THE SCINE READUCT DEVELOPERS:

MORITZ BENSBERG, CHRISTOPH BRUNKEN, KATJA-SOPHIA CSIZI, STEPHANIE GRIMMEL, STEFAN GUGLER, JAN-GRIMO SOBEZ, MIGUEL STEINER, PAUL TÜRTSCHER, JAN UNSLEBER, ALAIN C. VAUCHER, THOMAS WEYMUTH, AND MARKUS REIHER

USER MANUAL

SCINE READUCT 6.0.0

ETH ZÜRICH

Contents

Introduction

The SCINE project requires stable algorithms for the refinement of elementary-reaction paths and associated transition-state structures. The SCINE ReaDuct module was designed to serve this purpose and can be driven from SCINE Chemoton. However, as with all SCINE modules it is a stand-alone program that can drive standard quantum chemical software.

SCINE ReaDuct is a command-line tool that allows to carry out structure optimizations, transition state searches and intrinsic reaction coordinate (IRC) calculations among other things. For these calculations, it relies on a backend program to provide the necessary quantum chemical properties (such as nuclear gradients). Currently, SCINE Sparrow[1], Gaussian[2], ORCA[3], Turbomole[4], CP2K[5], Serenity[6], and xtb[7] are supported as backend programs.

In this manual, we describe the installation of the software, an example calculation as a hands-on introduction to the program, and the most import functions and options.[8] A prospect on features in future releases and references for further reading are added at the end of this manual.

[1] Husch, T.; Vaucher, A. C.; Reiher, M. Semiempirical molecular orbital models based on the neglect of diatomic differential overlap approximation, *Int. J. Quantum Chem.* **2018,** *118,* e25799

[2] Frisch, M. J. *et al.* "Gaussian 09, revision D. 01", 2009 Gaussian, Inc., Wallingford CT

[3] Neese, F. The ORCA program system, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012,** *2,* 73–78

[4] "TURBOMOLE V7.4.1 2019, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from `http://www.turbomole.com`.",

[5] Kühne, T. D. *et al.* CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations, *J. Chem. Phys.* **2020,** *152,* 194103

[6] Unsleber, J. P.; Dresselhaus, T.; Klahr, K.; Schnieders, D.; Böckers, M.; Barton, D.; Neugebauer, J. Serenity: A subsystem quantum chemistry program, *J. Comput. Chem.* **2018,** *39,* 788–798

[7] Bannwarth, C.; Caldeweyher, E.; Ehlert, S.; Hansen, A.; Pracht, P.; Seibert, J.; Spicher, S.; Grimme, S. Extended tight-binding quantum chemistry methods, *WIREs Comput. Mol. Sci.* **2020,** *11,* e01493

[8] Throughout this manual, the most import information is displayed in the main text, whereas useful additional information is given as a side note like this one.

Obtaining the Software

ReaDuct is distributed as open source software in the framework of the SCINE project (www.scine.ethz.ch). Visit our website (www.scine.ethz.ch/download/readuct) to obtain the software.

System Requirements

ReaDuct itself has only modest requirements regarding the hardware performance. However, the underlying quantum-chemical calculations might become resource intensive if extremely large systems are studied. We advise to first explore the software with the fast semiempirical methods provided in ReaDuct. This allows one to quickly understand what to expect from the software rather than being confused by possibly long times waiting for more involved quantum chemical calculations to finish.

Installation

ReaDuct is distributed as an open source code. In order to compile ReaDuct from this source code, you need

- a C++ compiler supporting the C++17 standard (GCC at least 7.3.0),
- CMake (at least 3.9.0),
- the Boost library (at least 1.65.0), and
- the Eigen3 library (at least 3.3.2).

In order to compile the software, clone the repository with git, change to the source directory and execute the following steps:

```
git submodule init
git submodule update
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SPARROW=ON -DCMAKE_INSTALL_PREFIX=../install ..
make
make test
make install
export SCINE_MODULE_PATH=$PWD/../install/lib
export PATH=$PATH:$PWD/../install/bin
```

This will configure everything, compile your software, run the tests, and install the software into the folder "install". Finally, it will add the ReaDuct binary to your `PATH` such that you can use it without having to specify its full location.

In case you need support with the setup of ReaDuct, please contact us by writing to scine@phys.chem.ethz.ch.

Using the Standalone Binary

ReaDuct is a command-line-only binary; there is no graphical user interface. Therefore, you always work with the ReaDuct binary on a command line such as the Gnome Terminal or KDE Konsole.

All functionality is accessed via an input file following the YAML syntax. The program is then run with the command

```
readuct -c <input file>
```

where you have to give the actual filename of your input file for `<input file>`.

General Structure of the Input File

The general structure of a ReaDuct input file is as follows:

```
systems:
  - name: [system name]
    path: [path to coordinates file]
    program: [program name]
    method_family: [method_family name]
    settings:
      [settings key]: [settings value]
      ...

tasks:
  - type: [task type name]
    input: [input system name]
    output: [output system name]
    settings:
      [settings key]: [settings value]
      ...
```

There are two major blocks, namely a `systems` block and a `tasks` block. You can define multiple systems in the `systems` block and multiple tasks in the `tasks` block (see also section Task Chaining).

A system is a combination of nuclear coordinates (given as an XYZ file), a calculation program (such as SCINE Sparrow or ORCA), a method family (such as DFT). Depending on the program and method family used, different settings (such as molecular charge, spin multiplicity, and convergence thresholds) can be given. A task specifies that a certain calculation type (such as a structure optimization) should be carried out with a given (input) system. Different tasks can have different settings. For every task, an output system can be assigned to be used in further tasks (for instance, the output system of a structure optimization task contains the optimized nuclear coordinates).

For example, in order to do a simple structure optimization, you can use the following input file:

```
systems:
  - name: water
    path: h2o.xyz
    program: Sparrow
    method_family: PM6
    settings:
      molecular_charge: 0
      spin_multiplicity: 1

tasks:
  - type: geoopt
    input: [water]
    output: [water_opt]
    settings:
      optimizer: bfgs
```

This specifies a system named `water`, the nuclear coordinates are given by the XYZ file `h2o.xyz`. Any calculation performed on this system will use the PM6 method provided by SCINE Sparrow. For this system, a structure optimization will be carried out; the structure will be optimized with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

Supported Programs and Methods

SCINE SPARROW

SCINE SPARROW is fully supported by SCINE READUCT. If built with the CMake option `-DBUILD_SPARROW=ON` as described in section Installation, it will be automatically downloaded and integrated into READUCT at compile time.

In order to use SCINE SPARROW with READUCT, specify `program: Sparrow` in the respective system block and the desired calculation method family (such as `PM6`) in the `method_family` key. All semiempirical methods are considered their own family of methods. All options supported by SPARROW can be defined in the settings block. See the SPARROW manual for a complete list of these options (the option names are identical to the command line option names of the SPARROW standalone binary).

External Programs

SCINE READUCT supports calculations with ORCA[9] (version 4.2.0), TURBOMOLE[10] (version 7.4.1), CP2K (8.1)[11], GAUSSIAN[12] (version 09 Rev. D01), SERENITY[13] (1.4), and xtb (6.4.1)[14]. Support is currently not fully tested. For each program, there might be specific calculation types and/or settings which do not work. Also, we cannot guarantee compatibility with any version different from the one's mentioned above since we have no control over the output format of an external program. If you encounter any problems when using one of these software packages together with READUCT, please write a short message to scine@phys.chem.ethz.ch.

For these programs, the following settings can be applied.

- `molecular_charge`: This specifies the molecular charge. It can take on values between -10 and 10; by default, it is zero.
- `spin_multiplicity`: This specifies the spin multiplicity. It can take on values between 1 and 10; by default, it is 1.
- `spin_mode`: The spin mode that should be employed for the calculation. Available spin modes are: `restricted, unrestricted, restricted_open_shell, none` and `any`, where in the latter case, the program determines the spin mode automatically. Note that not every program supports all of these options.

[9] Neese, F. The ORCA program system, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012,** *2*, 73–78

[10] "TURBOMOLE V7.4.1 2019, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from `http://www.turbomole.com`.",

[11] Kühne, T. D. *et al.* CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations, *J. Chem. Phys.* **2020,** *152*, 194103

[12] Frisch, M. J. *et al.* "Gaussian 09, revision D. 01", 2009 Gaussian, Inc., Wallingford CT

[13] Unsleber, J. P.; Dresselhaus, T.; Klahr, K.; Schnieders, D.; Böckers, M.; Barton, D.; Neugebauer, J. SERENITY: A subsystem quantum chemistry program, *J. Comput. Chem.* **2018,** *39*, 788–798

[14] Bannwarth, C.; Caldeweyher, E.; Ehlert, S.; Hansen, A.; Pracht, P.; Seibert, J.; Spicher, S.; Grimme, S. Extended tight-binding quantum chemistry methods, *WIREs Comput. Mol. Sci.* **2020,** *11*, e01493

- `external_program_nprocs`: The number of processors to use in the calculations. By default, it is one, *i.e.*, a serial calculation is carried out. For ORCA, please note that you have to specify the full ORCA binary path in case you want to do a parallel calculation. For SERENITY, this setting is currently not supported.
- `base_working_directory`: This specifies the directory in which the files for the calculations will be stored. By default, this is set to the current directory. For each calculation, a new directory will be created inside the directory specified by `base_working_directory` to keep the files related to that specific calculation. Note that this setting is not available for xtb.
- `self_consistence_criterion`: The threshold to which the electronic energy should be converged (given in hartree). The default is `1.0e-7` (*i.e.*, 10^{-7} hartree) for all programs.
- `max_scf_iterations`: The maximum number of SCF iterations allowed by the program. By default, it is 100. Note that this keyword is currently not available in conjunction with GAUSSIAN.
- `electronic_temperature`: The electronic temperature used for the calculation. By default, it is 0 K, except for xtb, in which case it is 300 K.
- `temperature`: The temperature used for the calculation of thermochemical data. By default, it is 298.15 K. Note that the calculation of thermochemistry data is currently not supported for GAUSSIAN.

A form of dispersion correction must be specified within the method following a hypen, e.g. `PBE-D3BJ`. The supported dispersion correction between programs may vary. Additional program-specific settings are compiled in the subsequent sections.

ORCA

In order to use ORCA with READUCT, specify `program: ORCA` in the respective system block and the desired calculation method family (e.g., `DFT`) in the `method_family` key. The method (e.g., `PBE`) can be specified in the system settings. You can carry out a Hartree–Fock calculation by specifying both the `method_family` and `method` to be `HF`.

Implicit solvation can be activated specifying the `solvent` and `solvation` key. To model implicit solvation, the Conductor-like Polarizable Continuum Model[15] (C-PCM) or the Solvation Model based on Density[16] (SMD) each with the default solvent dependent settings implemented

[15] Barone, V.; Cossi, M. Quantum Calculation of Molecular Energies and Energy Gradients in Solution by a Conductor Solvent Model, *J. Phys. Chem. A* **1998,** *102,* 1995–2001

[16] Marenich, A. V.; Cramer, C. J.; Truhlar, D. G. Universal solvation model based on solute electron density and on a continuum model of the solvent defined by the bulk dielectric constant and atomic surface tensions, *J. Phys. Chem. B.* **2009,** *113,* 6378–6396

in ORCA can be used. These solvent dependent settings cannot be changed.

The path to the ORCA binary must be set via the environment variable `ORCA_BINARY_PATH`.

You can specify the following settings in the settings block:

- `method`: This specifies the calculation method to be used. Note that the name of the method should match the string used in a typical ORCA input file. By default, it is `PBE`.
- `basis_set`: This specifies the basis set string. By default, it is `def2-SVP`. You can specify any valid ORCA basis set string (see the ORCA manual for a complete list).
- `scf_damping`: This specifies whether SCF damping is switched on to aid SCF convergence. By default, it is set to `false`.
- `solvent`: This specifies the solvent for implicit solvation. By default, it is empty. You can specify any valid ORCA C-PCM solvent name.
- `solvation`: This specifies the implicit solvation model. By default, it is empty. You can specify `cpcm` or `smd`.
- `external_program_memory`: The total amount of memory in MB that should be available for ORCA to use. By default set to `1024`.
- `orca_filename_base`: This specifies the basic filename (prefix) used for all files related to the ORCA calculations. By default, it is set to `orca_calc`; therefore, the generated input file will be named `orca_calc.inp`.
- `point_charges_file`: Sets the filename of an ORCA point charges file (containing the values and coordinates of external point charges to be included in the calculation[17]). By default it is set to an empty string, which results in no point charges considered.
- `delete_tmp_files`: Whether temporary files (i.e., all files with a ".tmp" extension) should be deleted in case the calculation fails. By default, this is set to `true`.
- `hessian_calculation_type`: The way to calculate a Hessian, i.e., either "analytical" or "numerical". By default, it is set to "analytical", unless it is already specified in our code that ORCA does not provide an analytical Hessian for the specified method. However, this list is not complete and may vary from version to version.

[17] For the formatting of an ORCA point charges file see: https://sites.google.com/site/orcainputlibrary/geometry-input

- `special_option`: A possibility to add an input that is not covered by our settings, only recommended for experts.
- `perform_broken_symmetry_calculation`: Whether a broken-symmetry DFT calculation should be performed. By default, this is set to `false`. Note that there is no guarantee that the calculation converges to the desired broken-symmetry solution. This needs to be checked *a posteriori* by inspection of the atomic spin populations or the spin density or analyzing the unrestricted corresponding orbitals.
- `initial_spin_multiplicity`: The high-spin multiplicity for which the SCF should be converged before flipping spin-density at one or multiple local sites.
- `spin_flip_sites`: The atom indices of the sites at which local spin density should be flipped after converging to the high-spin solution. By default, this is set to an empty list, *i.e.*, needs to be specified explicitly.
- `calculate_moessbauer`: Whether to calculate the ^{57}Fe Mössbauer parameters. The obtained parameters comprise the quadrupole splitting ΔE_Q, the asymmetry parameter η, and the electron density ρ_0 for each Fe nucleus.

Turbomole

In order to use Turbomole with ReaDuct, specify `program: TURBOMOLE` in the respective system block and the desired calculation method family (e.g., `DFT`) in the `method_family` key. The method (e.g., `PBE`) can be specified in the system settings. You can carry out a Hartree–Fock calculation by specifying both the `method_family` and `method` to be `HF`.

Implicit solvation can be activated specifying the `solvent` key. To model implicit solvation, the Conductor-like Screening Model [18] (COSMO) with the default solvent dependent settings implemented in Turbomole is used. These solvent dependent settings cannot be changed.

[18] Klamt, A.; Schüürmann, G. COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient, *J. Chem. Soc. Perkin Trans. 2* **1993,** 799–805

The path to the directory containing the Turbomole binaries must be set via the environment variable `TURBODIR`.

You can specify the following settings in the settings block:

- `method`: This specifies the calculation method to be used. By default, it is `PBE`. Dispersion correction can be added to this method

string, e.g: `PBE-D3BJ`. Current supported dispersion corrections are D3[19], D3BJ[20], and D4[21]. Note that the name of the method should match the string in a typical TURBOMOLE input file, but without a delimiter, e.g: "B3LYP" instead of "B3-LYP".

- `basis_set`: This specifies the basis set string. By default, Turbomole assigns the `def-SV(P)` basis set. You can specify any valid TURBOMOLE basis set string from the Karlsruhe basis sets[22], Dunning's correlation-consistent basis sets[23] or Pople-style basis sets[24] (see the TURBOMOLE manual for a complete list).
- `solvation`: This specifies the implicit solvation model. By default, it is empty. Currently, you can specify `cosmo`.
- `solvent`: This specifies the solvent for implicit solvation. By default, it is empty. Current supported solvents are: `water/h2o`, `acetone`, `benzene` , `dmso`, `ethanol`, `methanol`, `hexane`, `toluene`, `ammonia`, `chloroform`, `nitrobenzene`, `aceticacid`, `acetonitrile`, `aniline`, `benzylalcohol`, `bromoform`, `butanol`, `isobutanol`, `tertbutanol`, `carbondisulfide`, `carbontetrachloride`, `cyclohexane`, `cyclohexanone`, `dichlorobenzene`, `diethylether`, `dioxane`, `dmfa`, `ethylacetate`, `dichloroethane`, `ethyleneglycol`, `formicacid`, `isopropanol` and `thf`. A custom solvent may be specified as `user_defined(epsilon, probe-radius)`, where probe-radius denotes the radius of the solvent sphere used to construct the molecular cavity in angstrom and epsilon is the solvent's dielectric constant.
- `scf_damping`: This specifies whether SCF damping should be applied to aid SCF convergence. Available settings are `true` and `false`. By default set to `false`.
- `scf_damping_value`: The exact vlue to be used for the SCF damping. The default value is 0.5.
- `scf_orbitalshift`: Shifts either the energies of unoccupied MOs to higher energies or, in open-shell cases, the energies of closed-shell MOs to lower energies to aid convergence. The default is set to `0.1` hartree.
- `steer_orbitals`: This specifies whether the orbitals should be perturbed following a randomized scheme after a converged single point calculation[25]. By default, this is set to `false`.
- `point_charges_file`: Sets the filename of a TURBOMOLE point charges file (containing the values and coordinates of external point charges to be included in the calculation). The point charges file should be formatted such that the first line contains the overall number of point charges to include in the calculation, followed

[19] Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, *J. Chem. Phys.* **2010,** *132,* 154104

[20] Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the damping function in dispersion corrected density functional theory, *J. Comput. Chem.* **2011,** *32,* 1456–1465

[21] Caldeweyher, E.; Ehlert, S.; Hansen, A.; Neugebauer, H.; Spicher, S.; Bannwarth, C.; Grimme, S. A generally applicable atomic-charge dependent London dispersion correction, *J. Chem. Phys.* **2019,** *150,* 154122

[22] Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy, *Phys. Chem. Chem. Phys.* **2005,** *7,* 3297–3305

[23] Dunning, T. H. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen, *J. Chem. Phys.* **1989,** *90,* 1007–1023

[24] Ditchfield, R.; Hehre, W. J.; Pople, J. A. Self-consistent molecular-orbital methods. IX. An extended gaussian-type basis for molecular-orbital studies of organic molecules, *J. Chem. Phys.* **1971,** *54,* 720–723

[25] Vaucher, A. C.; Reiher, M. Steering Orbital Optimization out of Local Minima and Saddle Points Toward Lower Energy, *J. Chem. Theory Comput.* **2017,** *13,* 1219–1228

by the coordinates and value of the external point charge in the following format: `<x> <y> <z> <q>`, each in one line. By default, this setting is set to an empty string, which results in no point charges being considered.

- `enable_ri`: Enables the Resolution of the Identity (RI) approximation. The default is set to `true`.

CP2K

In order to use CP2K with ReaDuct, specify `program: CP2K` in the respective system block and the desired calculation method family (e.g., `DFT`') in the `method_family` key. The method (e.g., `PBE`) can be specified in the system settings. Dispersion correction can be added to this method string, e.g: `PBE-D3BJ`. Current supported dispersion corrections are:

- D2
- D3
- D3BJ
- DRSLL
- LMKLL
- RVV10

Implicit solvation is currently not supported.

The path to the CP2K binary must be set via the environment variable `CP2K_BINARY_PATH`.

You can specify the following settings in the settings block:

- `method`: This specifies the calculation method to be used. Note that the name of the method should match the string used in a typical CP2K input file. By default, it is `PBE`.
- `basis_set`: This specifies the basis set string. By default, it is `DZVP-MOLOPT-GTH`. Currently, only MOLOPT basis sets are supported. It is not necessary to specify the SR part of the basis set name.
- `periodic_boundaries`: This specifies the periodic boundaries, with the cell lengths and angles in a string separated by ',', e.g., '3.0,3.0,3.0,90.0,90.0,90.0' to specify a cubic box of a length of 3 Bohr. The default is a cube with a length of 15 Angstrom.

- `plane_wave_cutoff`: Sets the plane wave cutoff of the finest grid in Ry. The default is 300.
- `relative_multi_grid_cutoff`: Determines the grid at which a Gaussian is mapped, giving the cutoff in Ry used for a Gaussian with exponent (denoted alpha in the CP2K manual) of one. The default is 60.
- `n_grids`: Sets the desired number of grids. The default is set to 5.
- `additional_mos`: Specify the number of additional molecular orbitals. By default, it is set to zero.
- `orbital_transformation`: Specify an orbital transformation minimizer, such as "cg". By default, it is set to "none", which deactivates orbital transformation. The available options are:
  - broyden
  - cg
  - diis
  - sd
- `outer_scf`: Maximum number of outer SCF iterations. The default is 0, which deactivate outer SCF cycles. The standard setting "max_scf_iterations" available for all different programs, specifies the number of inner SCF cycles for CP2K
- `poisson_solver`: Specify the poisson solver. "none" picks the CP2K default solver based on the periodicity, which is the default. The available options are:
  - analytic
  - implicit
  - mt
  - multipole
  - periodic
  - wavelet
- `allow_unconverged_scf`: Whether an unconverged SCF throws an error or not. By default, it is set to "false", which means an error is thrown.
- `scf_guess`: The guess for the SCF start as provided by CP2K. The default is "restart" which defaults to "atomic" if no restart is available in CP2K v8.1. The available options are

  - restart
  - atomic
  - core
  - history_restart
  - mopac
  - random

- `scf_damping`: This specifies whether SCF damping is switched on to aid SCF convergence. By default, it is set to `broyden_mixing`. The possible options are:
  - broyden_mixing
  - broyden_mixing_new
  - direct_p_mixing
  - kerker_mixing
  - multisecant_mixing
  - none_mixing
  - pulay_mixing

- `cp2k_filename_base`: This specifies the basic filename (prefix) used for all files related to the CP2K calculations. By default, it is set to "cp2k_calc"; therefore, the generated input file will be named "cp2k_calc.inp".

- `additional_output_file`: Sets the filename of an CP2K file containing additional matrix output if bond orders were requested. By default it is set to "additional_output", which results in a generated file named "additional_output-1.0.Log". If this setting is identical to "cp2k_filename_base", no separate files will be generated.

- `delete_tmp_files`: Whether temporary files (i.e., all files with a ".bak" extension) should be deleted in case the calculation fails. By default, this is set to `true`.

- `electronic_temperature`: The temperature used for Fermi smearing. By default, it is 0.0 K and therefore deactivated. Setting it to a value also requires to set the number of `additional_mos`.

Gaussian

In order to use Gaussian with ReaDuct, specify `program: GAUSSIAN` in the respective system block and the desired calculation method family (e.g., `DFT`) in the `method_family` key. The method (e.g., `PBEPBE`) can be specified in the system settings. Dispersion correction can be added to this method string, e.g: `PBE-D3BJ`. Current supported dispersion corrections are D2, D3[26] and D3BJ.[27]

Implicit solvation can be activated specifying the `solvent` key. To model implicit solvation, per default the Conductor-like Polarizable Continuum Model[28] (C-PCM) with the default solvent dependent settings implemented in Gaussian is used.

The path to the Gaussian binary must be set via the environment variable `GAUSSIAN_BINARY_PATH`.

You can specify the following settings in the settings block:

- `method`: This specifies the calculation method to be used. Note that the name of the method should match the string used in a typical Gaussian input file. By default, it is `PBEPBE`.
- `basis_set`: This specifies the basis set string. By default, it is `def2SVP`. You can specify any valid Gaussian basis set string (see the Gaussian manual for a complete list).
- `solvation`: This specifies the solvation model to be used. By default, it is empty. Available settings are `cpcm`, `pcm`, `dipole`, `ipcm`, `scipcm`, and `smd`.
- `solvent`: This specifies the solvent for implicit solvation. By default, it is empty. You can specify any valid Gaussian solvent name.
- `external_program_memory`: The total amount of memory in MB that should be available for Gaussian to use. By default set to `1024`.
- `gaussian_filename_base`: This specifies the basic filename (prefix) used for all files related to the Gaussian calculations. By default, it is set to "gaussian_calc"; therefore, the generated input file will be named "gaussian_calc.inp".
- `scf_guess` The guess for the SCF. Available options are `read`, `harris`, `hucke`, `core`, `only` and `(only, read)`. Per default, it is set to `read`, reading in a checkpoint file from a previous calculation is read in, if present.

[26] Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, *J. Chem. Phys.* **2010,** *132,* 154104

[27] Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the damping function in dispersion corrected density functional theory, *J. Comput. Chem.* **2011,** *32,* 1456–1465

[28] Barone, V.; Cossi, M. Quantum Calculation of Molecular Energies and Energy Gradients in Solution by a Conductor Solvent Model, *J. Phys. Chem. A* **1998,** *102,* 1995–2001

Serenity

In order to use Serenity with ReaDuct, specify `program: Serenity` in the respective system block and the desired calculation method family (e.g., `DFT`) in the `method_family` key. The method (e.g., `PBE`) can be specified in the system settings. Dispersion correction can be added to this method string, e.g: `PBE-D3BJ`.

Serenity is called via the SCINE Serenity wrapper. Therefore, you have to install this wrapper (refer to the README provided with the wrapper for detailed installation instructions).

You can specify the following settings in the settings block:

- `basis_set`: This specifies the basis set string. By default, it is `6-31GS`. You can specify any valid Serenity basis set string.
- `solvation`: This specifies the solvation model to be used. By default, it is empty. Available settings are `cpcm` and `iefpcm`.
- `solvent`: This specifies the solvent for implicit solvation. By default, it is empty. You can specify any valid Serenity solvent name.
- `show_serenity_output`: Turns on or off the text output by Serenity. By default set to `false`.
- `scf_initialguess`: The initial guess for creating molecular orbitals. Available options are `hcore` (take the solutions of the core Hamiltonian, *i.e.*, without electron–electron interactions, `eht` (use extended Hückel theory), `atom_dens` (combine atomic densities from a minimal basis), `atom_scf` (combine atomic densities from a def2-QZVP basis), `atom_scf_inplace` (create atomic densities on the fly from an atom-wise SCF), and `sap` (use a superposition of atomic potentials). The default is `atom_scf`.
- `basis_auxJLabel`: Basis set label for the auxiliary basis for Coulomb integrals (RI). By default set to `RI_J_Weigend`.
- `basis_auxCLabel`: Basis set label for the auxiliary basis for correlation treatments.
- `basis_makeSphericalBasis`: Whether to use a spherical basis. By default `true`. When set to `false`, a Cartesian basis is used.
- `basis_integralThreshold`: The threshold for prescreening in integral evaluations. The default values it is calculated as $1 \cdot 10^{-8}/(3M)$, where M is the number of Cartesian basis functions.

- `basis_basisLibPath`: The path to the basis set files. This value is inferred from the environment variable `SERENITY_RESOURCES`.
- `basis_firstECP`: The nuclear charge number of the first atom to receive effective core potentials. By default set to 37.
- `grid_gridType`: Procedure how to combine atomic grids. Available options are `Becke`[29], `SSF`[30], and `Voronoi`. The default is `SSF`.
- `grid_accuracy`: The accuracy of the integration grid. This can take on values between 1 and 7; by default, it is 5. Larger numbers correspond to a finer, more accurate grid.
- `grid_smallGridAccuracy`: The accuracy of the smaller integration grid used in temporary steps. This can take on values between 1 and 7; be default, it is 3. Larger numbers correspond to a finer, more accurate grid.
- `scf_seriesDampingInitialSteps`: The number of initial dampening steps. By default set to 5.
- `pcm_alpha`: The sharpness parameter for the molecular surface model function for Delley-type surfaces. Has to be at least zero; by default set to 50.
- `pcm_scaling`: If true, the atomic radii used for the PCM cavity construction are scaled by a factor of 1.2. By default set to false.
- `pcm_radiiType`: The set of atomic radii to be used in the PCM cavity construction. By default set to `uff`.

[29] Becke, A. D. A multicenter numerical integration scheme for polyatomic molecules, *J. Chem. Phys.* **1988,** *88,* 2547-2553

[30] Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. Achieving linear scaling in exchange–correlation density functional quadratures, *Chem. Phys. Lett.* **1996,** *257,* 213-223

XTB

In order to use XTB with READUCT, specify `program: XTB` in the respective system block and the desired calculation method family (e.g., `GFN2`) in the `method_family` key.

It can be downloaded and integrated into READUCT automatically at compilation time with the CMake option `-DBUILD_XTB=ON` (*cf.* also section Installation).

You can specify the following settings in the settings block:

- `solvation`: This specifies the solvation model to be used. By default, it is empty. To enable implicit solvation, specify `gbsa`.
- `solvent`: This specifies the solvent for implicit solvation. By default, it is empty. For GFN1 and GFN2, available options are `acetone`, `acetonitrile`, `benzene`, `ch2cl2`, `chcl3`, `cs2`, `dmso`, `ether`,

`methanol`, `toluene`, `thf`, `water`, `h2o`. For GFN-FF, they are `acetone`, `acetonitrile`, `benzene`, `ch2cl2`, `chcl3`, `cs2`, `dmf`, `dmso`, `ether`, `toluene`, `thf`, `water`, `h2o`

- `symmetry_number`: This specifies the symmetry number used thermochemical calculations. The default is 1.
- `print_level`: This specifies the verbosity of the output. Possible values are 0, 1, and 2. By default, it is set to 0 (least verbose).

Tasks

Single Point Calculation

The single point task can be used to obtain the electronic energy of a given system. In order to carry out this task, specify any of the following in the respective task block: `type: single_point`, `type: singlepoint`, `type: sp`, or `type: energy`. The single point task will print partial atomic charges if the given model provides any. If charges are not required the keyword `require_charges: false` can be given in the tasks settings. In this case the program will not abort if a method that does not provide charges is requested. Furthermore, the gradients with respect to the nuclear coordinates can be requested by setting the keyword `require_gradients: true`, bond orders can be requested by setting the keyword `require_bond_orders: true`, and the orbital energies can be requested by setting the keyword `orbital_energies: true`. Separated energy and gradient contributions in QM/MM can be requested with the keywords `require_partial_energies: true` and `require_partial_gradients: true`, respectively. This task also allows to carry out test calculations with other spin multiplicities to ensure that the ground state is calculated. This option can be activated by setting `spin_propensity_check` to an integer value, which specifies the range to check above and below the current spin multiplicity. For example, setting it to 1 will check a multiplicity of 1 and 5 for a system with a spin multiplicity of 3. This setting is defaulted to 0, which deactivates the check. Like in any task, the setting `stop_on_error` can be given to control whether the program throws an exception for a failed energy calculation or simply returns `false` for the task and proceeds with the remaining tasks. The default value is `true`. Like in any task, the setting `silent_stdout_calculator` can be given to control whether any standard output of the calculation itself should be printed. The default value is `true`.

Bond Order Analysis

This task has been deprecated; it will be removed in future versions. Please use the Single Point Task instead.

Depending on the chosen method it is possible to generate Mayer bond orders for a given system. In order to carry out this task, specify any of the following in the respective task block: `type: bond_orders`, `type: bondorders`, `type: bonds`, `type: bos`, or `type: bo`. This task also generates and states the electronic energy. Like in any task, the setting `stop_on_error` can be given to control whether the program throws an exception for a failed energy calculation or simply returns `false` for the task and proceeds with the remaining tasks. The default value is `true`. Like in any task, the setting `silent_stdout_calculator` can be given to control whether any standard output of the calculation itself should be printed. The default value is `true`.

Hessian Calculation

This task calculates the Hessian of a given system and outputs the vibrational frequencies as well as thermochemical data. In order to carry out this task, specify any of the following in the respective task block: `type: hessian`, `type: frequency_analysis`, `type: frequencyanalysis`, `type: frequencies`, `type: frequency`, or `type: freq`.

You can adjust the temperature for the calculation of thermochemical data with the keyword `temperature` in the system settings block. Also, when using Sparrow as the backend program, the molecular symmetry number σ has to be specified there if it differs from the default value of one. See the Sparrow manual for details. Like in any task, the setting `stop_on_error` can be given to control whether the program throws an exception for a failed energy calculation or simply returns `false` for the task and proceeds with the remaining tasks. The default value is `true`. Like in any task, the setting `silent_stdout_calculator` can be given to control whether any standard output of the calculation itself should be printed. The default value is `true`.

Structure Optimization

This task is used in order to optimize the structure of a given system to a minimum on the potential energy surface. In order to carry out

this task, specify any of the following in the respective task block: `type: geometry_optimization`, `type: geometryoptimization`, `type: geoopt`, or `type: opt`.

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `optimizer`: This sets the desired optimization algorithm. You can set `bfgs` for the BFGS algorithm including G-DIIS, `lbfgs` for the L-BFGS algorithm, `steepestdescent` or `sd` for a steepest descent algorithm, and `newtonraphson` or `nr` for a Newton–Raphson algorithm. By default, it is set to `bfgs`.
- `convergence_step_max_coefficient`: The convergence threshold for the maximum absolute element of the last step taken. By default set to `1.0e-4`.
- `convergence_step_rms`: The convergence threshold for the root mean square of the last step taken. By default set to `5.0e-4`.
- `convergence_gradient_max_coefficient`: The convergence threshold for the maximum absolute element of the gradient. By default set to `5.0e-5`.
- `convergence_gradient_rms`: The convergence threshold for the root mean square of the gradient. By default set to `1.0e-5`.
- `convergence_delta_value`: The convergence threshold for the change in the functional value. By default set to `1.0e-7`.
- `convergence_max_iterations`: The maximum number of iterations. By default set to `150`.
- `convergence_requirement`: The number of criteria that have to converge besides the value criterion. This has to be between `0` and `4`; by default it is set to `3`.
- `geoopt_coordinate_system`: Select the representation of the coordinate system, either `internal` [31], `cartesianWithoutRotTrans`, or `cartesian`.
- `geoopt_constrained_atoms`: A list of atoms for which the Cartesian coordinates are constrained during the structure optimization. It is given as a list containing the corresponding atom indices (e.g., `[0, 12, 32, 42]`). This setting can only be set for a true Cartesian coordinate system. By default, this list is empty.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply

[31] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. The default value is `true`.

- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

- `unitcelloptimizer`: This sets the desired optimization algorithm for the unit cell. You can set `bfgs` for the BFGS algorithm including G-DIIS, `lbfgs` for the L-BFGS algorithm, and `steepestdescent` or `sd` for a steepest descent algorithm. By default, it is deactivated. A unit cell optimization requires a program that support the calculation of a stress tensor.

If you specified a `unitcelloptimizer`, you can also set the following options:

- `cellopt_optimize_angles`: Specify whether the angles of the unit cell shall be fixed (true) or constrained (false). By default set to `true`.

- `cellopt_optimize_a`: Specify whether the *a* vector of the unit cell (defined along the positive x-axis) shall be optimized. By default set to `true`.

- `cellopt_optimize_b`: Specify whether the *b* vector of the unit cell (defined x-y-plane) shall be optimized. By default set to `true`.

- `cellopt_optimize_c`: Specify whether the *c* vector of the unit cell (defined in the positive z-direction) shall be optimized. By default set to `true`.

- `cellopt_geoopt_max_convergence_iterations`: Define the maximum number of microiterations in the optimization of the geometry. By default set to `100`.

- `cellopt_cellopt_max_convergence_iterations`: Define the maximum number of microiterations in the optimization of the unitcell. By default set to `100`. The total number of iterations is still controlled by `geoopt_max_convergence_iterations`.

If you specified `optimizer: bfgs`, the default coordinate system is `internal` and you can also set the following options:

- `bfgs_use_gdiis`: Switch to enable the use of a G-DIIS possibly accelerating convergence. By default set to `true`.

- `bfgs_gdiis_max_store`: The maximum number of old steps used in the G-DIIS. By default set to `5`.
- `bfgs_use_trust_radius`: Whether to use the trust radius. By default set to `true`.
- `bfgs_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.3`.
- `bfgs_min_iterations`: The minimum number of cycles before the convergence criteria are checked. By default set to `1`.
  Setting this option to values larger than `1` enforces one or multiple updates of the B matrix. This might help for structures with small absolute values of imaginary frequencies.

If you specified `optimizer: lbfgs`, the default coordinate system is `cartesianWithoutRotTrans` and you can also set the following options:

- `lbfgs_maxm`: The number of parameters and gradients from previous iterations to keep. By default set to `10`.
- `lbfgs_linesearch`: Whether to use a line search or not. By default set to `true`.
- `lbfgs_c1`: The first parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.0001`.
- `lbfgs_c2`: The second parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.9`.
- `lbfgs_step_length`: The initial step length. By default set to `1.0`.
- `lbfgs_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `lbfgs_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.

If you specified `optimizer: steepestdescent` or `optimizer: sd`, the default coordinate system is `internal` and you can also set the following options:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `0.1`.
- `sd_use_trust_radius`: Whether to use the trust radius. By default set to `false`.

- `sd_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.
- `sd_dynamic_multiplier`: A factor by which the `sd_factor` is multiplied each step. By default set to `1.0`. If it is set to values larger than `1.0`, `sd_use_trust_radius` must be set to `true`.

If you specified `optimizer: newtonraphson` or `optimizer: nr`, the default coordinate system is `cartesianWithoutRotTrans` and you can also set the following options:

- `nr_trust_radius`: The trust radius (maximum movement in any cartesian direction by any atom) of a taken step. By default set to `0.5`.
- `nr_svd_threshold`: The threshold for the singular value decomposition of the Hessian. By default set to `1.0e-12`.

Transition State Optimization

This task is used to optimize the structure of a given system to a transition state on the potential energy surface. In order to carry out this task, specify any of the following in the respective task block: `type: transition_state_optimization`, `type: transitionstate_optimization`, `type: tsopt`, or `type: ts`.

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `optimizer`: This sets the desired optimization algorithm. You can set `bofill` for Bofill's algorithm[32], or any of `eigenvector_following`, `eigenvectorfollowing`, `ef`, `evf`, or `ev` for a eigenvector following algorithm, or `dimer` for the Dimer algorithm [33]. By default, it is set to `bofill`.
- `automatic_mode_selection`: Specify the indices of atoms that are essential for the reaction in a list to automatically select the imaginary frequency mode to follow. Out of all modes in mass-weighted coordinates, the maximum mode score considering the contributions by movement of the selected atoms (default: `0.5`) as well as the imaginary frequency of the mode (default: `0.5`) determines which mode is selected.
- `automatic_mode_selection_wavenumber_weight`: The weight of the imaginary frequency to determine the mode score in the automatic mode selection. By default set to `0.5`.

[32] Bofill, J. M. Updated Hessian Matrix and the Restricted Step Method for Locating Transition Structures, *J. Comput. Chem.* **1994,** *15,* 1-11; and Farkas, O.; Schlegel, H. B. Methods for optimizing large molecules, *Phys. Chem. Chem. Phys.* **2002,** *4,* 11–15

[33] Henkelman, G.; Jónsson, H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, *J. Chem. Phys.* **1999,** *111,* 7010–7022; Kästner, J.; Sherwood, P. Superlinearly converging dimer method for transition state search, *J. Chem. Phys* **2008,** *128,* 014106; and Shang, C.; Liu, Z.-P. Constrained Broyden minimization combined with the dimer method for locating transition state of complex reactions, *J. Chem. Theory Comput.* **2010,** *6,* 1136–1144

- `automatic_mode_selection_contribution_weight`: The weight of the contribution in terms of movement of the selected atoms to determine the mode score in the automatic mode selection. By default set to `0.5`.
- `convergence_step_max_coefficient`: The convergence threshold for the maximum absolute element of the last step taken. By default set to `1.0e-4`.
- `convergence_step_rms`: The convergence threshold for the root mean square of the last step taken. By default set to `5.0e-4`.
- `convergence_gradient_max_coefficient`: The convergence threshold for the maximum absolute element of the gradient. By default set to `5.0e-5`.
- `convergence_gradient_rms`: The convergence threshold for the root mean square of the gradient. By default set to `1.0e-5`.
- `convergence_delta_value`: The convergence threshold for the change in the functional value. By default set to `1.0e-7`.
- `convergence_max_iterations`: The maximum number of iterations. By default set to `150`.
- `convergence_requirement`: The number of criteria that have to converge besides the value criterion (`convergence_delta_value`). This has to be between `0` and `4`; by default it is set to `3`.
- `geoopt_coordinate_system`: Select the representation of the coordinate system, either `internal` [34], `cartesianWithoutRotTrans`, or `cartesian`.
- `geoopt_constrained_atoms`: A list of atoms for which the Cartesian coordinates are constrained during the structure optimization. It is given as a list containing the corresponding atom indices (e.g., `[0, 12, 32, 42]`). This setting can only be set for a true Cartesian coordinate system. By default, this list is empty.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. The default value is `true`.
- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

[34] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

- `write_selected_mode`: Determine wheter the selected mode is written to a trajectory file. The default value is `false`. This option is not available for the `dimer` optimizer.

If you specified `optimizer: bofill`, the default coordinate system is `cartesianWithoutRotTrans` and you can also set the following options:

- `bofill_trust_radius`: The maximum movement in any Cartesian direction by any atom. By default set to `0.1`.
- `bofill_hessian_update`: The number of iterations using the Bofill update scheme in between full hessian calculations. By default set to `5`.
- `bofill_follow_mode`: The number of the eigenvector to follow starting from `0` with `0` being the eigenvector with the lowest eigenvalue. By default set to `0`. Please be aware that the algorithm does not always maximize the n-th mode, but the mode with the highest overlap to the mode followed in the previous step. Therefore, any changes in the ranking of normal modes during the optimization are not affecting the optimization.

If the optimization does not converge and the single optimization steps are often limited by the trust radius, it is recommended to try another optimization with a larger trust radius.

If you specified `optimizer: eigenvector_following`, `eigenvectorfollowing`, `ef`, `evf`, or `ev`, the default coordinate system is `cartesianWithoutRotTrans` and you can also set the following options:

- `ev_trust_radius`: The maximum movement in any Cartesian direction by any atom. By default set to `0.1`.
- `ev_follow_mode`: The number of the eigenvector to follow starting from `0` with `0` being the eigenvector with the lowest eigenvalue. By default set to `0`. Please be aware that the algorithm does not always maximize the n-th mode, but the mode with the highest overlap to the mode followed in the previous step. Therefore, any changes in the ranking of normal modes during the optimization are not affecting the optimization.

If the optimization does not converge and the single optimization steps are often limited by the trust radius, it is recommended to try another optimization with a larger trust radius.

If you specified `optimizer: dimer`, the default coordinate system is `cartesianWithoutRotTrans` and you can also set the following options:

- Options to initialize the dimer:
  - `dimer_calculate_hessian_once`: Calculate the Hessian matrix in the beginning to create the dimer along the lowest frequency eigenvector and skip the first rotation. By default set to `false`. This option saves about 30 – 60 gradient calculations, so if the calculation of the Hessian matrix of your system is faster than these gradient calculations and you do not have a guess vector for the dimer at hand, set this to true.
  - `dimer_guess_vector_file`: File name in which a row vector is stored. It will be read and used as a guess for the dimer axis. By default the dimer is still rotated before the first translation. The B-Spline task allows to write out the tangent at the maximum of the spline for this purpose (see section).
  - `dimer_discrete_guesses`: Calculate the vector between two structures given as XYZ files to create the dimer. A list of file names of structures has to be provided. This vector then forms the dimer axis. By default the dimer is rotated before the first translation.

    If none of these three options is given, a random vector is used for the initialization of the dimer. If more than one of these three options is set, only one will be used, because there can only be one dimer axis. The options will be preferred according to the above list from top to bottom. If a guess is provided the dimer will be constructed towards the point higher in energy than the input structure. If the energy decreases in both directions of the input vector, the direction with the lower curvature is preferred. Please be aware that these criteria can lead to a direction inverse to the input vector.
- `dimer_follow_mode`: The number of the eigenvector to follow starting from `0` with `0` being the eigenvector with the lowest eigenvalue. If this option is set, the option `dimer_calculate_hessian_once` is automatically set to `true` and therefore all other options to initialize the dimer are overruled. If simply the option `dimer_calculate_hessian_once` was set to true, this option is set to `0` by default.
- `dimer_skip_first_rotation`: Skip the first rotation. Recommended if a very reliable guess vector is available. By default set to `false`. If the option to calculate the Hessian matrix for the first step was set to `true`, this option is automatically set to `true`.
- `dimer_only_one_rotation`: Only rotate the dimer in the first step. By default set to `false`.

- `dimer_rotation_lbfgs`: Option to use L-BFGS in the rotation. By default set to `true` and automatically set to `false` if `dimer_rotation_conjugate_gradient` is set. If both are set to false, a steepest descent is performed.
- `dimer_rotation_conjugate_gradient`: Option to use conjugate gradient method in the rotation. By default set to `false`.
- `dimer_lbfgs_memory`: Number of saved rotation steps in L-BFGS. By default set to `5`.
- `dimer_translation`: Specify algorithm for the translation step of the dimer as a string. The following options are available:
  - `bfgs`: Option to use BFGS in the translation (default).
  - `linesearch`: Option to use a stepsize scaling based on the change of projection of the modified force onto the dimer axis [35].
  - `amsgrad`: Option to use AMSGRAD [36] in the translation.
- `dimer_grad_rmsd_threshold`: Threshold for applying stepsize scaling or stepvector modification. By default set to `1.0e-3`.
- `dimer_minimization_cycle`: Cycle after which all modes but the dimer mode are minimized even if the curvature estimation is positive. Before this cycle only the dimer mode is maximized and all others are kept unchanged if the curvature estimation is positive.
- `dimer_bfgs_start`: Cycle in which the BFGS is activated. By default set to `16`. The combination of the three just mentioned parameters showed the best results in the Baker-Chan set [37].
- `dimer_trust_radius`: The maximum movement in any Cartesian direction by any atom. By default set to `0.2`. If you notice that this trust radius is often reached, it is recommended to increase it, which might lead to better convergence.
- `dimer_multi_scale`: Option to apply step size scaling onto the scaled step of the previous step. By default set to `true`. This affects only the `linesearch` algorithm.
- `dimer_default_translation_step`: Scaling factor for steepest descent translation. By default set to `1.0`.
- `dimer_radius`: Radius of the dimer. By default set to `0.01`.
- `dimer_gradient_interpolation`: Option to estimate the gradient during the rotation [38]. By default set to `true`.

[35] Kästner, J.; Sherwood, P. Superlinearly converging dimer method for transition state search, *J. Chem. Phys* **2008,** *128,* 014106

[36] Reddi, S. J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond, **2019,** arXiv:1904.09237 [cs.LG]

[37] Baker, J.; Chan, F. The Location of Transition States: A Comparison of Cartesian, Z-Matrix, and Natural Internal Coordinates, *J. Comput. Chem.* **1996,** *17,* 888–904

[38] Kästner, J.; Sherwood, P. Superlinearly converging dimer method for transition state search, *J. Chem. Phys* **2008,** *128,* 014106

- `dimer_max_rotations_first_cycle`: Maximum number of allowed individual rotations in the first rotation. By default set to `100`.
- `dimer_max_rotations_other_cycle`: Maximum number of allowed individual rotations in all rotations except the first. By default set to `100`.
- `dimer_phi_tolerance`: Threshold for convergence of rotation with ϕ method [39]. By default set to `1.0e-3`.
- `dimer_rotation_gradient_first`: Threshold for convergence of rotation in the first cycle. By default set to `1.0e-7`.
- `dimer_rotation_gradient_other`: Threshold for convergence of rotation in all cycles but the first. By default set to `1.0e-4`.
- `dimer_interval_of_rotations`: Number of translation steps after which it is checked whether a rotation should be performed. By default set to `5`.
- `dimer_decrease_rotation_gradient_threshold`: Option to decrease the threshold for the gradient in the rotation after a certain number of cycles. By default set to `false`.
- `dimer_lowered_rotation_gradient`: Threshold for convergence of rotation, if lowered by `decrease_rotation_gradient_threshold`. By default set to `1.0e-3`.
- `dimer_cycle_of_rotation_gradient_decrease`: Number of rotation cycles after which the rotation gradient threshold is decreased. By default set to `5`.

[39] Kästner, J.; Sherwood, P. Superlinearly converging dimer method for transition state search, *J. Chem. Phys* **2008,** *128,* 014106

At the moment, it is recommended to perform the optimization with the `Dimer` optimizer in Cartesian coordinates, which is the default. All of the above options set to their default value should be suitable for most applications. If you encounter a convergence issue, try the projection linesearch for the translation. This will increase the number of needed steps, but might converge. However, we did not observe general better convergence with this option. It might succeed in cases where the BFGS fails though. If you notice that the trust radius is hit in multiple optimization steps, try to increase it.

Intrinsic Reaction Coordinate Calculation

This task is used to perform an intrinsic reaction coordinate (IRC) calculation using mass-weighted gradients. In order to carry out this task, specify any of the following in the respective task block: `type: ircopt`, or `type: irc`. Note that for this task you have to specify two

output systems. The first one will contain the results of the forward IRC calculation while the second one will contain the result of the backward IRC calculation.

You usually want to set the following setting:

- `irc_mode`: This sets the normal mode which should be used for the IRC calculation. By default set to `0` (designates the first normal mode).

The task works without the specification of any additional settings. However, if desired, the following settings can always be set:

- `optimizer`: This sets the desired optimization algorithm. You can set `bfgs` for the BFGS algorithm including G-DIIS, `lbfgs` for the L-BFGS algorithm, and `steepestdescent` or `sd` for a steepest descent algorithm. By default, it is set to `sd`.
- `convergence_step_max_coefficient`: The convergence threshold for the maximum absolute element of the last step taken. By default set to `1.0e-4`.
- `convergence_step_rms`: The convergence threshold for the root mean square of the last step taken. By default set to `5.0e-4`.
- `convergence_gradient_max_coefficient`: The convergence threshold for the maximum absolute element of the gradient. By default set to `5.0e-5`.
- `convergence_gradient_rms`: The convergence threshold for the root mean square of the gradient. By default set to `1.0e-5`.
- `convergence_delta_value`: The convergence threshold for the change in the functional value. By default set to `1.0e-7`.
- `convergence_max_iterations`: The maximum number of iterations. By default set to `150`.
- `convergence_requirement`: The number of criteria that have to converge besides the value criterion. This must be between `0` and `4`; by default it is set to `3`.
- `irc_coordinate_system`: Select the representation of the coordinate system, either `internal` [40], `cartesianWithoutRotTrans`, or `cartesian`. The default for the IRC task is `cartesianWithoutRotTrans`.
- `irc_initial_step_size`: Maximum displacement of one coordinate of one atom along the given mode. All other coordinates are scaled down accordingly; by default it is set to `0.3`.

[40] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. Especially, when using a SD type optimizer this option can be helpful. The default value is `true`.
- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

If you specified `optimizer: bfgs`, you can also set the following options:

- `bfgs_use_gdiis`: Switch to enable the use of a G-DIIS possibly accelerating convergence. By default set to `true`.
- `bfgs_gdiis_max_store`: The maximum number of old steps used in the G-DIIS. By default set to `5`.
- `bfgs_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `bfgs_trust_radius`: The maximum size of a taken step. By default set to `0.1`.

If you specified `optimizer: lbfgs`, you can also set the following options:

- `lbfgs_maxm`: The number of parameters and gradients from previous iterations to keep. By default set to `10`.
- `lbfgs_linesearch`: Whether to use a line search or not. By default set to `true`.
- `lbfgs_c1`: The first parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.0001`.
- `lbfgs_c2`: The second parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.9`.
- `lbfgs_step_length`: The initial step length. By default set to `1.0`.
- `lbfgs_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `lbfgs_trust_radius`: The maximum size of a taken step. By default set to `0.1`.

If you specified `optimizer: steepestdescent` or `optimizer: sd`, you can also set the following options:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `2.0`.
- `sd_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `sd_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.
- `sd_dynamic_multiplier`: A factor by which the `sd_factor` is multiplied each step. By default set to `1.0`. If it is set to values larger than `1.0`, `sd_use_trust_radius` must be set to `true`.

Artificial Force Induced Reaction Calculation

This task is used in order to do an artificial force induced reaction (AFIR[41]) calculation. In order to carry out this task, specify any of the following in the respective task block: `type: afir_optimization`, `type: afiroptimization`, `type: afiropt`, or `type: afir`. The energy given in the output includes the artifical force term.

[41] Maeda, S.; Morokuma, K. Communications: A systematic method for locating transition structures of A+B → X type reactions, *J. Chem. Phys.* **2010,** *132*, 241102; and Maeda, S.; Morokuma, K. Finding Reaction Pathways of Type A + B → X: Toward Systematic Prediction of Reaction Mechanisms, *J. Chem. Theory Comput.* **2011,** *7*, 2335–2345

You usually want to set the following settings:

- `afir_rhs_list`: This specifies list of indices of atoms to be artificially forced onto or away from those in the LHS list (see below). By default, this list is empty. Note that the first atom has the index zero.
- `afir_lhs_list`: This specifies list of indices of atoms to be artificially forced onto or away from those in the RHS list (see above). By default, this list is empty. Note that the first atom has the index zero.

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `afir_weak_forces`: This activates an additional, weakly attractive force applied to all atom pairs. By default set to `false`.
- `afir_attractive`: Specifies whether the artificial force is attractive or repulsive. By default set to `true`, which means that the force is attractive.
- `afir_energy_allowance`: The maximum amount of energy to be added by the artifical force, in kJ/mol. By default set to `1000.0`.

- `afir_phase_in`: The number of steps over which the full attractive force is gradually applied. By default set to `30`.
- `afir_coordinate_system`: Select the representation of the coordinate system, either `internal` [42], `cartesianWithoutRotTrans`, or `cartesian`. The default for the AFIR task is `cartesianWithoutRotTrans`.
- `afir_use_max_fragment_distance`: Whether to stop the AFIR optimization signalling convergence after the shortest distance between the atoms specified in the LHS list and those in the RHS list exceeds a threshold value. This option can be useful when performing a repulsive AFIR scan resulting into a molecule's dissociation. By default set to `false`.
- `afir_max_fragment_distance`: The distance threshold to be used when the `afir_use_max_fragment_distance` setting is enabled in atomic units. By default set to `8.0`.
- `optimizer`: This sets the desired optimization algorithm. You can set `bfgs` for the BFGS algorithm including G-DIIS, `lbfgs` for the L-BFGS algorithm, and `steepestdescent` or `sd` for a steepest descent algorithm. By default, it is set to `bfgs`.
- `convergence_step_max_coefficient`: The convergence threshold for the maximum absolute element of the last step taken. By default set to `1.0e-4`.
- `convergence_step_rms`: The convergence threshold for the root mean square of the last step taken. By default set to `5.0e-4`.
- `convergence_gradient_max_coefficient`: The convergence threshold for the maximum absolute element of the gradient. By default set to `5.0e-5`.
- `convergence_gradient_rms`: The convergence threshold for the root mean square of the gradient. By default set to `1.0e-5`.
- `convergence_delta_value`: The convergence threshold for the change in the functional value. By default set to `1.0e-7`.
- `convergence_max_iterations`: The maximum number of iterations. By default set to `150`.
- `convergence_requirement`: The number of criteria that have to converge besides the value criterion. This has to be between `0` and `4`; by default it is set to `3`.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining

[42] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. Especially, when using a SD type optimizer this option can be helpful. The default value is `true`.

- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

If you specified `optimizer: bfgs`, you can also set the following options:

- `bfgs_use_gdiis`: Switch to enable the use of a G-DIIS possibly accelerating convergence. By default set to `true`.
- `bfgs_gdiis_max_store`: The maximum number of old steps used in the G-DIIS. By default set to `5`.
- `bfgs_use_trust_radius`: Whether to use the trust radius. By default set to `true`.
- `bfgs_trust_radius`: The maximum size of a taken step. By default set to `0.1`.

If you specified `optimizer: lbfgs`, you can also set the following options:

- `lbfgs_maxm`: The number of parameters and gradients from previous iterations to keep. By default set to `10`.
- `lbfgs_linesearch`: Whether to use a line search or not. By default set to `true`.
- `lbfgs_c1`: The first parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.0001`.
- `lbfgs_c2`: The second parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.9`.
- `lbfgs_step_length`: The initial step length. By default set to `1.0`.
- `lbfgs_use_trust_radius`: Whether to use the trust radius. By default set to `true`.
- `lbfgs_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.

If you specified `optimizer: steepestdescent` or `optimizer: sd`, you can also set the following options:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `0.1`.
- `sd_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `sd_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.
- `sd_dynamic_multiplier`: A factor by which the `sd_factor` is multiplied each step. By default set to `1.0`. If it is set to values larger than `1.0`, `sd_use_trust_radius` must be set to `true`.

B-Spline Interpolation and Optimization

This task is used in order to approximate a reaction path between a given start and end structure by means of an interpolation based on B-splines[43]. This interpolated path can be optimized to yield a better approximation to the true reaction path. Furthermore, from the optimized path, a guess for the transition state structure can be extracted. In order to carry out this task, specify any of the following in the respective task block: `type: bspline_interpolation`, `type: bsplineinterpolation`, or `type: bspline`.

[43] Vaucher, A. C.; Reiher, M. Minimum Energy Paths and Transition States by Curve Optimization, *J. Chem. Theory Comput.* **2018,** *14,* 3091–3099

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `optimizer`: This sets the desired optimization algorithm. You can set `lbfgs` for the L-BFGS algorithm, and `steepestdescent` or `sd` for a steepest descent algorithm. By default, it is set to `lbfgs`.
- `optimize`: Whether the interpolated path should be optimized. By default set to `true`.
- `trajectory_guess`: A list of possibly concatenated XYZ files that should be added as data points when interpolating the initial spline. The files may, but do not need to, include the start and end structures as first and last structure.
- `extract_ts_guess`: Whether a guess for the transition state structure should be extracted from the optimized path. By default set to `true`. If set to true, the structure is written into the file `<output_name>_tsguess.xyz`.
- `extract_threshold`: Specifies the threshold for the extraction of the maximum energy structure from a reaction profile. For the extraction, the part of the spline around its maximum is discretized

to five grid points. These points are refined until the energy difference between the two points neighboring the point with maximal energy is less than two times this threshold. By default this threshold is set to `1e-3` (hartree).

- `extract_ts_guess_neighbours`: Whether the structures before and after the transition state structure should be extracted from the optimized path. By default set to `false`. If set to true, the structures are written into the file `<output_name>_tsguess-1.xyz` and `<output_name>_tsguess+1.xyz`.
- `tangent_file`: The name of the file in which the tangent of the spline at the TS guess will be stored as a row vector. This can be used for a single ended TS optimization. If a relative path is given, it is interpreted relative to the output directory of the B-spline interpolation task. By default no tangent is written out.
- `align_structures`: Whether to remove the overall rotation and translation of the end structure. By default set to `true`.
- `num_control_points`: The number of control points for the B-spline representing the reaction path. This number is directly proportional to the number of parameters to optimize. By default set to `5`.
- `num_integration_points`: The number of integration points used during the optimization of the B-spline. A higher number of integration points increases the accuracy but also the computational cost. By default set to `21`.
- `num_structures`: Sets the number of structures into which a reaction path is discretized for the final output (*i.e.,* when writing it to a XYZ trajectory file). By default set to `10`.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. Especially, when using a SD type optimizer this option can be helpful. The default value is `true`.
- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

If you specified `optimizer: lbfgs`, you can also set the following options:

- `lbfgs_maxm`: The number of parameters and gradients from previous iterations to keep. By default set to `10`.
- `lbfgs_linesearch`: Whether to use a line search or not. By default set to `false`.
- `lbfgs_c1`: The first parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.0001`.
- `lbfgs_c2`: The second parameter of the Wolfe conditions. This option is only relevant if line search is used (see above). By default set to `0.9`.
- `lbfgs_step_length`: The initial step length. By default set to `1.0`.
- `lbfgs_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `lbfgs_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.

If you specified `optimizer: steepestdescent` or `optimizer: sd`, you can also set the following options:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `0.1`.
- `sd_use_trust_radius`: Whether to use the trust radius. By default set to `false`.
- `sd_trust_radius`: The maximum movement in any cartesian direction by any atom. By default set to `0.1`.
- `sd_dynamic_multiplier`: A factor by which the `sd_factor` is multiplied each step. By default set to `1.0`. If it is set to values larger than `1.0`, `sd_use_trust_radius` must be set to `true`.

Newton Trajectory Calculations

These tasks are used to generate transition state guesses. To this end, sets of atoms are force onto one another, or pushed apart. There are currently two algorithms available for these types of calculations: Newton Trajectory 1 (`type: nt`, `type: nt1`) and Newton Trajectory 2 (`type: nt2`).

Newton Trajectory Algorithm 1

This task is used to generate transition state guesses. Two groups of atoms are forced onto one another. In order to carry out this task, specify any of the following in the respective task block: `type: newtontrajectory`, `type: ntoptimization`, `type: ntopt`, `type: nt` or `type: nt1`.

In this version of the algorithm two groups of atoms (LHS and RHS) are pused together or pulled appart. You usually want to set the following settings:

- `nt_rhs_list`: This specifies list of indices of atoms to be forced onto or away from those in the LHS list (see below). By default, this list is empty. Note that the first atom has the index zero.
- `nt_lhs_list`: This specifies list of indices of atoms to be forced onto or away from those in the RHS list (see above). By default, this list is empty. Note that the first atom has the index zero.

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `nt_attractive`: Specifies whether the applied force is attractive or repulsive. By default set to `true`, which means that the force is attractive.
- `nt_movable_side`: Specifies which side shall be moved with the applied force. You can choose:
  - both (default)
  - lhs
  - rhs
- `nt_total_force_norm`: The norm of the force applied onto each of the atoms to be moved along the geometric center of the LHS atoms and the center of the RHS atoms. If both sides shall be moved with the applied force, each side is made subject to half of this value. By default it is set to `0.1`.
- `nt_extraction_criterion`: Specify the criterion for the extracted guess; available options are:
  - `highest_maximum` Pick the highest local maximum along the trajectory (default).

  - `first_maximum` Pick the first local maximum along the trajectory.

- `nt_coordinate_system`: Select the representation of the coordinate system, either `internal` [44], `cartesianWithoutRotTrans`, or `cartesian`. The default is `cartesianWithoutRotTrans`. Please be aware, that this only affects the SD steps, while the BFGS microcycles will always be performed with true Cartesian coordinates due to constraints.

[44] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

- `nt_constrained_atoms`: A list of atoms for which the Cartesian coordinates are constrained during the optimization. It is given as a list containing the corresponding atom indices (e.g., `[0, 12, 32, 42]`). This setting can only be set for a true Cartesian coordinate system.

- `nt_use_micro_cycles`: Use micro cycles inbetween forced steps that move the constrained atoms. In these micro cycles a BFGS/GDIIS will be used to optimize the geometry with the lhs/rhs atoms fixed in place. By default it is set to `true`.

- `nt_fixed_number_of_micro_cycles`: Use a fixed number of micro cycles per macro cycle. If set to `false`, the number of micro cycles will grow with the number of macro iterations. It will grow by one micro cycle per macro iteration performed until `nt_number_of_micro_cycles` is reached. By default it is set to `true`.

- `nt_number_of_micro_cycles`: The maximum number of micro cycles used. By default it is set to `10`.

- `nt_filter_passes`: This task uses a Savitzky–Golay filter before analyzing the reaction curve. The number of passes through this filter is adjusted by the `nt_filter_passes` option. By default the number of passes is `10`.

- `convergence_max_iterations`: The maximum number of Newton trajectory macro iterations. By default set to `500`.

- `convergence_attractive_stop`: The distance between the forced atoms at which the algorithm stops and tries to identify a transition state guess. The given value is applied as a factor to the covalent radii of the compared atoms. For groups of multiple atoms in each list it is applied without covalent radii scaling to the geometric centers of the atom groups. In the case of an attractive force, this criterion set to 1.0 would stop the procedure as soon as two atoms defined in the lists are closer than 1.0 times the sum of

their covalent radii, or if the geometric centers are closer than 1.0 a.u. By default the value is set to `0.9`.

- `convergence_repulsive_stop`: The distance between the forced atoms at which the algorithm stops and tries to identify a transition state guess. The given value is applied as a factor to the covalent radii of the compared atoms. For groups of multiple atoms in each list it is applied without covalent radii scaling to the geometric centers of the atom groups. The repulsive procedure would stops if all atom pairs across the lists were further apart than the set value times the respective covalent radii sums, or if the geometric centers of the two groups are further than the value in a.u. apart. By default the value is set to `4.0`.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. Especially, when using a SD type optimizer this option can be helpful. The default value is `true`.
- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

The Newton trajectory task uses a steepest descent algorithm internally; for this reason the following setting is available:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `1.0`.

Newton Trajectory Algorithm 2

In order to carry out this task, specify any of the following in the respective task block: `type: newtontrajectory2`, `type: ntoptimization2`, `type: ntopt2`, or `type: nt2`.

In this version of the task any number of pairs of atoms are pushed together or pulled apart. You usually want to set the following settings:

- `nt_associations`: This specifies list of indices of atoms pairs to be forced onto another. Pairs are given in a flat list such that indices $2i$ and $2i + 1$ form a pair.
- `nt_dissociations`: This specifies list of indices of atoms pairs to

be forced away from another. Pairs are given in a flat list such that indices $2i$ and $2i + 1$ form a pair.

The task works without the specification of any additional settings; the default settings work usually fine. However, if desired, the following settings can always be set:

- `nt_total_force_norm`: The maximum norm of the force applied to an atom listed in any of the pairs. Note that some atoms may have forces applied that are less than the given value, all forces applied will be scaled such that the "fastest" atom will be receiving at maximum a force with the norm given in this setting. By default it is set to `0.1`.
- `nt_extraction_criterion`: Specify the criterion for the extracted guess; available options are:
  - `last_maximum_before_first_target` If an NT2 stop criterion is fulfilled in step n of a trajectory, pick the last local maximum in the first n steps of the trajectory; if no such maximum is present, pick the first local maximum after step n of the trajectory. If not NT2 stop criterion is fulfilled, pick the largest local maximum of the entire trajectory (default).
  - `highest_maximum` Pick the highest local maximum along the trajectory.
  - `first_maximum` Pick the first local maximum along the trajectory.
- `nt_coordinate_system`: Select the representation of the coordinate system, either `internal`[45], `cartesianWithoutRotTrans`, or `cartesian`. Please be aware that this only affects the SD steps, while the BFGS micro cycles will always be performed with true Cartesian coordinates due to constraints.
- `nt_constrained_atoms`: A list of atoms for which the Cartesian coordinates are constrained during the optimization. It is given as a list containing the corresponding atom indices (e.g., `[0, 12, 32, 42]`). This setting can only be set for a true Cartesian coordinate system.
- `nt_use_micro_cycles`: Use micro cycles inbetween forced steps that move the constrained atoms. In these micro cycles a BFGS/GDIIS will be used to optimize the geometry with the lhs/rhs atoms fixed in place. By default it is set to `true`.
- `nt_fixed_number_of_micro_cycles`: Use a fixed number of micro cycles per macro cycle. If set to `false`, the number of micro

[45] Meli, R. "IRC", https://github.com/RMeli/irc, 2019

cycles will grow with the number of macro iterations. It will grow by one micro cycle per macro iteration performed until `nt_number_of_micro_cycles` is reached. By default it is set to `true`.

- `nt_number_of_micro_cycles`: The maximum number of micro cycles used. By default it is set to `10`.
- `nt_filter_passes`: This task uses a Savitzky–Golay filter before analyzing the reaction curve. The number of passes through this filter is adjusted by the `nt_filter_passes` option. By default the number of passes is `10`.
- `convergence_max_iterations`: The maximum number of Newton trajectory macro iterations. By default set to `500`.
- `convergence_attractive_stop`: The distance between the forced atoms at which the algorithm stops and tries to identify a transition state guess. The given value is applied as a factor to the covalent radii of the compared atoms. In the case of an attractive force between two atoms, this criterion set to 1.0 would stop the NT procedure as soon as the two atoms are closer than 1.0 times the sum of their covalent radii. By default the value it is set to `0.9`.
- `stop_on_error`: Determine whether the program throws an exception for failed energy calculations or optimizations or simply returns `false` for the failed task and proceeds with the remaining tasks. In case of optimizations, the resulting structure after the maximum number of optimization steps has been reached, is still safed, when this is set to `false`. Especially, when using a SD type optimizer this option can be helpful. The default value is `true`.
- `silent_stdout_calculator`: Determine whether any standard output of the calculation itself should be printed. The default value is `true`.

The newton trajectory task uses a steepest descent algorithm internally; for this reason the following setting is available:

- `sd_factor`: The scaling factor to be used in the steepest descent algorithm. By default set to `1.0`.

Integral Evaluation

In order to carry out this task, specify any of the following in the respective task block: `type: INTEGRALS`, `type: INTS`.

This tasks calculates integrals for the molecular system and writes them to a file. By default the task writes the one-electron integrals to file. The following settings are supported:

- `require_one_electron_integrals`: If true, the one-electron integrals are written to the file `<system-name>.hcore.dat`. The default value is `true`.
- `silent_stdout_calculator`: If true, no standard output from the electronic structure program is printed. The default value is `true`.
- `stop_on_error`: If true, the program stops if the underlying electronic structure programs returns an error. The default value is `true`.

Task Chaining

You can specify multiple tasks to be executed after each other. Tasks are processed in the order in which they are given in the input file. For example, the following input file would first carry out a structure optimization, and then calculate the vibrational frequencies of the optimized structure:

```
systems:
  - name: water
    path: h2o.xyz
    program: Sparrow
    method_family: PM6

tasks:
  - type: geoopt
    input: [water]
    output: [water_opt]
  - type: hessian
    input: [water_opt]
```

Using the Python Library

ReaDuct provides Python bindings such that all functionality of ReaDuct can be accessed also via the Python programming language. In order to build the Python bindings, you need to specify `-DSCINE_BUILD_PYTHON_BINDINGS=ON` when running CMake (see also chapter Installation).

In order to use the Python bindings, you need to specify the path to the Python library in the environment variable `PYTHONPATH`, *e.g.*, you have to run the command

```
export PYTHONPATH=$PYTHONPATH:<source code directory>/install/lib64/python<version>/site-packages
```

where `<version>` is the Python version you are using (*e.g.*, 3.6). Now, you can simply import the library and use it as any other Python library. For example, in order to carry out a structure optimization, you could use the following Python script:

```
import scine_utilities as utils
import scine_readuct as readuct

water = utils.core.load_system_into_calculator('h2o.xyz', 'PM6', program='Sparrow',
                                               molecular_charge=0, spin_multiplicity=1)

systems = {}
systems['water'] = water

systems, success = readuct.run_opt_task(systems, ['water'], output=['water_opt'],
                                        optimizer='bfgs')

if success:
    print(systems['water_opt'].positions)
```

Note that we use a dictionary called "systems" to store all systems (Calculators) we deal with in one central data structure. As second argument, the structure optimization task accepts a list of the systems

which should be optimized, *i.e.*, the dictionary "systems" can contain more systems but these will not be optimized (all other tasks work with the same concept). The output system(s) will be automatically added to the systems dictionary.

In addition to the basic execution of tasks, the Python package features extra functionalities. As an example, it is possible to attach custom observer functions to optimization algorithms logging/forking additional data. A valid observer function has the following signature:

```
def observe(cycle_number: int, atoms: utils.AtomCollection, results: utils.Results tag: str):
    # do something with the data
```

An example use case could be the storage of all intermediate structures and energies in a geometry optimization:

```
import scine_utilities as utils
import scine_readuct as readuct

water = utils.core.load_system_into_calculator('h2o.xyz', 'PM6', program='Sparrow',
                                               molecular_charge=0, spin_multiplicity=1)
systems = {
    'water': water
}

class Gatherer:
    """ Collects energies and structures """
    def __init__(self):
        self.count = 0
        self.energies = []
        self.structures = []
    def collect(self, cycle_number, atoms, results, tag):
        self.count += 1
        self.energies.append(results.energy)
        self.structures.append(atoms)

observer = Gatherer()
readuct.run_optimization_task(
    systems,
    ['water'],
    False,
    [observer.collect],
    stop_on_error=False
)
```

```
print(f"Gathered {observer.count} energies and structures.")
print(f"Final energy was: {observer.energies[observer.count-1]} Hartree")
```

A detailed list of all the functions provided by the ReaDuct Python library can be found by running

```
import scine_readuct

help(scine_readuct)
```

Important References

Please consult the following references for more details on ReaDuct. We kindly ask you to cite the following reference in any publication of results obtained with ReaDuct.

A. C. Vaucher, M. Reiher "Minimum Energy Paths and Transition States by Curve Optimization", *J. Chem. Theory Comput.*, **2018**, *16*, 3091.

Bibliography

[1] Husch, T.; Vaucher, A. C.; Reiher, M. Semiempirical molecular orbital models based on the neglect of diatomic differential overlap approximation, *Int. J. Quantum Chem.* **2018,** *118,* e25799.

[2] Frisch, M. J. *et al.* "Gaussian 09, revision D. 01", 2009 Gaussian, Inc., Wallingford CT.

[3] Neese, F. The ORCA program system, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012,** *2,* 73–78.

[4] "TURBOMOLE V7.4.1 2019, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from http://www.turbomole.com.", .

[5] Kühne, T. D. *et al.* CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations, *J. Chem. Phys.* **2020,** *152,* 194103.

[6] Unsleber, J. P.; Dresselhaus, T.; Klahr, K.; Schnieders, D.; Böckers, M.; Barton, D.; Neugebauer, J. Serenity: A subsystem quantum chemistry program, *J. Comput. Chem.* **2018,** *39,* 788–798.

[7] Bannwarth, C.; Caldeweyher, E.; Ehlert, S.; Hansen, A.; Pracht, P.; Seibert, J.; Spicher, S.; Grimme, S. Extended tight-binding quantum chemistry methods, *WIREs Comput. Mol. Sci.* **2020,** *11,* e01493.

[8] Barone, V.; Cossi, M. Quantum Calculation of Molecular Energies and Energy Gradients in Solution by a Conductor Solvent Model, *J. Phys. Chem. A* **1998,** *102,* 1995–2001.

[9] Marenich, A. V.; Cramer, C. J.; Truhlar, D. G. Universal solvation model based on solute electron density and on a continuum model of the solvent defined by the bulk dielectric constant and atomic surface tensions, *J. Phys. Chem. B.* **2009,** *113,* 6378–6396.

[10] Klamt, A.; Schüürmann, G. COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient, *J. Chem. Soc. Perkin Trans. 2* **1993,** 799–805.

[11] Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, *J. Chem. Phys.* **2010,** *132,* 154104.

[12] Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the damping function in dispersion corrected density functional theory, *J. Comput. Chem.* **2011,** *32,* 1456–1465.

[13] Caldeweyher, E.; Ehlert, S.; Hansen, A.; Neugebauer, H.; Spicher, S.; Bannwarth, C.; Grimme, S. A generally applicable atomic-charge dependent London dispersion correction, *J. Chem. Phys.* **2019,** *150,* 154122.

[14] Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy, *Phys. Chem. Chem. Phys.* **2005,** *7,* 3297–3305.

[15] Dunning, T. H. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen, *J. Chem. Phys.* **1989,** *90,* 1007–1023.

[16] Ditchfield, R.; Hehre, W. J.; Pople, J. A. Self-consistent molecular-orbital methods. IX. An extended gaussian-type basis for molecular-orbital studies of organic molecules, *J. Chem. Phys.* **1971,** *54,* 720–723.

[17] Vaucher, A. C.; Reiher, M. Steering Orbital Optimization out of Local Minima and Saddle Points Toward Lower Energy, *J. Chem. Theory Comput.* **2017,** *13,* 1219–1228.

[18] Becke, A. D. A multicenter numerical integration scheme for polyatomic molecules, *J. Chem. Phys.* **1988,** *88,* 2547-2553.

[19] Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. Achieving linear scaling in exchange–correlation density functional quadratures, *Chem. Phys. Lett.* **1996,** *257,* 213-223.

[20] Meli, R. "IRC", `https://github.com/RMeli/irc`, 2019.

[21] Bofill, J. M. Updated Hessian Matrix and the Restricted Step Method for Locating Transition Structures, *J. Comput. Chem.* **1994,** *15,* 1-11.

[22] Farkas, O.; Schlegel, H. B. Methods for optimizing large molecules, *Phys. Chem. Chem. Phys.* **2002,** *4,* 11–15.

[23] Henkelman, G.; Jónsson, H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, *J. Chem. Phys.* **1999,** *111,* 7010–7022.

[24] Kästner, J.; Sherwood, P. Superlinearly converging dimer method for transition state search, *J. Chem. Phys* **2008,** *128,* 014106.

[25] Shang, C.; Liu, Z.-P. Constrained Broyden minimization combined with the dimer method for locating transition state of complex reactions, *J. Chem. Theory Comput.* **2010,** *6,* 1136–1144.

[26] Reddi, S. J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond, **2019,** arXiv:1904.09237 [cs.LG].

[27] Baker, J.; Chan, F. The Location of Transition States: A Comparison of Cartesian, Z-Matrix, and Natural Internal Coordinates, *J. Comput. Chem.* **1996,** *17,* 888–904.

[28] Maeda, S.; Morokuma, K. Communications: A systematic method for locating transition structures of A+B $\rightarrow$ X type reactions, *J. Chem. Phys.* **2010,** *132,* 241102.

[29] Maeda, S.; Morokuma, K. Finding Reaction Pathways of Type A + B $\rightarrow$ X: Toward Systematic Prediction of Reaction Mechanisms, *J. Chem. Theory Comput.* **2011,** *7,* 2335–2345.

[30] Vaucher, A. C.; Reiher, M. Minimum Energy Paths and Transition States by Curve Optimization, *J. Chem. Theory Comput.* **2018,** *14,* 3091–3099.