THE SCINE READUCT DEVELOPERS:

CHRISTOPH BRUNKEN, JAN UNSLEBER, ALAIN VAUCHER, AND MARKUS REIHER

USER MANUAL

SCINE READUCT 1.0.0

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

[3] 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 can be used on any computer with a 64-bit x86 architecture. The software 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 invovled 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++14 standard (we recommend gcc 7.3.0),
- cmake (we recommend version 3.9.0),
- the Boost library (we recommend version 1.64.0), and
- the Eigen3 library (we recommend version 3.3.2).

In order to compile the software, either directly clone the repository with git or extract the downloaded tarball, change to the source directory and execute the following steps:

```
git submodule init
git submodule update
mkdir build install
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SPARROW=ON -DCMAKE_INSTALL_PREFIX=../install ..
make
make test
make install
export SCINE_MODULE_PATH=<source code directory>/install/lib
export PATH=$PATH:<source code directory>/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 this last command, you have to replace `<source code directory>` with the full path where you stored the source code of ReaDuct.

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.

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: [method 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) and a calculation program (such as SCINE Sparrow or ORCA) and method (such as PM6). Depending on the program and method 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: 'PM6'
    settings:
      molecular_charge: 0
      spin_multiplicity: 1

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

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 L-BFGS algorithm[4].

[4] Nocedal, J. Updating Quasi-Newton Matrices With Limited Storage, *Math. Comp.* **1980**, *35*, 773–782

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 (such as `'PM6'`) in the `method` key. 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).

ORCA

Important note: Support for ORCA[5] is currently not fully tested. There might be specific calculation types and/or settings which do not work. Also, we cannot guarantee compatibility with any ORCA version different from 4.1.0 since we have no control over the output format of an external program. If you encounter any problems when using ORCA together with ReaDuct, please write a short message to scine@phys.chem.ethz.ch.

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

In order to use ORCA with ReaDuct, specify `program: 'OSUtils'` and `method: 'ORCA'` in the respective system block. You can specify the following settings in the settings block:

- `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.
- `orca_method`: This specifies the method string. By default, it is `PBE def2-SVP`, *i.e.,* a DFT calculation with the PBE exchange–correlation functional and the def2-SVP basis set is carried out. You can specify any valid ORCA method string (see the ORCA manual for a complete list).
- `self_consistence_criterion`: The threshold to which the electronic energy should be converged (given in hartree). By default, it is 10^{-6} hartree.
- `orca_nprocs`: The number of processors to use in the ORCA calculations. By default, it is one, *i.e.*, a serial calculation is carried out. Note that you have to specify the full ORCA binary path in case you want to do a parallel calculation (see below).
- `orca_binary_path`: This is used to specify the path to the ORCA binary. By default, this is set to “orca”, *i.e.*, this option need not be specified if ORCA is in your path and you want to do a serial calculation (`orca_nprocs: 1`). For a parallel calculation, you have to specify the full (absolute) path to your ORCA binary here.

- `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".
- `base_working_directory`: This specifies the directory in which the files for the ORCA calculations will be stored. By default, this is set to the current directory. For each ORCA calculation a new directory will be created inside the directory specified by `base_working_directory` to keep the files related to that specific calculation.

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'`.

Hessian Calculation

This task calculates the Hessian of a given system and outputs the vibrational frequencies. 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'`.

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 `'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 `'lbfgs'`.

- `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 `100`.
- `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`.

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 `50`.
- `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 `0.1`.

If you specified `optimizer: 'newtonraphson'` or `optimizer: 'nr'`, you can also set the following options:

- `nr_trust_radius`: The trust radius (maximum root mean square) 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[6], or any of `'eigenvector_following'`, `'eigenvectorfollowing'`, `evf`, or `ev` for a eigenvector following algorithm. By default, it is set to `'bofill'`.
- `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`.

[6] 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

- `convergence_max_iterations`: The maximum number of iterations. By default set to `100`.
- `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`.

If you specified `optimizer: 'bofill'`, you can also set the following option:

- `bofill_trust_radius`: The maximum root mean square of a taken step. By default set to `0.1`.

If you specified `optimizer: 'eigenvector_following'`, `'eigenvectorfollowing'`, `evf`, or `ev`, you can also set the following option:

- `ev_trust_radius`: The maximum root mean square of a taken step. By default set to `0.5`.

Intrinsic Reaction Coordinate Calculation

This task is used to an intrinsic reaction coordinate (IRC) calculation. 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 on will contain the result of the backward IRC calculation.

You usually want to set the following settings:

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

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'`.
- `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 `100`.

- `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`.

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 `50`.

- `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 `0.1`.

Artificial Force Induced Reaction Calculation

This task is used in order to do an artificial force induced reaction (AFIR[7]) 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'`.

[7] 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`.
- `afir_phase_in`: The number of steps over which the full attractive force is gradually applied. By default set to `100`.
- `afir_transform_coordinates`: Whether to transform the coordinates from a Cartesian basis into an internal space. By default set to `true`.
- `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'`.
- `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 `100`.
- `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`.

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 `50`.
- `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 `0.1`.

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: ’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/lib
```

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_readuct

system1 = scine_readuct.load_system('h2o.xyz', 'PM6', program='Sparrow',
                                    molecular_charge=0, spin_multiplicity=1)

systems = {}
systems['water'] = system1

task1 = scine_readuct.run_opt_task(systems, ['water'], output=['water_opt'],
                                   optimizer='lbfgs')
systems['water_opt'].positions
```

Note that we use a dictionary called “systems” to store all systems 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.

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

```
import scine_readuct

help(scine_readuct)
```

Extensions Planned in Future Releases

- Interfaces to other quantum chemical packages such as GAUSSIAN and SERENITY[8]
- Implementation of B-Spline optimization of transition state structures[9]

[8] 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

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

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

[3] Nocedal, J. Updating Quasi-Newton Matrices With Limited Storage, *Math. Comp.* **1980,** *35,* 773–782.

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

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

[6] 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.

[7] 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.

[8] 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.

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