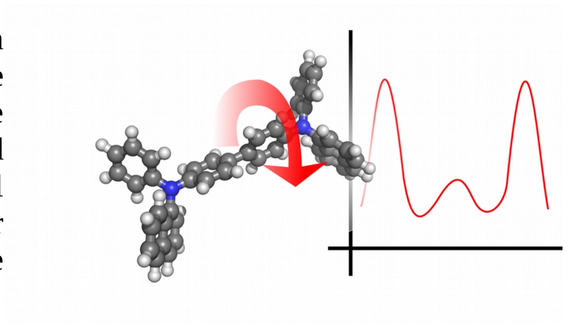
Parametrizer & DihedralParametrizer

Parametrizer and Dihedralparametrizer are the first steps in the predictive multiscale modeling approach and parametrize single molecules on a quantum-mechanical level. The Parametrizer module optimizes molecular geometries and calculates non-bonded interaction parameters for classical simulations (e.g. Deposit) and the DihedralParametrizer generates customized dihedral force fields for flexible molecules.



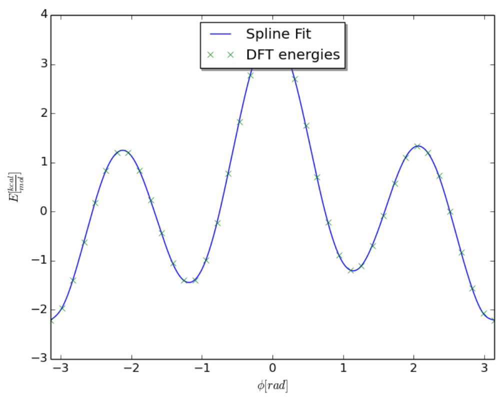
Program Specifications

Input	- Initial guess of molecular geometry (mol2-file)
Output	- Optimized molecular geometry (pdb file) - Force-field files for Deposit containing Lennard-Jones parameters, partial charges and dihedral force field - Plots of dihedral potentials and dihedral force-fields
Requirements	- Python 2.7 - Turbomole 6.x or 7.x or Gaussian 09 - openmpi

Method

The Parametrizer module performs a geometry optimization using DFT with basis set and functional set by the user. For a single point calculation on the optimized geometry, partial charges are computed using either electrostatic potential fit (ESP-fit) or Mullikan-approach.

For the computation of the dihedral potential, each dihedral is stepwise rotated around 360°. For each configuration the geometry is relaxed with the respective dihedral angle kept fixed and the total energy is calculated using DFT, again with basis set and functional set by the user. A spline is fitted to the data points for efficient computation of energies during a classical simulation later on in the OLED workflow.



Dihedral potential of the molecule Succinonitrile, computed by the DihedralParametrizer module.

Performance

The Parametrizer for geometry optimization and computation of partial charges runs well on single core in a few minutes to hours (depending on the quality of the initial guess for the atom positions). The speed-up due to parallelization of the DihedralParametrizer scales linearly with the number of cores on a single node up to the total number of reference points for the spline fit. If, for example, 32 data points are intended for the spline fit, an optimal use of resources is to request 32 cores per node.