | `xtp_run -o options.xml -f state_ref.sql -e panalyze`

692 with the options defined as follows:

693 options_analyze.xml

```

694 <options>
695   <panalyze>
696     <resolution_space>0.05</resolution_space>
697   </panalyze>
698 </options>
699

```

702 The only parameter needed is the spacial resolution, i.e., the bin size for calculating the probabil-
 703 ities. The *panalyze* calculator outputs a file **panalyze.distanceprobability.out** with the respective
 704 probabilities. Now this file has to be imported into the coarse-grained state file

 **Import the reference pair connectivity (neighborlist) and reproduce it in stochastic network**
 | `xtp_run -o options.xml -f state_cg.sql -e neighborlist`

705 using the following options:

706 options_import.xml

```

707 <options>
708   <neighborlist>
709     <probabilityfile>panalyze.distanceprobability.out</probabilityfile>
710   </neighborlist>
711 </options>
712

```

715 For testing purposes, you can run the *panalyze* calculator on your coarse-grained state file and
 716 compare the probability function to the reference. An example is shown in figure 2.6(b). You
 717 can also also look at the file **panalyze.distanceprobability.out** for both state files, which has the
 718 distribution of coordination numbers (number of neighbors) and its average in.

Site energies

Site energies in amorphous organic semiconductors are roughly Gaussian distributed, with the width of the Gaussian, σ , called the energetic disorder. However, there are correlations between sites if they are close enough to each other. The aim in this section is therefore to reproduce the correlated energetic landscape. The first step is to get a spatial correlation function as well as the mean energy and the energetic disorder from your reference state file:

 **Analyze the energy distribution and correlation in the reference system**
 | `xtp_run -o options.xml -f state_ref.sql -e eanalyze`

with the following options:

options_analyze.xml

```
<options>
  <eanalyze>
    <resolution_sites>0.05</resolution_sites>
    <resolution_pairs>0.05</resolution_pairs>
    <resolution_space>0.3</resolution_space>
    <states>1,-1</states> <!-- +1 for hole transport, -1 for electron transport -->
    <distancemode>centreofmass</distancemode>
  </eanalyze>
</options>
```

The first three parameters determine bin sizes, then you can choose to look at hole and/or electron energy. The keyword *centreofmass* means, that the correlation function is calculated as a function of the centre-of-mass distance of molecules and not as a function of their nearest segments. For the stochastic simulations you always have to use the *centreofmass* mode!

The output files of this calculator that we need are **eanalyze.sitecorr_e.out** (for electrons) and **eanalyze.sitecorr_h.out** (for holes). In the second line of this file, you find mean and sigma of the energy distribution, as well as the mean of the static energies (without induction):

```
# EANALYZE: SPATIAL SITE-ENERGY CORRELATION
# AVG -0.4412655 STD 0.1739638 MIN_R 0.8365040 MAX_R 14.4771496 AVGESTATIC
-0.4730655
...
```

These values have to be inserted manually into the options file for importing to the coarse-grained system (see below). Apart from that, the file contains the spatial correlation function.

You generate energies following this distribution and correlation by using the *eimport* calculator

 **Import the energy distribution and correlation and reproduce it in stochastic network**
 | `xtp_run -o options.xml -f state_ref.sql -e eimport`

with the options:

options_import.xml

```
<options>
  <eimport>
    <probabilityfile_h>reference/eanalyze.sitecorr_h.out</probabilityfile_h>
    <sigma_h>0.1763163</sigma_h>
    <avgstatic_h>-0.5913265</avgstatic_h>
    <probabilityfile_e>reference/eanalyze.sitecorr_e.out</probabilityfile_e>
```

```

766         <sigma_e>0.1739638</sigma_e>
767         <avgestatic_e>-0.4730655</avgestatic_e>
768         <cutoff>8.5</cutoff>
769         <seed>1</seed>
770     </eimport>
771 </options>
772

```

773 The *cutoff* keyword can be used to read in the correlation function only up to a certain distance, which can be useful if larger distances yield unphysical results.

775 Transfer Integrals

776 The last ingredient reproduced by the stochastic approach are transfer integrals J . The idea is that $\log_{10}(J^2/\text{eV}^2)$ is roughly Gaussian distributed, with mean and error of the distribution varying with distance (see figure 2.6 (d)). Use the calculator

779

 **Analyze the distance-depend distribution of transfer integrals in the reference system**
| `xtp_run -o options.xml -f state_ref.sql -e ianalyze`

780 with options

781

782 options_analyze.xml

```

783 <options>
784   <ianalyze>
785     <resolution_logJ2>0.05</resolution_logJ2>
786     <resolution_space>0.05</resolution_space>
787     <states>1,-1</states> <!-- +1 for hole transport, -1 for electron transport -->
788   </ianalyze>
789 </options>
790
791

```

792 That will generate the files **ianalyze.ispatial_e.out** and **ianalyze.ispatial_h.out**, which contain means and errors as a function of centre-of-mass distance.

794

795 Now use the *iimport* calculator to generate transfer integrals in the coarse grained state file, following the same statistics.

796

 **Import distance dependent distribution of transfer integrals and reproduce in stochastic network**
| `xtp_run -o options.xml -f state_cg.sql -e iimport`

797 options_import.xml

```

798 <options>
799   <iimport>
800     <TI_tag></TI_tag>
801     <TI_file></TI_file>
802     <idft_jobs_file></idft_jobs_file>
803     <probabilityfile_h>reference/ianalyze.ispatial_h.out</probabilityfile_h>
804     <probabilityfile_e>reference/ianalyze.ispatial_e.out</probabilityfile_e>
805   </iimport>
806 </options>
807
808

```

809 einternal

810 Run the *einternal* calculator, just as you do it for the reference system.

Rates

If you followed the steps in this section, you have everything at hand to calculate charge transfer rates for the coarse grained system from the stochastic ingredients:



Calculate rates in the stochastic network

```
xtp_run -o options.xml -f state_cg.sql -e rates
```

Options are the same as for the reference file. You can check the result by comparing rates from your reference to the coarse-grained system, see figure 2.6(e) for an example. The resulting charge transport network can be used for kinetic Monte Carlo simulations with VOTCA. If everything goes well, mobilities for both systems should agree, as shown in figure 2.6(f).

2.11 Macroscopic observables

sec:analysis

Spatial distributions of charge and current densities can provide a better insight in the microscopic mechanisms of charge transport. If O is an observable which has a value O_α in a state α , its ensemble average at time t is a sum over all states weighted by the probability P_α to be in a state α at time t

$$\langle O \rangle = \sum_{\alpha} O_{\alpha} P_{\alpha}. \quad (2.36) \quad \text{equ:ensemble}$$

If O does not explicitly depend on time, the time evolution of $\langle O \rangle$ can be calculated as

$$\frac{d\langle O \rangle}{dt} = \sum_{\alpha, \beta} [P_{\beta} \Omega_{\beta\alpha} - P_{\alpha} \Omega_{\alpha\beta}] O_{\alpha} = \sum_{\alpha, \beta} P_{\beta} \Omega_{\beta\alpha} [O_{\alpha} - O_{\beta}]. \quad (2.37)$$

If averages are obtained from KMC trajectories, $P_{\alpha} = s_{\alpha}/s$, where s_{α} is the number of Markov chains ending in the state α after time t , and s is the total number of chains.

Alternatively, one can calculate time averages by analyzing a single Markov chain. If the total occupation time of the state α is τ_{α} then

$$\overline{O} = \frac{1}{\tau} \sum_{\alpha} O_{\alpha} \tau_{\alpha}, \quad (2.38) \quad \text{equ:time}$$

where $\tau = \sum_{\alpha} \tau_{\alpha}$ is the total time used for time averaging.

For ergodic systems and sufficient sampling times, ensemble and time averages should give identical results. In many cases, the averaging procedure reflects a specific experimental technique. For example, an ensemble average over several KMC trajectories with different starting conditions corresponds to averaging over injected charge carriers in a time-of-flight experiment. In what follows, we focus on the single charge carrier (low concentration of charges) case.

2.11.1 Charge density

sec:occupation

For a specific type of particles, the microscopic charge density of a site i is proportional to the occupation probability of the site, p_i

$$\rho_i = e p_i / V_i, \quad (2.39)$$

where, for an irregular lattice, the effective volume V_i can be obtained from a Voronoi tessellation of space. For reasonably uniform lattices (uniform site densities) this volume is almost independent of the site and a constant volume per site, $V_i = V/N$, can be assumed. In the macroscopic limit, the charge density can be calculated using a `sxtpthing` kernel function, i.e. a distance-weighted average over multiple sites. Site occupations p_i can be obtained from eq. (2.36) or eq. (2.38) by using the occupation of site i in state α as an observable.

If the system is in thermodynamic equilibrium, that is without sources or sinks and without circular currents (and therefore no net flux) a condition, known as detailed balance, holds

$$p_j \omega_{ji} = p_i \omega_{ij}, \quad (2.40) \quad \text{equ:detailed_balance}$$

It can be used to test whether the system is ergodic or not by correlating $\log p_i$ and the site energy E_i . Indeed, if $\lambda_{ij} = \lambda_{ji}$ the ratios of forward and backward rates are determined solely by the energetic disorder, $\omega_{ji}/\omega_{ij} = \exp(-\Delta E_{ij}/k_B T)$ (see eq. (2.31)).

2.11.2 Current

If the position of the charge, \vec{r} , is an observable, the time evolution of its average $\langle \vec{r} \rangle$ is the total current in the system

$$\vec{J} = e \langle \vec{v} \rangle = e \frac{d \langle \vec{r} \rangle}{dt} = e \sum_{i,j} p_j \omega_{ji} (\vec{r}_i - \vec{r}_j). \quad (2.41) \quad \text{equ:current_def}$$

Symmetrizing this expression we obtain

$$\vec{J} = \frac{1}{2} e \sum_{i,j} (p_j \omega_{ji} - p_i \omega_{ij}) \vec{r}_{ij}, \quad (2.42) \quad \text{equ:current}$$

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$. Symmetrization ensures equal flux splitting between neighboring sites and absence of local average fluxes in equilibrium. It allows to define a local current through site i as

$$\vec{J}_i = \frac{1}{2} e \sum_j (p_j \omega_{ji} - p_i \omega_{ij}) \vec{r}_{ij}. \quad (2.43) \quad \text{equ:site_current}$$

A large value of the local current indicates that the site contributes considerably to the total current. A collection of such sites thus represents most favorable charge pathways [44].

2.11.3 Mobility and diffusion constant

For a single particle, e.g. a charge or an exciton, a zero-field mobility can be determined by studying particle diffusion in the absence of external fields. Using the particle displacement squared, $\Delta \mathbf{r}_i^2$, as an observable we obtain

$$2dD_{\gamma\delta} = \frac{d \langle \Delta r_{i,\gamma} \Delta r_{i,\delta} \rangle}{dt} = \sum_{\substack{i,j \\ i \neq j}} p_j \omega_{ji} (\Delta r_{i,\gamma} \Delta r_{i,\delta} - \Delta r_{j,\gamma} \Delta r_{j,\delta}) = \sum_{\substack{i,j \\ i \neq j}} p_j \omega_{ji} (r_{i,\gamma} r_{i,\delta} - r_{j,\gamma} r_{j,\delta}). \quad (2.44) \quad \text{equ:diffusion}$$

Here \vec{r}_i is the coordinate of the site i , $D_{\gamma\delta}$ is the diffusion tensor, $\gamma, \delta = x, y, z$, and $d = 3$ is the system dimension. Using the Einstein relation,

$$D_{\gamma\delta} = k_B T \mu_{\gamma\delta}, \quad (2.45)$$

one can, in principle, obtain the zero-field mobility tensor $\mu_{\gamma\delta}$. Eq. (2.44), however, does not take into account the use of periodic boundary conditions when simulating charge dynamics. In this case, the simulated occupation probabilities can be compared to the solution of the Smoluchowski equation with periodic boundary conditions (see the supporting information for details).

Alternatively, one can directly analyze time-evolution of the KMC trajectory and obtain the diffusion tensor from a linear fit to the mean square displacement, $\overline{\Delta r_{i,\gamma} \Delta r_{i,\delta}} = 2dD_{\gamma\delta}t$.

The charge carrier mobility tensor, $\hat{\mu}$, for any value of the external field can be determined either from the average charge velocity defined in eq. (2.41)

$$\langle \vec{v} \rangle = \sum_{i,j} p_j \omega_{ji} (\vec{r}_i - \vec{r}_j) = \hat{\mu} \vec{F}, \quad (2.46)$$

or directly from the KMC trajectory. In the latter case the velocity is calculated from the unwrapped (if periodic boundary conditions are used) charge displacement vector divided by the total simulation time. Projecting this velocity on the direction of the field \vec{F} yields the charge carrier mobility in this particular direction. In order to improve statistics, mobilities can be averaged over several KMC trajectories and MD snapshots.

2.11.4 Spatial correlations of energetic disorder

Long-range, e.g. electrostatic and polarization, interactions often result in spatially correlated disorder [45], which affects the onset of the mobility-field (Poole-Frenkel) dependence [40, 46, 47]. To quantify the degree of correlation, one can calculate the spatial correlation function of E_i and E_j at a distance r_{ij}

$$C(r_{ij}) = \frac{\langle (E_i - \langle E \rangle) (E_j - \langle E \rangle) \rangle}{\langle (E_i - \langle E \rangle)^2 \rangle}, \quad (2.47)$$

where $\langle E \rangle$ is the average site energy. $C(r_{ij})$ is zero if E_i and E_j are uncorrelated and 1 if they are fully correlated. For a system of randomly oriented point dipoles, the correlation function decays as $1/r$ at large distances [48].

For systems with spatial correlations, variations in site energy differences, ΔE_{ij} , of pairs of molecules from the neighbor list are smaller than variations in site energies, E_i , of all individual molecules. Since only neighbor list pairs affect transport, the distribution of ΔE_{ij} rather than that of individual site energies, E_i , should be used to characterize energetic disorder.

Note that the `eanalyze` calculator takes into account *all* contributions to the site energies

```

 Analyze distribution and correlations of site energieies
| xtp_run -o options.xml -f state.sql -e eanalyze

```

2.12 Random Facts

This section is a collection of facts we discovered about xtp and ctp which should be included in the manual at some point, but lack proper background.

2.12.1 xqmm

The cutoffs used should not exceed the dimension of the cell, at least for a non orthogonal unit cell. XQMM throws an error if your cutoff is larger than the box, but it does not take the extension of the molecule into account, so often you may still have overlap.

XQMMM takes the segment coordinates from the MPS_Files so be very careful which MPS_File you put in.

2.12.2 EWALD

- Use `pewald3d`, as a calculator. I am not sure what the rest is for. All still use the ewald options. The shape factor massively influences the results. For bulk systems "none" is the option of choice. Other options are "xyslab", "sphere", "cube" but I do not know what they do.
- The induction cutoff should hardly ever exceed 3nm, because the calculation is expensive
- If you want to use induction, you before have to run `ewdbgp` calculator and specify `polar_top.bgp` in the options file for ewald. All the other parameters should be the same in `ewdbgp` and `pewald3d`

2.12.3 GW-BSE

There is a wide range of different approximations for *GW*-BSE. Here I try to summarize common abbreviations and methods and explain what our code does. This is not complete and certainly has some mistakes in it.

- COHSEX: RPA is only calculated for $\omega = 0$. This amounts to $\varepsilon(\bar{\mathbf{r}}, \bar{\mathbf{r}}', \omega) = \varepsilon(\bar{\mathbf{r}}, \bar{\mathbf{r}}')$. This is also called the static approximation to *GW*
- Plasmon Pole model; RPA is calculated mostly twice, once at $\omega = 0$ and $\omega = \omega_0$, then these values are used to fit an analytic model to interpolate $\varepsilon(\bar{\mathbf{r}}, \bar{\mathbf{r}}', \omega)$.
- Imaginary frequency integration, ε is numerically integrated along the Imaginary frequency axis. This is done because ε is much smoother along the Imaginary axis and thus requires less functional evaluations. Afterwards ε is extended to real frequencies via analytic continuation.

The calculation of *GW*. In VOTCA we also have the **shift** option. This is commonly called a scissor operator. This allows you to shift the unoccupied KS-orbitals up in energy, making the resulting spectrum closer to the *GW* spectrum. This is often a better starting point for the self-consistent evaluations.

- G_0W_0 . G and ε are calculated once from DFT results. W is evaluated once from that. Then energies ε_i^{GW} are corrected via:

$$\varepsilon_i^{GW} = \varepsilon_i^{KS} + Z_i \langle \phi_i^{KS} | \Sigma(\varepsilon_i^{KS}) - V_{xc} | \phi_i^{KS} \rangle \quad (2.48)$$

This is not implemented in VOTCA, because GW_0 is nearly as fast.

- GW_0 . ε is calculated once from DFT results. W is evaluated once from that. Then Σ is calculated. The resulting energies are fed back into Σ , until self-consistency is achieved. This is denoted **fixed** in VOTCA.

$$\varepsilon_i^{GW} = \varepsilon_i^{KS} + \langle \phi_i^{KS} | \Sigma(\varepsilon_i^{GW}) - V_{xc} | \phi_i^{KS} \rangle \quad (2.49)$$

- *scQPGW*. As GW_0 , but after ε_i^{GW} are converged, these energies are used to recalculate ε and then W , this is repeated until self-consistency is achieved. This is denoted **iterate** in VOTCA. This converges the “eigenvalues” of QP particles but along the $|\phi^{GW}\rangle$ states
- *scGW*. As *scQPGW* but instead of correcting only the energies, the full *Sigma* matrix is calculated and the eigenvalue problem for the QP is set up and solved and the whole system is then self-consistently solved in $|\phi^{GW}\rangle$ states and not in $|\phi^{KS}\rangle$. This fully converges the eigenvalues and eigenvectors of the QP particles in the space spanned by $|\phi^{KS}\rangle$, e.g. $|\phi^{GW}\rangle$ are linear combinations of $|\phi^{KS}\rangle$. This is reported to be unstable because the vertex corrections are important now. This is at the moment implemented in VOTCA.

The calculations in BSE

-

MORPHOLOGY		
	EXTRACT	REPRODUCE
RDF and coarse grained potential	GROMACS <code>g_rdf -f traj.xtc -s topol.tpr</code>	VOTCA::CSG <code>csg_inverse -options settings.xml</code>
morphology		GROMACS <code>mdrun</code>

RATES		
	EXTRACT	REPRODUCE
neighbor list (pairs)	<code>xtp_run -e panalyze -o options.xml -f state_reference.sql</code>	<code>xtp_run -e neighborlist -o options.xml -f state_cg.sql</code>
site energies	<code>xtp_run -e eanalyze -o options.xml -f state_reference.sql</code>	<code>xtp_run -e eimport -o options.xml -f state_cg.sql</code>
transfer integrals	<code>xtp_run -e ianalyze -o options.xml -f state_reference.sql</code>	<code>xtp_run -e iimport -o options.xml -f state_cg.sql</code>
rates	not used directly	<code>xtp_run -e rates -o options.xml -f state_cg.sql</code>

Figure 2.5: Stochastic Model in VOTCA. Overview of the different steps for generating stochastic charge transport networks in VOTCA. The Molecular Dynamics software GROMACS allows to analyze the radial distribution function of a morphology, which is then used by VOTCA::CSG to generate a coarse-grained potential that reproduces this distribution function. This potential can then be used for coarse-grained simulations in GROMACS. For calculating rates in the coarse-grained morphology, first the relevant parameters are extracted (panalyze, eanalyze, ianalyze) from the reference morphology and then reproduced in the coarse-grained morphology (neighborlist, eimport, iimport). With all these at hand, the rates calculator can be used in the coarse-grained morphology.



Figure 2.6: Comparison of the atomistic ($17 \times 17 \times 17$ nm³) and coarse-grained ($50 \times 50 \times 120$ nm³) models. (a) Radial distribution function, $g(r)$. (b) Probability of two sites to be connected (added to the neighbor list) as a function of their separation. (c) Spatial site energy autocorrelation function, $\kappa(r)$; Inset: Site energy distribution. (d) Mean m and width σ of a distribution of the logarithm of electronic couplings, $\log_{10}(J^2/eV^2)$, for molecules at a fixed separation r . (e) Rate distributions. (f) Mobility as a function of hole density, plotted for four different electric fields.

fig:stochastic

Chapter 3

Input and output files

3.1 Atomistic topology

If you are using GROMACS for generating atomistic configurations, it is possible to directly use the topology file provided by GROMACS (`topology.tpr`). In this case the GROMACS residue and atom names should be used to specify the coarse-grained topology and conjugated segments. A custom topology can also be defined using an XML file. Moreover, it is possible to partially overwrite the information provided in, for example, GROMACS topology file. We will illustrate how to create a custom topology file using DCV2T. The structure of DCV2T, together with atom type definitions, is shown in fig. 3.1. DCV2T has two thiophene (THI) and two dicyanovinyl (NIT) residues. The pdb file which contains residue types, residue numbering, atom names, atom types, and atom coordinates is shown in listing 3.1.



Figure 3.1: (a) DCV2T with atoms labelled according to `residue_number:residue_name:atom_name`. There are four residues and two residue types: thiophene (THI) and dicyanovinyl (NIT). The corresponding pdb file is shown in listing 3.1. Atom numbering is used to split conjugated segments on rigid fragments and to link atomistic ((b) from GROMACS topology) and quantum descriptions (c).

fig:dcv2t

list:pdb

Listing 3.1: pdb file of DCV2T.

949	HETATM	1	N1	NIT	1	2.388	8.533	11.066	1.00	4.14	N
950	HETATM	2	CN1	NIT	1	1.984	9.553	10.718	1.00	2.54	C
951	HETATM	3	N2	NIT	1	-1.138	10.872	10.087	1.00	3.24	N
952	HETATM	4	CN2	NIT	1	0.003	10.871	10.213	1.00	2.37	C
953	HETATM	5	CC1	NIT	1	1.441	10.824	10.327	1.00	1.91	C
954	HETATM	6	C1	NIT	1	2.193	11.939	10.071	1.00	1.61	C
955	HETATM	7	HN1	NIT	1	1.715	12.710	9.872	1.00	1.97	H
956	HETATM	8	S1	THI	2	4.758	10.743	10.130	1.00	1.52	S
957	HETATM	9	CA1	THI	2	3.613	12.024	9.948	1.00	1.22	C
958	HETATM	10	CA2	THI	2	6.099	11.836	9.997	1.00	1.30	C
959	HETATM	11	CB1	THI	2	4.251	13.243	9.782	1.00	1.39	C
960	HETATM	12	CB2	THI	2	5.658	13.131	9.818	1.00	1.45	C
961	HETATM	13	HC1	THI	2	3.800	14.047	9.660	1.00	1.66	H
962	HETATM	14	HC2	THI	2	6.230	13.860	9.731	1.00	1.74	H
963	HETATM	15	S1	THI	3	8.803	12.414	9.882	1.00	1.38	S
964	HETATM	16	CA1	THI	3	7.456	11.347	10.094	1.00	1.37	C
965	HETATM	17	CA2	THI	3	9.940	11.122	10.152	1.00	1.42	C
966	HETATM	18	CB1	THI	3	7.873	10.048	10.355	1.00	1.73	C
967	HETATM	19	CB2	THI	3	9.267	9.926	10.399	1.00	1.82	C
968	HETATM	20	HC1	THI	3	7.288	9.335	10.487	1.00	2.05	H
969	HETATM	21	HC2	THI	3	9.704	9.123	10.576	1.00	2.21	H
970	HETATM	22	N1	NIT	4	11.235	14.572	9.094	1.00	3.08	N
971	HETATM	23	CN1	NIT	4	11.665	13.566	9.441	1.00	2.04	C
972	HETATM	24	N2	NIT	4	14.733	12.005	10.009	1.00	2.17	N
973	HETATM	25	CN2	NIT	4	13.590	12.149	9.933	1.00	1.77	C
974	HETATM	26	CC1	NIT	4	12.156	12.282	9.861	1.00	1.71	C
975	HETATM	27	C1	NIT	4	11.363	11.220	10.154	1.00	1.59	C
976	HETATM	28	HN1	NIT	4	11.813	10.440	10.389	1.00	1.89	H

tab:map

Table 3.1: Description of the XML mapping file (`map.xml`).

topology	Definitions of molecules, segments, and fragments.
molecules	Container for all molecules.
molecule	Mapping of a single molecule.
name	Name of the molecule in the coarse-grained model.
ident	Name (identification) of the molecule in the all-atom representation. This must match the molecule name in the atomistic representation.
segments	Partitioning of the molecule on conjugated segments.
segment	Description of a conjugated segment.
name	Name of a conjugated segment in a molecule.
fragments	Container for all fragments in a segment.
fragment	Description of a rigid fragment.
name	Name of the rigid fragment in a conjugated segment
mdatoms	List of all atoms belonging to the rigid fragment in the format residue number:residue name:atom name.
qmatoms	List of atoms of the rigid fragment in its ground state geometry, atom number:atom type.
weights	Weights are used to determine the fragment center. The order should be the same as in the mdatoms and qmatoms definitions. If the mass of a nucleus in atomic mass units is used, the center of the rigid fragment will be its center of mass.
localframe	Three atoms which define a local frame for each rigid fragment.

979

3.2 Mapping file

sec:xmlmap

980

The mapping file (referred here as `map.xml`) is used by the program `xtp_map` to convert an atomistic trajectory to a trajectory with conjugated segments and rigid fragments. This trajectory is stored in a state file and contains positions, names, types of atoms belonging to rigid fragments. The description of the mapping options is given in table 3.1. An example of `map.xml` for a DCV2T molecule is shown in listing 3.2.

985

The file `map.xml` contains the whole electrostatic information about the molecules as well as the structural information. The `toolpdb2map` creates a `map.xml` from a `pdb` file and is a good starting point for further refinement.

987

list:map

Listing 3.2: Examl of `map.xml` for DCV2T. Each rigid fragment (coarse-grained bead) is defined by a list of atoms. Atom numbers, names, and residue names should correspond to those used in GROMACS topology (see the corresponding listing 3.1 of the `pdb` file).

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

```

<!-- this file is used to conver an atomistic trajectory to conjugated segments -->
<!-- molecules -->
<!-- molecule -->
  <!-- name of the conjugated molecule -->
  <!-- corresponding name of this molecule in the MD trajectory, should be
  the same as the name given at the end of topol.top -->
  <!-- segments -->
  <!-- segment -->
    <!-- name of the conjugated segment within the molecule -->
    <!-- QM coordinates of the conjugated segment -->
    <!-- IZINDO INPUT -->
    <!-- basisset -->
    <!-- orbitals -->
    <!-- torbital_h -->
    <!-- EMULTIPOLE INPUT -->
    <!-- multipoles_n -->
    <!-- multipoles_h -->
    <!-- map2md -->
    <!-- EINTERNAL INPUT -->
    <!-- U_c_c_n_h -->

```

```

1016 <U_nC_nN_h>0.1</U_nC_nN_h> <!-- Reorg. discharge -->
1017 <U_nC_nC_h>0.1</U_nC_nC_h> <!-- Reorg. charge -->
1018
1019 <!-- MD QM MP Mapping -->
1020 <fragments>
1021 <fragment>
1022 <name>N1</name> <!-- name of the rigid fragment within the segment -->
1023 <!-- list of atoms in the fragment resnum:resname:atomname -->
1024 <mdatoms>1:NIT:N1 1:NIT:CN1 1:NIT:N2 1:NIT:CN2 1:NIT:CC1 1:NIT:C1 1:NIT:HN1</mdatoms>
1025 <!-- corresponding ground state geometry atomnumber:atomtype read from .xyz file-->
1026 <qmatoms> 20:N 19:C 14:N 13:C 12:C 11:C 23:H </qmatoms>
1027 <!-- corresponding group state geometry multipoles read from .mps files -->
1028 <mpoles> 20:N 19:C 14:N 13:C 12:C 11:C 23:H </mpoles>
1029 <!-- weights to determine the fragment center (here CoM is used) -->
1030 <weights> 14 12 14 12 12 12 1 </weights>
1031 <!-- three atoms: define a cartesian local frame, two atoms: fragment is assumed to be
1032 rotationally invariant around the axis, one atom: fragment is assumed isotropic -->
1033 <localframe> 20 19 14 </localframe>
1034 <!-- Optional parameters (if not set <localframe> is used): used when atom labels in the .mps
1035 and .xyz file differ or more sites in the .mps file are used, so refers to <mpoles> -->
1036 <localframe_mps> 20 19 14 </localframe_mps>
1037 <!-- Optional parameters (if not set <localframe> is used): weights to determine the
1038 fragment center (here CoM is used), used when atom labels in the .mps and .xyz file
1039 differ or additional sites in the .mps file are used -->
1040 <weights_mps> 14 12 14 12 12 12 1 </
1041 weights_mps>
1042 <!-- Optional flag: says if a site is virtual or not, (virtual=1, real=0)-->
1043 <virtual_mps> 0 0 0 0 0 0 0 </
1044 virtual_mps>
1045 </fragment>
1046
1047 <fragment>
1048 <name>TH1</name>
1049 <mdatoms>2:THI:S1 2:THI:CA1 2:THI:CA2 2:THI:CB1 2:THI:CB2 2:THI:HC1 2:THI:HC2</mdatoms>
1050 <qmatoms> 7:S 8:C 6:C 9:C 10:C 24:H 25:H </qmatoms>
1051 <mpoles> 7:S 8:C 6:C 9:C 10:C 24:H 25:H </mpoles>
1052 <weights> 32 12 12 12 12 1 1 </weights>
1053 <localframe> 7 8 6 </localframe>
1054 </fragment>
1055
1056 <fragment>
1057 <name>TH2</name>
1058 <mdatoms>3:THI:S1 3:THI:CA1 3:THI:CA2 3:THI:CB1 3:THI:CB2 3:THI:HC1 3:THI:HC2</mdatoms>
1059 <qmatoms> 3:S 4:C 2:C 5:C 1:C 26:H 27:H </qmatoms>
1060 <weights> 32 12 12 12 12 1 1 </weights>
1061 <localframe> 3 4 2 </localframe>
1062 </fragment>
1063
1064 <fragment>
1065 <name>N12</name>
1066 <mdatoms>4:NIT:N1 4:NIT:CN1 4:NIT:N2 4:NIT:CN2 4:NIT:CC1 4:NIT:C1 4:NIT:HN1</mdatoms>
1067 <qmatoms> 22:N 21:C 18:N 17:C 16:C 15:C 28:H </qmatoms>
1068 <mpoles> 22:N 21:C 18:N 17:C 16:C 15:C 28:H </mpoles>
1069 <weights> 14 12 14 12 12 12 1 </weights>
1070 <localframe> 22 21 18 </localframe>
1071 </fragment>
1072 </fragments>
1073 </segment>
1074 </segments>
1075 </molecule>
1076 </molecules>
1077 </topology>

```

3.3 Molecular orbitals

If the semi-empirical method is used to calculate electronic coupling elements, molecular orbitals of all molecules must be supplied. They can be generated using Gaussian program. The Gaussian input file for DCV2T is shown in listing 3.3. Provided with this input, Gaussian will generate fort.7 file which contains the molecular orbitals of a DCV2T. This file can be renamed to DCV2T.orb. Note that the order of the atoms in the input file and the order of coefficients should always match. Therefore, the coordinate part of the input file must be supplied together with the orbitals. We will assume the coordinates, in the format atom_type: x y z, is saved to the DCV2T.xyz file.

**Be careful!**

Izindo requires the specification of orbitals for hole and electron transport in `map.xml`. They are the HOMO and LUMO respectively and can be retrieved from the `log` file from which the `DCV2T.orb` file is generated. The number of alpha electrons is the HOMO, the LUMO is HOMO+1

list:zindo_orbitals

Listing 3.3: Gaussian input file `get_orbitals.com` used for generating molecular orbitals. The first line contains the name of the check file, the second the requested RAM. `int=zindos` requests the method ZINDO, `punch=mo` states that the molecular orbitals ought to be written to the `fort.7` file, `nosymm` forbids use of symmetry and is necessary to ensure correct position of orbitals with respect to the provided coordinates. The two integer numbers correspond to the charge and multiplicity of the system: 0 1 corresponds to a neutral system with a multiplicity of one. They are followed by the types and coordinates of all atoms in the molecule.

```

1088 %chk=DCV2T.chk
1089 %mem=100Mb
1090 #p int=zindos punch=mo nosymm
1091
1092
1093 DCV2T molecular orbitals
1094
1095 0 1
1096 S      -1.44650      2.12185      0.00135
1097 C      -2.43098      0.58936     -0.00048
1098 C      -1.59065     -0.51859     -0.00146
1099 C      -0.21222     -0.22233     -0.00095
1100 C       0.07761      1.13376      0.00040
1101 S       2.87651      0.79316      0.00148
1102 C       3.86099      2.32565      0.00235
1103 C       3.02066      3.43359      0.00231
1104 C       1.64223      3.13733      0.00162
1105 C       1.35240      1.78125      0.00114
1106 C      -3.85350      0.52245     -0.00081
1107 C      -4.79569      1.52479     -0.00008
1108 C      -6.18500      1.18622     -0.00117
1109 C      -4.47544      2.91565      0.00081
1110 C       5.28350      2.39256      0.00296
1111 C       6.22569      1.39020      0.00327
1112 C       7.61500      1.72876      0.00432
1113 C       5.90542     -0.00064      0.00333
1114 N      -7.32389      0.89743     -0.00195
1115 N      -4.21872      4.06274      0.00142
1116 N       8.75389      2.01754      0.00510
1117 N       5.64864     -1.14772      0.00361
1118 H      -1.98064     -1.52966     -0.00256
1119 H       0.55785     -0.98374     -0.00169
1120 H       3.41065      4.44466      0.00272
1121 H       0.87216      3.89874      0.00147
1122 H      -4.24640     -0.49192     -0.00188
1123 H       5.67641      3.40692      0.00337

```

3.4 Monomer calculations for DFT transfer integrals

list:edft_gaussian_xml

Listing 3.4: Example `package.xml` file for the Gaussian package required in the options of the `edft calculator` for the monomer calculations as preparation for the determination of transfer integrals using DIPRO.

```

1126 <package>
1127   <name>gaussian</name>
1128   <executable>g09</executable>
1129   <checkpoint></checkpoint>
1130

```

```

1131 <scratch></scratch>
1132
1133 <charge>0</charge>
1134 <spin>1</spin>
1135 <options># pop=minimal pbepbe/6-311g** scf=tight punch=mo nosymm test</options>
1136 <memory>1Gb</memory>
1137 <threads>2</threads>
1138
1139 <cleanup></cleanup>
1140 </package>

```

list:edft_turbomole.xml

Listing 3.5: Example package.xml file for the Turbomole package required in the options of the edft calculator for the monomer calculations as preparation for the determination of transfer integrals using DIPRO.

```

1142 <package>
1143 <name>turbomole</name>
1144 <executable>ridft</executable>
1145 <scratch>/tmp</scratch>
1146
1147 <options>
1148 TITLE
1149 a coord
1150 *
1151 no
1152 b all def-TZVP
1153 *
1154 eht
1155 y
1156 0
1157 y
1158 dft
1159 on
1160 func
1161 pbe
1162 grid
1163 m3
1164 *
1165 ri
1166 on
1167 m 300
1168 *
1169 scf
1170 conv
1171 7
1172 iter
1173 200
1174
1175 marij
1176
1177 q
1178 </options>
1179
1180 <cleanup></cleanup>
1181 </package>

```

list:edft_nwchem.xml

Listing 3.6: Example package.xml file for the NWChem package required in the options of the edft calculator for the monomer calculations as preparation for the determination of transfer integrals using

DIPRO.

```

1184
1185 <package>
1186   <name>nwchem</name>
1187   <executable>nwchem</executable>
1188   <checkpoint></checkpoint>
1189   <scratch>/tmp/nwchem</scratch>
1190   <charge>0</charge>
1191   <spin>1</spin>
1192   <threads>1</threads>
1193   <memory></memory>
1194   <options>
1195     start
1196     basis
1197     * library 6-311gss
1198     end
1199     memory 1500 mb
1200
1201     dft
1202       xc xpbe96 cpbe96
1203       direct
1204       iterations 100
1205       noprint "final vectors analysis"
1206     end
1207     task dft
1208   </options>
1209   <cleanup></cleanup>
1210 </package>

```

3.5 Pair calculations for DFT transfer integrals

list:ldft_gaussian.xml

Listing 3.7: Example package.xml file for the Gaussian package required in the options of the `ldft calculator` for the pair calculations and the determination of transfer integrals using DIPRO.

```

1213
1214 <package>
1215   <name>gaussian</name>
1216   <executable>g09</executable>
1217   <checkpoint></checkpoint>
1218   <scratch></scratch>
1219
1220   <charge>0</charge>
1221   <spin>1</spin>
1222   <options># pop=minimal pbepbe/6-311g** nosymm IOp(3/33=1,3/36=-1) punch=mo guess=cards scf=
1223   <memory>1Gb</memory>
1224   <threads>1</threads>
1225
1226   <cleanup></cleanup>
1227 </package>

```

list:ldft_turbomole.xml

Listing 3.8: Example package.xml file for the Turbomole package required in the options of the `ldft calculator` for the pair calculations and the determination of transfer integrals using DIPRO.

```

1229
1230 <package>
1231   <name>turbomole</name>
1232   <executable>ridft</executable>
1233   <scratch>/tmp</scratch>
1234

```

```

1235 <options>
1236 $intsdebug cao
1237 a coord
1238 *
1239 no
1240 b all def-TZVP
1241 *
1242 eht
1243 y
1244 0
1245 y
1246 dft
1247 on
1248 func
1249 pbe
1250 grid
1251 m3
1252 *
1253 ri
1254 on
1255 m 300
1256 *
1257 scf
1258 conv
1259 7
1260 iter
1261 1
1262 diis
1263 3
1264 damp
1265 0.00
1266
1267
1268
1269 marij
1270
1271 q
1272 </options>
1273
1274 <cleanup></cleanup>
1275 </package>

```

list:ldft_nwchem.xml

Listing 3.9: Example package.xml file for the NWChem package required in the options of the `ldft` calculator for the pair calculations and the determination of transfer integrals using DIPRO.

```

1277
1278 <package>
1279 <name>nwchem</name>
1280 <executable>nwchem</executable>
1281 <checkpoint></checkpoint>
1282 <scratch>/tmp/nwchem</scratch>
1283 <charge>0</charge>
1284 <spin>1</spin>
1285 <memory></memory>
1286 <threads>1</threads>
1287 <options>
1288 start
1289 basis
1290 * library 6-311gss
1291 end

```

```

1292 memory 1500 mb
1293
1294 dft
1295   print "ao overlap"
1296   xc xpbe96 cpbe96
1297   direct
1298   iterations 1
1299   convergence nodamping nodiis
1300   noprint "final vectors analysis"
1301   vectors input system.movecs
1302 end
1303 task dft
1304 </options>
1305   <cleanup></cleanup>
1306 </package>
1307

```

1308 3.6 DFT transfer integrals

list:TI.xml

Listing 3.10: Example TI.xml file created as the output of a DIPRO calculation. Due to slightly different implementations, the orbitals indices refer to monomer indices in a Gaussian run but to indices in the merged dimer guess in a Turbomole run.

```

1309 <pair name="pair_100_155">
1310   <parameters>
1311     <HOMO_A>162</HOMO_A>
1312     <NoccA>1</NoccA>
1313     <LUMO_A>164</LUMO_A>
1314     <NvirtA>1</NvirtA>
1315     <HOMO_B>161</HOMO_B>
1316     <NoccB>1</NoccB>
1317     <LUMO_B>163</LUMO_B>
1318     <NvirtB>1</NvirtB>
1319   </parameters>
1320   <transport name="hole">
1321     <channel name="single">
1322       <J>1.546400416750696E-003</J>
1323       <e_A>-6.30726450715697</e_A>
1324       <e_B>-6.36775613794166</e_B>
1325     </channel>
1326     <channel name="multi">
1327       <molecule name="A">
1328         <e_HOMOm0>-6.30726450715697</e_HOMOm0>
1329       </molecule>
1330       <molecule name="B">
1331         <e_HOMOm0>-6.36775613794166</e_HOMOm0>
1332       </molecule>
1333       <dimer name="integrals">
1334         <T_00>1.546400416750696E-003</T_00>
1335         <J_sq_degen>2.391354248926727E-006</J_sq_degen>
1336         <J_sq_boltz>2.391354248926727E-006</J_sq_boltz>
1337       </dimer>
1338     </channel>
1339   </transport>
1340   <transport name="electron">
1341     <channel name="single">
1342       <J>-2.797473760331286E-003</J>
1343       <e_A>-4.50318366770689</e_A>
1344

```

```

1345         <e_B>-4.53143397059021</e_B>
1346     </channel>
1347     <channel name="multi">
1348         <molecule name="A">
1349             <e_LUMOp0>-4.50318366770689</e_LUMOp0>
1350         </molecule>
1351         <molecule name="B">
1352             <e_LUMOp0>-4.53143397059021</e_LUMOp0>
1353         </molecule>
1354         <dimer name="integrals">
1355             <T_00>-2.797473760331286E-003</T_00>
1356             <J_sq_degen>7.825859439742066E-006</J_sq_degen>
1357             <J_sq_boltz>7.825859439742066E-006</J_sq_boltz>
1358         </dimer>
1359     </channel>
1360 </transport>
1361 </pair>
1362

```

3.7 State file

1363

sec:statefile

1364 All data structures are saved to the `state.sql` file in sqlite3 format, see <http://www.sqlite.org/>.
1365 They are available in form of tables in the `state.sql` file as can be seen by the command
1366 `sqlite3 state.sql ".tables "`
1367 An example of such a table are molecules. The full table can be displayed using the command
1368 (similar for the other tables)
1369 `sqlite3 state.sql "SELECT * FROM molecules "`
1370 The meaning of all the entries in the table can be displayed by a command like
1371 `sqlite3 state.sql ".SCHEMA molecules "`
1372 The first and second entry are integers for internal and regular id of the molecule and the third
1373 entry is the name. A single field from the table like the name of the molecule can be displayed by
1374 a command like
1375 `sqlite3 state.sql "SELECT name FROM molecules "`
1376 Besides molecules, the following tables are stored in the `state.sql`:
1377 `conjseg_properties`:
1378 Conjugated segments are stored with id, name and x,y,z coordinates of the center of mass in nm.
1379 `conjsegs`:
1380 Reorganization energies for charging or discharging a conjugated segment are stored together
1381 with the coulomb energy and any other user defined energy contribution (in eV) and occupation
1382 probabilities.
1383 `pairs`:
1384 The pairs from the neighborlist are stored with the pair id, the id of the first and second segment,
1385 the rate from the first to the second , the rate from the second to the first (both in s⁻¹) and the
1386 x,y,z coordinates in nm of the distance between the first and the second segment.
1387 `pairintegrals`:
1388 Transfer integrals for all pairs are stored in the following way: The pair id , the number for count-
1389 ing possible different electronic overlaps (e.g if only the frontier orbitals are taken into account
1390 this is always zero, while an effective value is stored in addition to the different overlaps of e.g.
1391 HOMO-1 and HOMO-1 if more frontier orbitals are taken into account) and the integral in eV.
1392 `pairproperties`:
1393 The outer sphere reorganization energy of all pairs is stored by an id, the pair id, a string `lambda_outer`
1394 and the energy in eV.
1395 `conjsegs`:
1396 Conjugated segments are saved in the following way: The id, the name, the type, the molecule
1397 id, the time frame, the x,y,z coordinates in nm and the occupation probability.

```

1398 conjseg_properties:
1399 Properties of the conjugated segments like reorganization energies for charging or discharging a
1400 charge unit or the coulomb contribution to the site energy are stored by: id, conjugated segment
1401 id, a string like lambda_intra_charging, lambda_intra_discharging or energy_coulomb
1402 and a corresponding value in eV.
1403 The tables rigidfrag_properties, rigidfrags and frames offer information about rigid
1404 fragments and time frames including periodic boundary conditions.
1405 The data in the state.sql file can also be modified by the user. Here is an example how to
1406 modify the transfer integral between the conjugated segments number one and two assuming
1407 that they are in the neighborlist. Their pair id can be found by the command
1408 pair_ID='sqlite3 state.sql "SELECT _id FROM pairs WHERE conjseg1=1 AND conjseg2=2"'
1409 The old value of the transfer integral can be deleted using
1410 sqlite3 state.sql "DELETE FROM pair_integrals WHERE pair=$pair_ID"
1411 Finally the new transfer integral  $J$  can be written to the state.sql file by the command
1412 sqlite3 state.sql "INSERT INTO pair_integrals (pair,num,J) VALUES ($pair_ID,0,$J) "
1413 Here the num=0 indicates that only the effective transfer integrals is written to the file, while other
1414 values of num would correspond to overlap between other orbitals than the frontier orbitals.
1415 In a similar way the coulomb contribution to the site energy of the first conjugated segment can
1416 be overwritten by first getting its id
1417 c_ID='sqlite3 state.sql "SELECT _id from conjseg_properties where conjseg=1 AND
1418 key =\"energy_coulomb\""
1419 Then deleting the old value
1420 sqlite3 state.sql "DELETE FROM conjseg_properties WHERE _id=$c_ID"
1421 Then the new coulomb energy  $E$  can be written to this id
1422 sqlite3 state.sql "INSERT INTO conjseg_properties (_id,conjseg,key,value)
1423 VALUES ($c_ID,1,\"energy_coulomb\",$E) "
1424 Finally the resulting coulomb contribution to all conjugated segments can be displayed by
1425 sqlite3 state.sql "SELECT * from conjseg_properties WHERE key=\"energy_coulomb\""
1426

```


1427 Chapter 4

1428 Reference

sec:reference

1429 4.1 Programs

sec:programs

1430 Programs execute specific tasks (calculators).

1431 4.1.1 xtp_map

prog:xtp_map

1432 Generates QM|MD topology

1433 -h [--help] display this help and exit
1434 -v [--verbose] be loud and noisy
1435 -t [--topology] arg topology
1436 -c [--coordinates] arg coordinates or trajectory
1437 -s [--segments] arg definition of segments and fragments
1438 -f [--file] arg state file

1439 4.1.2 xtp_run

prog:xtp_run

1440 Runs excitation/charge transport calculators

1441 -h [--help] display this help and exit
1442 -v [--verbose] be loud and noisy
1443 -o [--options] arg calculator options
1444 -f [--file] arg sqlight state file, *.sql
1445 -i [--first-frame] arg (=1) start from this frame
1446 -n [--nframes] arg (=1) number of frames to process
1447 -t [--nthreads] arg (=1) number of threads to create
1448 -s [--save] arg (=1) whether or not to save changes to state file
1449 -e [--execute] arg List of calculators separated by ',' or ''
1450 -l [--list] Lists all available calculators
1451 -d [--description] arg Short description of a calculator

1452 4.1.3 xtp_tools

prog:xtp_tools

1453 Runs excitation/charge transport tools

1454 -h [--help] display this help and exit
1455 -v [--verbose] be loud and noisy
1456 -o [--options] arg calculator options
1457 -t [--nthreads] arg (=1) number of threads to create Tools:
1458 -e [--execute] arg List of tools separated by ',' or ''
1459 -l [--list] Lists all available tools

```
1460     -d [ --description ] arg Short description of a tool
```

1461 4.1.4 xtp_parallel

prog:xtp_parallel

1462 Runs job-based heavy-duty calculators

```
1463     -h [ --help ] display this help and exit
1464     -v [ --verbose ] be loud and noisy
1465     -o [ --options ] arg calculator options
1466     -f [ --file ] arg sqlite state file, *.sql
1467     -i [ --first-frame ] arg (=1) start from this frame
1468     -n [ --nframes ] arg (=1) number of frames to process
1469     -t [ --nthreads ] arg (=1) number of threads to create
1470     -s [ --save ] arg (=1) whether or not to save changes to state file
1471     -r [ --restart ] arg restart pattern: 'host(pc1:234) stat(FAILED)'
1472     -c [ --cache ] arg (=8) assigns jobs in blocks of this size
1473     -j [ --jobs ] arg (=run) task(s) to perform: input, run, import
1474     -m [ --maxjobs ] arg (=1) maximum number of jobs to process (-1 = inf)
1475     -e [ --execute ] arg List of calculators separated by ',' or ''
1476     -l [ --list ] Lists all available calculators
1477     -d [ --description ] arg Short description of a calculator
```

1478 4.1.5 xtp_dump

prog:xtp_dump

1479 Extracts information from the state file

```
1480     -h [ --help ] display this help and exit
1481     -v [ --verbose ] be loud and noisy
1482     -o [ --options ] arg calculator options
1483     -f [ --file ] arg sqlight state file, *.sql
1484     -i [ --first-frame ] arg (=1) start from this frame
1485     -n [ --nframes ] arg (=1) number of frames to process
1486     -t [ --nthreads ] arg (=1) number of threads to create
1487     -s [ --save ] arg (=1) whether or not to save changes to state file Extractors:
1488     -e [ --extract ] arg List of extractors separated by ',' or ''
1489     -l [ --list ] Lists all available extractors
1490     -d [ --description ] arg Short description of an extractor
```

1491 4.1.6 xtp_testsuite

prog:xtp_testsuite

1492 Performs tests en suite + optional arguments:

```
1493     -h, --help show this help message and exit
1494     -e [ [ ...]], --execute [ [ ...]] Tests to perform, accepts regex (def=".*")
1495     -l, --listonly List all tests available, then quit.
1496     -x , --xml Test-suite file (def="$VOTCASHARE/xtp/xml/testsuite.xml")
1497     -s , --source Test source input directory (def="source")
1498     -td , --testdirectory Test run directory (def="suite")
1499     -t , --target Directory where to store targets (def="targets")
1500     -r , --reference Folder with reference data to compare to (def="reference")
1501     -g, --generate Generate reference from targets (def=False)
1502     -cmp, --compareonly Only compare existing targets to reference (def=False)
1503     -v, --verbose The wordy version (def=False)
1504     -sh, --showoutput Display VOTCA::XTP exec. output (def=False)
1505     -c, --clean To clean or not to clean test dir. (def=False)
1506     -m , --mailto Mail the result. (def=False)
```

4.1.7 xtp_update

Updates the state file + optional arguments:

```
-h, --help show this help message and exit
-f SQLFILE, --file SQLFILE State file to update.
```

4.1.8 xtp_update_exciton

Updates the state file for singlets and triplets + optional arguments:

```
-h, --help show this help message and exit
-f SQLFILE, --file SQLFILE State file to update.
```

4.1.9 xtp_basisset

xtp_update, version 1.5-dev gitid: 9f8c877 Creates votca xml basissetfiles from NWChem basis-setfiles optional arguments:

```
-h, --help show this help message and exit
-f INPUT, --input INPUT NWchem file containing the basisset or Turbomole folder
with element names.
-o OUTPUTFILE, --outputvotca OUTPUTFILE Path of votca outputfile
-t TURBOBASIS, --turbomolebasisset TURBOBASIS For turbomole specify the ba-
sisset that is supposed to be extracted from Files, for auxbasis sets the basisset the aux ba-
sisset is supposed to be used for.
```

4.1.10 xtp_makeauxbasis

xtp_update, version 1.5-dev gitid: 9f8c877 Creates votca xml aux-basissetfiles from votca basis-setfiles optional arguments:

```
-h, --help show this help message and exit
-f BASISFILE, --basisfile BASISFILE xtp basissetfile
-o OUTFILE, --outfile OUTFILE optional file to write basisset to
-g GROUPING, --grouping GROUPING Cutoff at which basisfunctions are grouped to-
gether in deviation from the arithmetic mean; Default 0.1
-c CUTOFF, --cutoff CUTOFF Cutoff for very localised basisfunctions; Default 60
-e ELEMENT, --element ELEMENT Print out only the element specified
-l LMAX, --lmax LMAX maximum angular momentum in aux basisset: Default:4
```

4.2 Calculators

Calculator is a piece of code which computes specific system properties, such as site energies, transfer integrals, etc. `xtp_run`, `xtp_kmc_run` are wrapper programs which executes such calculators. The generic syntax is

```
xtp_run -e "calc1, calc2, ..." -o options.xml
```

File `options.xml` lists all options needed to run a specific calculator. The format of this file is explained in listing 4.1. A complete list of calculators is given in the `calculators` reference section.

Listing 4.1: A part of the `options.xml` file with options for the `calculator_name{1,2}` calculators.

```
<calculator_name1>
  <option1>value1</option1>
  <option2>value2</option2>
  ...
</calculator_name1>
```

```
1550 <calculator_name2>
1551     <option1>value1</option1>
1552     <option2>value2</option2>
1553     ...
1554 </calculator_name2>
1555 ...
1556
```

1557 A list of all calculators and their short descriptions can be obtain using

```
1558 xtp_run --list
```

1559 A detailed description of all options of a specific calculator(s) is available via

```
1560 xtp_run --desc calc1,calc2,...
```

1561 **4.3 Common options**

ref:options

name	Description of the option
------	---------------------------

Bibliography

- ruhle_microscopic_2011 [1] V. R  ijhle *et al.*, J. Chem. Theory Comput. **7**, 3335 (2011), bibtex: ruhle_microscopic_2011.
- ruhle_versatile_2009 [2] V. R  ijhle *et al.*, Journal of Chemical Theory and Computation **5**, 3211 (2009), bibtex: ruhle_versatile_2009.
- gamma_design_1995 [3] E. Gamma, R. Helm, and R. E. Johnson, *Design Patterns. Elements of Reusable Object-Oriented Software.*, 1st ed., reprint. ed. (Addison-Wesley Longman, Amsterdam, ADDRESS, 1995), bibtex: gamma_design_1995.
- ruhle_multiscale_2010 [4] V. R  ijhle, J. Kirkpatrick, and D. Andrienko, The Journal of Chemical Physics **132**, 134103 (2010), bibtex: ruhle_multiscale_2010.
- vukmirovic_charge_2008 [5] N. Vukmirovi   and L.-W. Wang, The Journal of Chemical Physics **128**, 121102 (2008), bibtex: vukmirovic_charge_2008.
- vukmirovic_charge_2009 [6] N. Vukmirovi   and L.-W. Wang, Nano Letters **9**, 3996 (2009), bibtex: vukmirovic_charge_2009.
- mcmahon_ad_2009 [7] D. P. McMahon and A. Troisi, Chemical Physics Letters **480**, 210 (2009), bibtex: mcmahon_ad_2009.
- bredas_charge-transfer_2004 [8] J.-L. Br    das, D. Beljonne, V. Coropceanu, and J. Cornil, Chemical Reviews **104**, 4971 (2004), bibtex: bredas_charge-transfer_2004.
- may_relationship_2011 [9] F. May *et al.*, Journal of Materials Chemistry **21**, 9538 (2011), bibtex: may_relationship_2011.
- breneman_determining_1990 [10] C. M. Breneman and K. B. Wiberg, Journal of Computational Chemistry **11**, 361 (1990), bibtex: breneman_determining_1990.
- stone_distributed_1985 [11] A. Stone and M. Alderton, Molecular Physics **56**, 1047 (1985), bibtex: stone_distributed_1985.
- chirlian_atomic_1987 [12] L. E. Chirlian and M. M. Francl, Journal of Computational Chemistry **8**, 894 (1987), bibtex: chirlian_atomic_1987.
- singh_approach_1984 [13] U. C. Singh and P. A. Kollman, Journal of Computational Chemistry **5**, 129 (1984), bibtex: singh_approach_1984.
- stone_distributed_2005 [14] A. J. Stone, J. Chem. Theory Comput. **1**, 1128 (2005), bibtex: stone_distributed_2005.
- stone_theory_1997 [15] A. J. Stone, *The Theory of intermolecular forces* (Clarendon Press, Oxford, 1997), bibtex: stone_theory_1997.
- thole_molecular_1981 [16] B. Thole, Chemical Physics **59**, 341 (1981), bibtex: thole_molecular_1981.
- van_duijnen_molecular_1998 [17] P. T. van Duijnen and M. Swart, The Journal of Physical Chemistry A **102**, 2399 (1998), bibtex: van_duijnen_molecular_1998-1.
- applequist_atom_1972 [18] J. Applequist, J. R. Carl, and K.-K. Fung, Journal of the American Chemical Society **94**, 2952 (1972), bibtex: applequist_atom_1972.
- ren_polarizable_2003 [19] P. Ren and J. W. Ponder, The Journal of Physical Chemistry B **107**, 5933 (2003), bibtex: ren_polarizable_2003.
- baessler_charge_1993 [20] H. Baessler, Physica Status Solidi B **175**, 15 (1993), bibtex: baessler_charge_1993.
- troisi_charge-transport_2006 [21] A. Troisi and G. Orlandi, Phys. Rev. Lett. **96**, (2006), bibtex: troisi_charge-transport_2006.
- troisi_charge_2009 [22] A. Troisi, D. L. Cheung, and D. Andrienko, Physical Review Letters **102**, 116602 (2009), bibtex: troisi_charge_2009.
- mcmahon_organic_2010 [23] D. P. McMahon and A. Troisi, ChemPhysChem **11**, 2067 (2010), bibtex: mcmahon_organic_2010.
- vehoff_charge_2010 [24] T. Vehoff *et al.*, The Journal of Physical Chemistry C **114**, 10592 (2010), bibtex: vehoff_charge_2010.
- baumeier_density-functional_2010 [25] B. Baumeier, J. Kirkpatrick, and D. Andrienko, Physical Chemistry Chemical Physics **12**, 11103 (2010), bibtex: baumeier_density-functional_2010.

- kirkpatrick_approximate_2008 [26] J. Kirkpatrick, International Journal of Quantum Chemistry **108**, 51 (2008), bibtex: kirkpatrick_approximate_2008.
- walker_electrical_2002 [27] A. B. Walker, A. Kambili, and S. J. Martin, Journal of Physics: Condensed Matter **14**, 9825 (2002), bibtex: walker_electrical_2002.
- borsenberger_charge_1991 [28] P. M. Borsenberger, L. Pautmeier, and H. Bässler, The Journal of Chemical Physics **94**, 5447 (1991), bibtex: borsenberger_charge_1991.
- pasveer_unified_2005 [29] W. F. Pasveer *et al.*, Physical Review Letters **94**, 206601 (2005), bibtex: pasveer_unified_2005.
- bredas_molecular_2009 [30] J.-L. Brédas, J. E. Norton, J. Cornil, and V. Coropceanu, Accounts of Chemical Research **42**, 1691 (2009), bibtex: bredas_molecular_2009.
- coropceanu_charge_2007 [31] V. Coropceanu *et al.*, Chemical Reviews **107**, 926 (2007), bibtex: coropceanu_charge_2007.
- nelson_modeling_2009 [32] J. Nelson, J. J. Kwiatkowski, J. Kirkpatrick, and J. M. Frost, Accounts of Chemical Research **42**, 1768 (2009), bibtex: nelson_modeling_2009.
- marcus_electron_1993 [33] R. A. Marcus, Reviews of Modern Physics **65**, 599 (1993), bibtex: marcus_electron_1993.
- hutchison_hopping_2005 [34] G. R. Hutchison, M. A. Ratner, and T. J. Marks, Journal of the American Chemical Society **127**, 2339 (2005), bibtex: hutchison_hopping_2005.
- chang_new_2005 [35] J.-L. Chang, Journal of Molecular Spectroscopy **232**, 102 (2005), bibtex: chang_new_2005.
- hoffman_reorganization_1996 [36] B. M. Hoffman and M. A. Ratner, Inorganica Chimica Acta **243**, 233 (1996), bibtex: hoffman_reorganization_1996.
- bortz_new_1975 [37] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, Journal of Computational Physics **17**, 10 (1975), bibtex: bortz_new_1975.
- scher_anomalous_1975 [38] H. Scher and E. Montroll, Physical Review B **12**, 2455 (1975), bibtex: scher_anomalous_1975.
- borsenberger_role_1993 [39] P. M. Borsenberger, E. H. Magin, M. D. VanAuweraer, and F. C. D. Schryver, Physica Status Solidi A **140**, 9 (1993), bibtex: borsenberger_role_1993.
- derrida_velocity_1983 [40] B. Derrida, Journal of Statistical Physics **31**, 433 (1983), bibtex: derrida_velocity_1983.
- cordes_one-dimensional_2001 [41] H. Cordes *et al.*, Physical Review B **63**, 094201 (2001), bibtex: cordes_one-dimensional_2001.
- seki_electric_2001 [42] K. Seki and M. Tachiya, Physical Review B **65**, 014305 (2001), bibtex: seki_electric_2001.
- lukyanov_extracting_2010 [43] A. Lukyanov and D. Andrienko, Physical Review B **82**, 193202 (2010), bibtex: lukyanov_extracting_2010.
- van_der_holst_modeling_2009 [44] J. J. M. van der Holst *et al.*, Physical Review B **79**, 085203 (2009), bibtex: van_der_holst_modeling_2009.
- dunlap_charge-dipole_1996 [45] D. Dunlap, P. Parris, and V. Kenkre, Physical Review Letters **77**, 542 (1996), bibtex: dunlap_charge-dipole_1996.
- novikov_essential_1998 [46] S. V. Novikov *et al.*, Physical Review Letters **81**, 4472 (1998), bibtex: novikov_essential_1998.
- nagata_atomistic_2008 [47] Y. Nagata and C. Lennartz, The Journal of Chemical Physics **129**, 034709 (2008), bibtex: nagata_atomistic_2008.
- novikov_cluster_1995 [48] S. V. Novikov and A. V. Vannikov, The Journal of Physical Chemistry **99**, 14573 (1995), bibtex: novikov_cluster_1995.