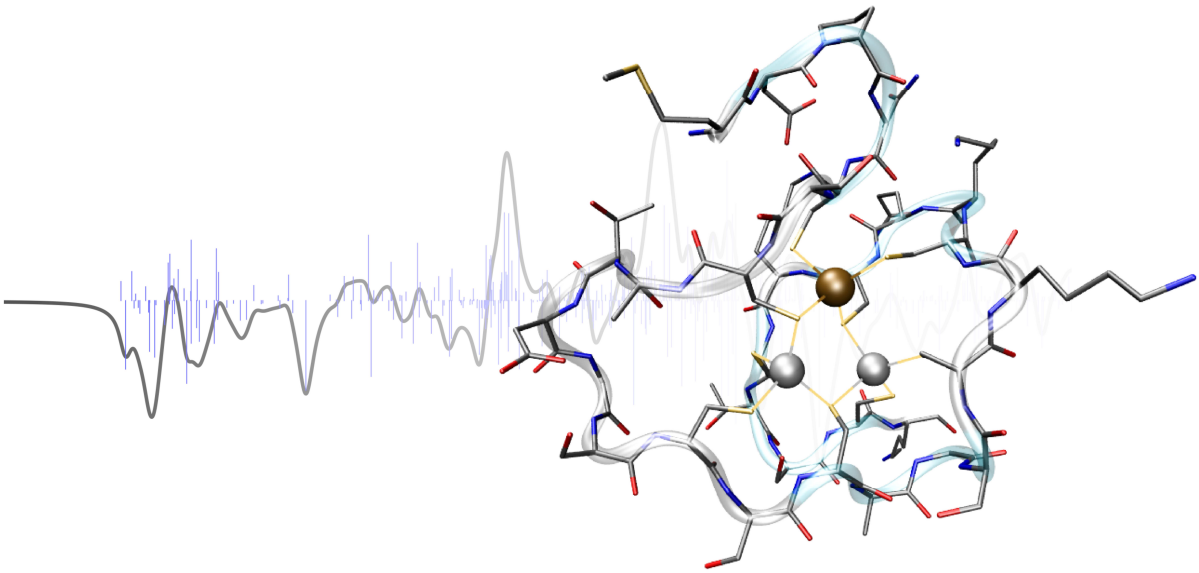
MOVIPAC

Quantum Chemical Calculation of Vibrational Spectra



Manual written by

Moritz P. Haag, Carmen Herrmann, Bernd A. Hess, Christoph R. Jacob, Carsten Kind, Sandra Luber, Johannes Neugebauer, Markus Reiher, Stephan Schenk, Thomas Weymuth

Laboratorium für Physikalische Chemie[1], ETH Zurich,
Wolfgang-Pauli-Strasse 10, 8093 Zurich, Switzerland

Version 1.0.1
January 2013

[1]email: markus.reiher@phys.chem.ethz.ch

Any use of this program that results in published material should **cite the following**:
Thomas Weymuth, Moritz P. Haag, Karin Kiewisch, Sandra Luber, Stephan Schenk, Christoph R. Jacob, Carmen Herrmann, Johannes Neugebauer, Markus Reiher. MoViPac: Vibrational Spectroscopy with a Massively Parallelized, Robust and Inverse Meta-Program. *J. Comput. Chem.*, **33** (2012) 2186–2198.

The cover picture shows the molecular structure of the β-domain of rat metallothionein, alongside with the corresponding Raman optical activity spectrum which has been calculated with MoViPac. The structure contains 30 amino acid residues, resulting in a total of 411 atoms. 2466 individual quantum chemical computations had to be performed in order to obtain the spectrum. This is the largest structure for which such a spectrum has ever been computed; this was only possible because of the efficient parallelization and restartability scheme of MoViPac (see also Ref. [2]).

Contents

1. Introduction

The aim of this manual is to give a description of the installation, the usage, and the performance of the vibrational spectra calculations using MoViPac. This package mainly includes the programs Snf, used for the parallel computation of (conventional) vibrational spectra, and Akira, which implements the techniques of Mode- and Intensity-Tracking.

For the underlying theory of vibrational spectrocopy we refer to the literature [3–7]. This software is under constant development.

If you applay the MoViPac software please make a proper reference to
Thomas Weymuth, Moritz P. Haag, Karin Kiewisch, Sandra Luber, Stephan Schenk, Christoph R. Jacob, Carmen Herrmann, Johannes Neugebauer, Markus Reiher. MoViPac: Vibrational Spectroscopy with a Massively Parallelized, Robust and Inverse Meta-Program. *J. Comput. Chem.*, **33** (2012) 2186–2198,
and to the specific paper that describes the feature(s) that you are using in order to make your work reproducable. The following authors have contributed to the MoViPac software: Noah S. Bieler, Moritz P. Haag, Carmen Herrmann, Bernd A. Hess, Christoph R. Jacob, Karin Kiewisch, Carsten Kind, Sandra Luber, Johannes Neugebauer, Markus Reiher, Stephan Schenk, Thomas Weymuth.

1.1 The program package Snf

The program package Snf has been developed [3] for the calculation of vibrational spectra. Infrared, Raman, and vibrational Raman optical activity (VROA) spectra can be obtained using the *harmonic* approximation for the frequencies and the *double harmonic* approximation for the intensities. Vibrational frequencies are determined using numerical differentiation of analytical gradients of the total electronic energy, while infrared, Raman and VROA intensities are obtained by numerical differentiation of dipole moments and (generalized) polarizability tensor components with respect to Cartesian nuclear coordinates.

Snf takes maximum advantage of the molecular point group[1]. By using the data collector Snfdc, all required single point calculations can be performed using *coarse-grained*

[1]Strictly speaking, we are concerned with the molecular symmetry groups [8–10]. However, as mentioned in [10, p. 16] the symmetry group of a molecule is usually referred to as the *point group* and thus, we shall use this name in the following

parallelization (different MPI versions can be applied) with automatic *load-balancing* on almost any Unix-compatible operating system.

Many different quantum chemical methods can be applied in order to perform the single point calculations, because input and output files are used as interfaces between SNF and different quantum chemical programs. Only their output files are processed to obtain the data, i.e., gradients of the total electronic energy, dipole moments, and polarizabilities. Existing interfaces allow one to use the following electronic-structure programs and methods: [2]

- TURBOMOLE [11]: SCF (HF), (RI-)DFT, (RI-)MP2, RICC2
- DALTON [12]: SCF (HF), MCSCF, DFT, CCS, CCSD, CC2
- GAUSSIAN [13]: DFT (other methods possible, but not tested)
- ADF [14]: DFT
- MOLPRO [15]: test interface for MCSCF calculations

A scheme of the program structure is shown in Fig. 1.1. Due to the numerical differentiation it is possible to use static as well as dynamic (frequency-dependent) polarizabilities for the calculation of the Raman intensities. By increasing the number of grid points for the numerical derivatives, it is possible to achieve an accurate error control.

SNF diagonalizes the Hessian and determines the translational and rotational contributions to the normal modes. In a second evaluation of the data, non-vibrational contributions to the matrix containing the second derivatives are projected off. The calculation is completed by a detailed thermochemical analysis, which takes electronic contributions into account (spin-only values are considered).

The preparation of the input files and the selection of most program options can be done by using the interactive program SNFDEFINE. The results of the calculations can easily be displayed since spectra plots are automatically created using GNUPLOT [16], and the normal modes can be written to files in GAUSSIAN98 [17] and MOPAC [18] format, such that tools like MOLDEN [19] or JMOL [20] may be utilized to visualize them.

1.2 The program package AKIRA

The AKIRA program [21] implements the MODE-TRACKING technique, which allows you to target normal modes directly and circumvents the computer-time demanding calculation of *all* normal modes. However, though we have put much effort into setting up a

[2] Note that VROA calculations are currently only possible for DALTON with HF/DFT, as well as with a special version of TURBOMOLE, which is, however, not open to the public. Raman intensities are not available for MP2 methods. However, SNF supports the calculation of MP2 gradients and dipole moments in combination with a SOPPA calculation of polarizabilities using DALTON.

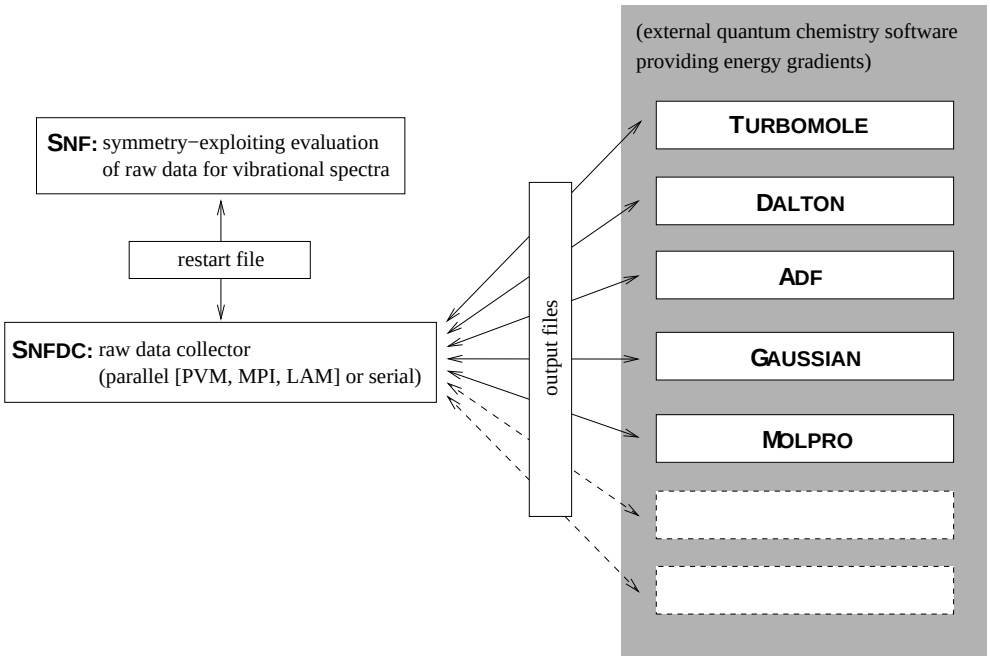


Figure 1.1: Hierarchical structure of programs.

computer program which is almost a *black box* code, it does not replace thinking about the particular problem to be solved and requires a basic understanding of vibrational analyses in general (in order to gain this, the reader is referred to the literature below and references cited therein). Therefore, when you have installed the program, our advice is to start with some small toy examples in order to learn about how the algorithm works (some test examples come with the package). In general, MODE-TRACKING is well suited in cases of

1. very large molecules for which a complete calculation of the spectrum is not feasible,

2. standard systems if only limited computer time is available so that a complete calculation cannot be carried out, and

3. smaller systems in combination with highly accurate *ab initio* calculations.

The development of the program AKIRA started in the Theoretical Chemistry department at the University of Erlangen–Nuremberg for an efficient calculation of vibrational frequencies and normal modes. The Mode-Tracking protocol developed for and used in AKIRA allows the specific calculation of characteristic normal modes as introduced in Ref. [21] (see also [22]), while normal frequency analyses always determine the full set of normal modes and frequencies of a molecule. This is usually much more information than

required, since in many cases only a small subset of characteristic vibrations is desired and necessary for comparison to or prediction of experimental data.

Akira originated in parts from the vibrational spectroscopy program package Snf of the Theoretical Chemistry Group at the University of Erlangen–Nuremberg. The original Snf package aims at the parallelized calculation of complete vibrational spectra within the double harmonic approximation (Snf [3]).

Akira follows a different strategy for the determination of vibrational normal modes involving predefined (collective) motions of the atoms in a molecule. These motions (= cartesian distortions) have to be selected by the user, and the algorithm applied in Akira will determine all normal modes of vibrations which involve these characteristic motions. Since the creation of the initial (guess) vibration is the most delicate step for the user, we have created a setup tool for this purpose:

The setup tool Akiradefine features several modules to create initial motions, either as Cartesian or internal coordinates, or from lower-level approximations like semiempirical models or force field calculations. Furthermore it is possible to use normal modes of former Snf or Turbomole calculations as an initial guess for the desired normal mode, so that calculations with smaller basis sets can be applied as initial guesses for the mode-tracking with a larger basis set. Another possibility is to use normal modes for a sub-system of the molecule as a guess for the normal mode of the complete molecule.

Akira uses subspace iteration techniques in order to find only those roots of the Hessian Matrix whose eigenvectors show the largest overlap with the initial guess vectors. Davidson- [23], Jacobi-Davidson- [24], and Lanczos-type [25] diagonalization schemes can be selected for this purpose.

1.3 Key references

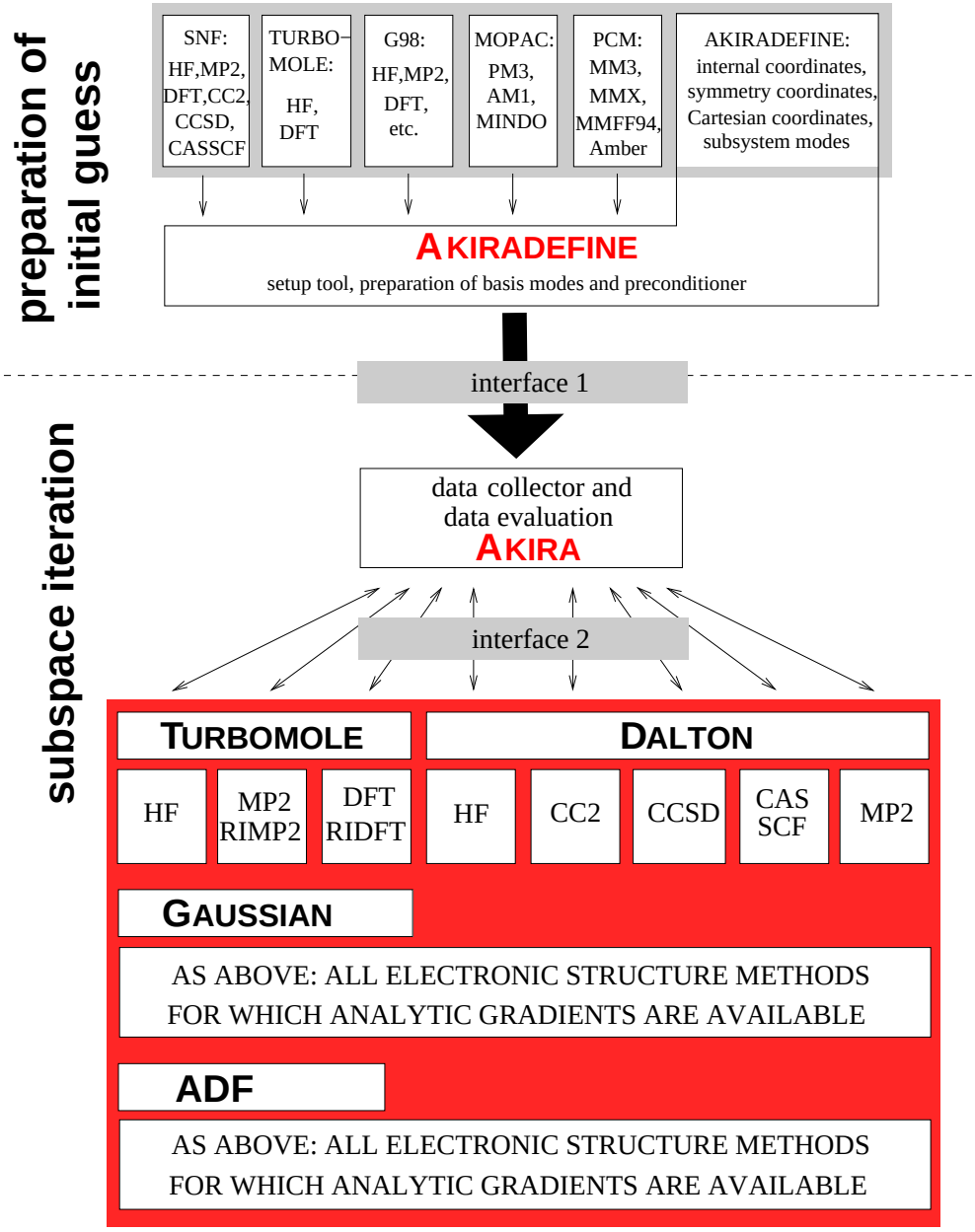
Many details of the implementation of MoViPac and the underlying theory as well as applications are available in the literature. The following list gives an overview of references for selected topics:

1.3.1 Algorithms and programs:

- **MoViPac:**
  Thomas Weymuth, Moritz P. Haag, Karin Kiewisch, Sandra Luber, Stephan Schenk, Christoph R. Jacob, Carmen Herrmann, Johannes Neugebauer, Markus Reiher. MoViPac: Vibrational Spectroscopy with a Massively Parallelized, Robust and Inverse Meta-Program. *J. Comput. Chem.*, **33** (2012) 2186–2198.

- **Snf, theory of infrared and Raman spectroscopy:**
  J. Neugebauer, M. Reiher, C. Kind, B. A. Hess. Quantum chemical calculation

Figure 1.2: Hierarchical structure of the mode-tracking program AKIRA.



of vibrational spectra of large molecules—Raman and IR spectra for Buckminsterfullerene. *J. Comput. Chem.*, **23**(9) (2002) 895–910.

- Akira, **Mode-Tracking:**
  M. Reiher, J. Neugebauer. A mode-selective quantum chemical method for tracking molecular vibrations applied to functionalized carbon nanotubes. *J. Chem. Phys.*, **118**(4) (2003) 1634–1641.

- **QM/MM vibrational mode tracking:**
  C. Herrmann, J. Neugebauer, M. Reiher. QM/MM vibrational mode tracking. *J. Comput. Chem.*, **29**(14) (2008) 2460–2470.

- LocVib, **localization of normal modes:**
  Christoph R. Jacob, Markus Reiher. Localizing normal modes in large molecules. *J. Chem. Phys.*, **130** (2009) 084106.

- **CTTM:**
  N. S. Bieler, M. P. Haag, C. R. Jacob, M. Reiher. Analysis of the Cartesian Tensor Transfer Method for Calculating Vibrational Spectra of Polypeptides. *J. Chem. Theory Comput.*, **7** (2011) 1867–1881.

- **Infrared Intensity-Tracking:**
  Sandra Luber, Johannes Neugebauer, Markus Reiher. Intensity tracking for theoretical infrared spectroscopy of large molecules. *J. Chem. Phys.*, **130**(6) (2009) 064105.

- **Raman and ROA Intensity-Tracking:**
  Sandra Luber, Markus Reiher. Intensity-carrying modes in raman and raman optical activity spectroscopy. *ChemPhysChem*, **10** (2009) 2049–2057.

- **resonance Raman Intensity-Tracking:**
  K. Kiewisch, J. Neugebauer, M. Reiher. Selective calculation of high-intensity vibrations in molecular resonance Raman spectra. *J. Chem. Phys.*, **129**(20) (2008) 204103.

- **general review of theoretical vibrational spectroscopy for large molecules:**
  C. Herrmann, M. Reiher. First-principles approach to vibrational spectroscopy of biomolecules. *Top. Curr. Chem.*, **268** (2007) 85–132.

- **review on Intensity-Tracking:**
  Karin Kiewisch, Sandra Luber, Johannes Neugebauer, Markus Reiher. Intensity tracking for vibrational spectra of large molecules. *CHIMIA*, **63** (2009) 270–274.

- **review of methods for the direct determination of vibrational signatures (cf., Mode-Tracking) in complex systems:**
  C. Herrmann, J. Neugebauer, M. Reiher. Finding a needle in a haystack: Direct determination of vibrational signatures in complex systems. *New J. Chem.*, **31**(6) (2007) 818–831.

1.3.2 Studies of interesting model systems

- **vibrational contribution to the entropy change associated with the low- to high-spin transition in iron complexes:**
  G. Brehm, M. Reiher, S. Schneider. Estimation of the vibrational contribution to the entropy change associated with the low- to high-spin transition in $Fe(phen)_2(NCS)_2$ complexes: Results obtained by IR and Raman spectroscopy and DFT calculations. *J. Phys. Chem. A*, **106**(50) (2002) 12024–12034.

- **fluorescence kinetics of aqueous solutions of tetracycline:**
  S. Schneider, M. O. Schmitt, G. Brehm, M. Reiher, P. Matousek, M. Towrie. Fluorescence kinetics of aqueous solutions of tetracycline and its complexes with Mg^{2+} and Ca^{2+}. *Photochem. Photobiol. Sci.*, **2**(11) (2003) 1107–1117.

- **IR and Raman spectra of tetracycline and derivatives:**
  C. F. Leypold, M. Reiher, G. Brehm, M. O. Schmitt, S. Schneider, P. Matousek, M. Towrie. Tetracycline and derivatives—assignment of IR and Raman spectra via DFT calculations. *Phys. Chem. Chem. Phys.*, **5**(6) (2003) 1149–1157.

- **chirality induced switch in hydrogen-bond topology:**
  T. B. Adler, N. Borho, M. Reiher, M. A. Suhm. Chirality-induced switch in hydrogen-bond topology: Tetrameric methyl lactate clusters in the gas phase. *Angew. Chem., Int. Ed.*, **45**(21) (2006) 3440–3445.

- **IR and Raman spectroscopical investigation of the low- to high-spin transition in novel iron complexes:**
  G. Brehm, M. Reiher, B. Le Guennic, M. Leibold, S. Schindler, F. W. Heinemann, S. Schneider. Investigation of the low-spin to high-spin transition in a novel $[Fe(pmea)(NCS)_2]$ complex by IR and Raman spectroscopy and DFT calculations. *J. Raman Spectrosc.*, **37**(1–3) (2006) 108–122.

- **importance of backbone angles versus amino acid configurations in peptide ROA:**
  C. Herrmann, K. Ruud, M. Reiher. Importance of backbone angles versus amino acid configurations in peptide vibrational Raman optical activity spectra. *Chem. Phys.*, **343**(2–3) (2008) 200–209.

- **VROA signatures of tryptophan side chains:**
  C. R. Jacob, S. Luber, M. Reiher. Calculated Raman optical activity signatures of tryptophan side chains. *ChemPhysChem*, **9**(15) (2008) 2177–2180.

- **VROA spectra of chiral transition metal complexes**:
  S. Luber, M. Reiher. Raman optical activity spectra of chiral transition metal complexes. *Chem. Phys.*, **346**(1–3) (2008) 212–223.
  Sandra Luber, Markus Reiher. Prediction of raman optical activity spectra of chiral 3-acetylcamphorato-cobalt complexes. *ChemPhysChem*, **11** (2010) 1876–1887

- **VROA signatures of α- and 3_{10}-helices:**
  Christoph R. Jacob, Sandra Luber, Markus Reiher. Understanding the Signatures of Secondary-Structure Elements in Proteins with Raman Optical Activity Spectroscopy. *Chem. Eur. J.*, **15** (2009) 13491–13508.

- **VROA signatures of β-turns:**
  Thomas Weymuth, Christoph R. Jacob, Markus Reiher. Identifying Protein β-Turns with Vibrational Raman Optical Activity. *ChemPhysChem*, **12** (2011) 1165–1175.

- **VROA spectrum of the β-domain of rat metallothionein:**
  S. Luber, M. Reiher. Theoretical Raman Optical Activity Study of the β Domain of Rat Metallothionein. *J. Phys. Chem. B*, **114** (2010) 1057–1063.

- **Calculated ROA spectra of 1,6-anhydro-β-D-glucopyranose:**
  Sandra Luber, Markus Reiher. Calculated raman optical activity spectra of 1,6-anhydro--d-glucopyranose. *J. Phys. Chem. A*, **113** (2009) 8268–8277.

1.3.3 Further references

- **benchmark and methods (DFT, CC, MCSCF, ...) comparison for Raman intensities from static and dynamic polarizabilities:**
  J. Neugebauer, M. Reiher, B. A. Hess. Coupled-cluster Raman intensities: Assessment and comparison with multiconfiguration and density functional methods. *J. Chem. Phys.*, **117**(19) (2002) 8623–8633.

- **methodological and benchmark study on vibrational Raman optical activity:**
  M. Reiher, V. Liégeois, K. Ruud. Basis set and density functional dependence of vibrational Raman optical activity calculations. *J. Phys. Chem. A*, **109**(33) (2005) 7567–7574.

- **role of anharmonicity contributions:**
  J. Neugebauer, B. A. Hess. Fundamental vibrational frequencies of small polyatomic

molecules from density-functional calculations and vibrational perturbation theory. *J. Chem. Phys.*, **118**(16) (2003) 7215–7225.

- **theoretical Raman scattering cross sections compared to experiment, error cancellation in harmonic frequencies obtained with the BP86 exchnage–correlation functional:**
  M. Reiher, G. Brehm, S. Schneider. Assignment of vibrational spectra of 1,10-phenanthroline by comparison with frequencies and Raman intensities from density functional calculations. *J. Phys. Chem. A*, **108**(5) (2004) 734–742.

- **intensities for selected vibrations of large systems:**
  M. Reiher, J. Neugebauer, B. A. Hess. Quantum chemical calculation of Raman intensities for large molecules: The photoisomerization of $[\{Fe`S_4`(PR_3)\}_2(N_2H_2)]$ (`S4'$^{2-}$ = 1,2-bis(2-mercaptophenylthio)-ethane(2−)). *Z. Phys. Chem.*, **217**(2) (2003) 91–103.

- **resonance Raman calculations, excited-state frequencies:**
  J. Neugebauer, B. A. Hess. Resonance Raman spectra of uracil based on Kramers–Kronig relations using time-dependent density functional calculations and multireference perturbation theory. *J. Chem. Phys.*, **120**(24) (2004) 11564–11577.

- **Mode-Tracking for a molecular subsystem:**
  J. Neugebauer, M. Reiher. Mode tracking of preselected vibrations of one-dimensional molecular wires. *J. Phys. Chem. A*, **108**(11) (2004) 2053–2061. J. Neugebauer, M. Reiher. Vibrational center–ligand couplings in transition metal complexes. *J. Comput. Chem.*, **25**(4) (2004) 587–597.

- **convergence analysis of the Mode-Tracking algorithm (systematic comparison of inital guesses, preconditioners, near-degeneracies):**
  M. Reiher, J. Neugebauer. Convergence characteristics and efficiency of mode-tracking calculations on pre-selected molecular vibrations. *Phys. Chem. Chem. Phys.*, **6**(19) (2004) 4621–4629.

- **comment on gradient-based direct normal-mode analysis:**
  M. Reiher, J. Neugebauer. Comment on "Gradient-based direct normal-mode analysis" [J. Chem. Phys. 122, 184106 (2005)]. *J. Chem. Phys.*, **123**(11) (2005) 117101.

- **theoretical VROA applied to large systems:**
  C. Herrmann, K. Ruud, M. Reiher. Can Raman optical activity separate axial from local chirality? A theoretical study of helical deca-alanine. *ChemPhysChem*, **7**(10) (2006) 2189–2196.

- **Mode-Tracking for adsorbates on surfaces, harmonic approximation for partially relaxed structures:**

C. Herrmann, M. Reiher. Direct targeting of adsorbate vibrations with mode-tracking. *Surf. Sci.*, **600**(9) (2006) 1891–1900.

- **high-performance Raman applications for large molecules:**
  C. Herrmann, J. Neugebauer, M. Presselt, U. Uhlemann, M. Schmitt, S. Rau, J. Popp, M. Reiher. The First photoexcitation step of ruthenium-based models for artificial photosynthesis highlighted by resonance Raman spectroscopy. *J. Phys. Chem. B*, **111**(21) (2007) 6078–6087.

- **relevance of the electric-dipole–electric-quadrupole contribution to VROA spectra:**
  S. Luber, C. Herrmann, M. Reiher. Relevance of the electric-dipole–electric-quadrupole contribution to Raman optical activity spectra. *J. Phys. Chem. B*, **112**(7) (2008) 2218–2232.

- **Analysis of secondary structure effects in terms of localized modes:**
  Christoph R. Jacob, Sandra Luber, Markus Reiher. Analysis of Secondary Structure Effects on the IR and Raman Spectra of Polypeptides in Terms of Localized Vibrations. *J. Phys. Chem. B*, **113** (2009) 6558–6573.

- **Predicting extended amide III frequencies with localized vibrations:**
  Thomas Weymuth, Christoph R. Jacob, Markus Reiher. A Local-Mode Model for Understanding the Dependence of the Extended Amide III Vibrations on Protein Secondary Structure. *J. Phys. Chem. B*, **114** (2010) 10649–10660.

- **Enhancement and de-enhancement effects in resonance ROA:**
  Sandra Luber, Johannes Neugebauer, Markus Reiher. Enhancement and de-enhancement effects in vibrational resonance raman optical activity. *J. Chem. Phys.*, **132**(4) (2010) 044113.

Part I

Snf

2. Introduction

This part of the manual covers the program SNF. It is organized as follows: Sect. 3 gives a short introduction to the most relevant issues for practical applications of SNF. In Sect. 4, a description of the installation of SNF is given, and the requirements for the parallelization are explained. Sect. 5 deals with usage of the programs; all parts are explained in detail, starting at the preparation of the input files with SNFDEFINE and ending with the evaluation of the data collected by SNFDC with SNF. Sect. 6 contains information about all possible keywords for SNF and their meaning. In Sect. 7, the visualization of the results obtained with SNF is explained. Sect. 8 outlines details of the implementations of thermochemistry, and in Sect. 11, the influence of step size, basis sets, SCF convergence, and grid type on the results of an SNF run is discussed. The different sections at the end concern the program history (chapter 13), the supported point groups (chapter 14), program modes (chapter 15), and compilers (chapter 16). Moreover, details of the elimination of vibrational and rotational modes can be found in chapter 17. Finally, chapter 18 presents an overview of the tools which are provided with the SNF Package.

Note that this program represents experimental code that is under constant development. No guarantees of any kind are provided, and the authors do not accept any responsibilities for the performance of the code or the correctness of the results.

3. Quick start

This section provides a short summary of the most relevant practical aspects of SNF. For a detailed discussion of these issues, see the next chapters.

Installation

To install SNF, you need the following steps:

- read this manual
- unpack: `tar -xvjf snf-4.2.1.tar.bz2`
- change to subdirectory: `cd snf-4.2.1`
- set up build environment: `autoreconf -i -v`
- configure package: `./configure --prefix=PREFIX_DIR --with-mpi`
- compile package: `make`
- install binaries to `PREFIX_DIR`: `make install`
- clean up installation process: `make clean`

`PREFIX_DIR` denotes the directory where the SNF binaries are installed to. The path to this directory needs to be assigned to the environment variable `SNF_PATH`. Moreover, `SNF_PATH/bin` needs to be added to the environment variable `PATH`.

Checklist

Before running SNF, assure that the SNF environment is properly set up, i.e., that

- the environment variable `\$SNF\_PATH` is set to the SNF directory where the binaries you want to test are stored (in `\$SNF\_PATH/bin`),
- your are able to log in on all nodes under their full names (e.g., `yourmachine.somewhere.org` and under any shortcut names you want to use (e.g., `yourmachine`) without having to enter any passwords or confirmations,

- the executables of the quantum chemistry programs needed for the slave jobs, i.e., TURBOMOLE, DALTON, ADF, GAUSSIAN, or MOLPRO, are available on every slave node directly after opening a shell there. If a special path "`path/to/executable`" is to be used, it needs to be defined in the `DCINPUT` file of the working directory as "`PROGDIR path/to/executable`",

- the script `mkdcinput` (stored in `\$SNF\_PATH/scripts`) is included in your `\$PATH`.

To apply the Cartesian Tensor Transfer Method (CTTM) the PYVIB2 library developed by Maxim Fedorovsky (http://pyvib2.sourceforge.net) has to be installed. The path to this library has to be added to `$PYTHONPATH` or the file `cttm.py` in the `bin` directory has to be changed accordingly.

Essential steps

In order to calculate a vibrational spectrum with SNF, you should proceed as follows:

1. Assure that all necessary input files are present in your working directory. Which input files are required depends on the quantum chemical program package to be used for the single-point calculations (see Sect. 5.2).

2. Run `snfdefine` in order to set up the calculation. SNF settings are written to `snf_control` (see Sect. 5.4).

3. Type `snfdc` in order to start the parallel evaluation of molecular gradients and properties for the distorted structures. These data will be written to the file `restart` (see Sect. 5.5).

4. Type `snf` in order to evaluate the raw data produced by `snfdc`. The output is contained in `snf.out` (see Sect. 5.6).

4. Installation

The package SNF was developed for large-scale numerical calculations of vibrational frequencies, which require hundreds of calculations for displaced structures. It is therefore intended to be used in a parallel environment, and the correct setup of such a parallel environment is thus a prerequisite for installing SNF. This makes the installation process for SNF somewhat more involved, although much effort was made to simplify this step. The different parallelization standards which can be used with SNF are described in the following section.

Since version 4.0.0, SNF also includes a serial version which allows new SNF users to get acquainted with the SNF machinery under "simplified" conditions, or to run SNF on single-processor machines. This version is automatically installed if you do not activate any of the parallelization methods below.

4.1 Parallel versions

SNF may run either serially where all calculations are performed successively on the same processor or in parallel. In order to use the parallelized version of the data collector in SNF, it is necessary to provide the software for one of the following parallelization standards.

4.1.1 MPI

MPI has evolved over time to be the de-facto standard for parallelizing applications. Different implementations of the MPI standard are available. OpenMPI, MPICH2, and LAM-MPI, resp., may be obtained from
`http://www.open-mpi.org`,
`http://www-unix.mcs.anl.gov/mpi/mpich2`, or
`http://www.lam-mpi.org`,
respectively. We use OpenMPI for development and, therefore, recommend its usage. SNF will look for an MPI Fortran77 compiler (like `mpif77`) and will use it for compilation.

To enable MPI support, you have to set the `--with-mpi` flag when running the `configure` script (see Sect. 4.2).

4.2 Installation of the binaries

SNF comes as a source code tar-ball. To install it, you first need to unpack the archive using

```
tar -xvjf snf-4.2.1.tar.bz2
```

Next, change to the unpacked directory and set up the build environment using

```
autoreconf -i -v.
```

Next, execute the configure script:

```
./configure --prefix PREFIX_DIR.
```

A more detailed description of this script is given below. After the configuration has finished, compile the sources using the `make` command, and install the binaries to the directory `PREFIX_DIR` using `make install`. Finally, you can clean up the installation process by typing `make clean`.

The `configure` script allows you to adjust the installation according to your needs. It may also be helpful to look at the output of `./configure --help` to get a a list of supported options and influential environment variables. To enable parallel calculations, you need to give the appropriate option to `configure`. Use `--with-mpi` to enable MPI support. To change the default Fortran compiler to, e.g., `ifort` run configure as

```
./configure FC=ifort F77=ifort
```

where `FC` is the Fortran90 compiler used for SNF and SNFDEFINE, whereas `F77` is the Fortran77 compiler used for SNFDC. Using `scp` for file transfer is usually much faster than the intrinsic methods provided by the parallel machinery. However, this requires that scp does not ask for a password if called from within a program. To enable scp transfers, run

```
./configure --with-scp
```

The configure script searches for system BLAS and LAPACK libraries. To use a specific version, run

```
./configure --with-blas=<linker flags for linking to BLAS>
```

and/or

```
./configure --with-lapack=<linker flags for linking to LAPACK>.
```

If you have problems with the system libraries, you can use

```
./configure --without-blas
```

and/or

```
./configure --without-lapack
```

to enforce usage of the routines provided by SNF. To build the specplot utility (see Sect, 18.2), run

```
./configure --enable-specplot.
```

The entire build system makes use of autoconf and automake. You need at least automake-1.11 for the non-recursive build in `src/` to succeed. Earlier versions do *not* correctly support Fortran in this respect. If you want to change anything inside a Makefile, make the changes to `Makefile.am`. All other changes will be lost since `Makefile.in` and `Makefile` are autogenerated. Similarily, `aclocal.m4` is autogenerated. If you change `configure.ac` or any `Makefile.am` or any M4 source, run `autoreconf -i -v` afterwards to update the build system.

One can run into problems with the installation if the environment variable `TMPDIR` is set to a non-existing directory. `configure` does need to create certain temporary files and directories. By default, this is done in `/tmp`. However, if `TMPDIR` is set, files are created in the directory indicated by `$TMPDIR`. Thus, if you have problems with `configure` aborting because it is unable to create temporary files you may try to unset `TMPDIR`. In `bash` this is done with `unset TMPDIR`.

If you want to make a distribution, run simply `make dist`. This, of course, requires that `configure` was run before. To make sure that everything compiles and packs fine, run `make distcheck`.

4.3 Further programs and scripts needed by SNF

In order to perform the single point calculations, one of the supported electronic-structure programs must be available on every node in the parallel machinery in `$PATH`.

Furthermore, the scripts from the `scripts` subdirectory of the SNF installation directory must be available in `$PATH`. The script `choose_nodes` also requires the file `ELIGIBLE` in the directory `$CIPROC`, which contains the names of computers available and their numbers of processors (see Sect. 5.5). An example file is provided with the SNF package. The variable `\$CIPROC` must be set by the user. `choose_nodes` also requires the `ruptime` command from the Linux Netkit (`rwho`). See, however, Section 5.5 for how to circumvent the usage of `ruptime` and `ELIGIBLE`. For the automatic generation of spectra plots, `gnuplot` must also be present in `$PATH`.

The variable `$SNF_PATH` should be set to the SNF home directory, so that executables can be taken from `$SNF_PATH/bin`.

4.4 Test suite

Starting from version 4.0.0, a test script, `$SNF_PATH/examples/snftest`, is available which runs SNF test jobs and controls the results automatically.

If invoked as `$SNF_PATH/examples/snftest all`, this script will run a series of test jobs using all quantum chemistry programs for which interfaces are available (except for GAUSSIAN98, which may be tested separately). It can be run with one or several of the arguments `turbomole, dalton, adf, gaussian` or `molpro`, or also with names of test directories such as `turbomole/ridft/ir_raman`, in order to restrict the test to specific programs or test jobs only. It will run the jobs in the test subdirectories of the directory `$SNF_TESTDIR`, so this path should be set accordingly (usually to `$SNF_PATH/examples/snftest`). `snftest` features an automatic check of the results of each test run, which is performed by calling the script `checkresults`. Information on whether (and which) test jobs failed is written to the screen and to the file `$SNF_TESTDIR/snftest.log`.

5. Frequency analysis with SNF

The frequency analysis and the calculation of vibrational spectra using SNF consists of three steps, which are described in Sects. 5.4, 5.5, and 5.6. The following two sections give an overview of the whole procedure and the preparations necessary before starting the calculation.

5.1 Overview

The first step in the parallelized calculation of vibrational spectra is the preparation of the input files. Therefore, it is necessary to copy master input files for the single point calculations into the directory which serves as the working directory for the calculation. These input files are modified by the preparation tool SNFDEFINE, which also creates a restart file for the frequency analysis and the additional input file `snf_control`. The latter contains general program and output control flags. The generic names of the input files needed for the different electronic structure programs are:

- TURBOMOLE: `control`, `coord`, `basis`, `mos` or (`alpha`, `beta`) [, `auxbasis`]
- DALTON: `molecule.dal`, `molecule.mol`
- ADF: `adf.in`
- GAUSSIAN: `gaussian.com`
- MOLPRO: `molpro.in`

A detailed description follows in Section 5.2.

The next step, the most time-consuming part, is the parallelized calculation of gradients of the total electronic energy, dipole moments, and polarizabilities for each distorted molecular structure. These calculations have to be performed in C_1 symmetry since the distortions destroy the molecular symmetry. The calculations are done by the data collector SNFDC, which distributes the single point calculations to all processors available. The data is stored in the `restart` file.

When the single point calculations are finished, the data is evaluated by SNF, which creates the output file `snf.out` containing the vibrational frequencies, the infrared and Raman intensities, the thermochemical data, the normal modes and other data depending

on the output control flags. Additional output files can be generated for a graphical representation of the spectra and the normal modes.[1]

5.2 Preparations

The program SNF requires master input files, which are distributed to all nodes of the parallel machine and modified for each slave process. In general, the type of the input files provided by the user determines the quantum chemical program package which SNF will use in the single-point calculations on the slave nodes. An exception is possible in the case of TURBOMOLE input files, which can be automatically converted into DALTON input files, so that DALTON can be used for the single-point calculations.

In general, the necessary input files are automatically generated by a structure optimization. It is recommended to delete sections referring to this preceding structure optimization (e.g., specifications of frozen coordinates or temporary directories) before running an SNF calculation.

5.2.1 TURBOMOLE

It is necessary to provide the following files in the working directory for the frequency analysis:

- `control`: general input file for the TURBOMOLE calculation
- `coord`: nuclear coordinates of the atoms; this must be the optimized structure of the molecule
- `basis`: basis set information
- `mos`: molecular orbital coefficients; these MO vectors will be used as starting vectors for every single point calculation in the frequency analysis
- `alpha`, `beta`: the files `alpha` and `beta` have to be supplied instead of the file `mos` for UHF/UKS calculations
- `auxbasis`: auxiliary basis set information; only necessary for RI-DFT calculations

If a geometry optimization with TURBOMOLE has been performed, all these files are already created and SNFDEFINE can be started.

The type of the calculation is in principle determined by the type of the `control` file provided. If the geometry optimization performed was a Hartree–Fock `dscf` calculation, SNF will also perform HF single point calculations. If it was a `ridft` calculation, density

[1] If no isotope settings are specified SNF uses the masses of the most abundant isotopes for all evaluations (see Sect. 6.1.2).

functional calculations using the RI-approximation will be done. Note that no Raman intensities can be calculated using MP2/RI-MP2 methods, but it is possible to calculate Raman intensities from SOPPA polarizabilities in combination with frequencies and infrared intensities from MP2 calculations.

Starting from TURBOMOLE input, it is also possible to request the use of DALTON for the single-point calculations (which is necessary, for example, if VROA intensities shall be calculated after a TURBOMOLE geometry optimization). Appropriate input files in DALTON format will be generated by SNFDEFINE in this case, where the user has to specify the DALTON settings in an interactive menu.

5.2.2 DALTON

It is also possible to use DALTON input files. The following two files must be provided for this type of preparation:

- `molecule.dal`: general input file for the DALTON calculation.
- `molecule.mol`: nuclear coordinates of the atoms and basis set specification.

For DALTON applications, the type of the calculation is not determined automatically. Therefore, the user has to select the type of calculation in an interactive menu during the `snfdefine` run.

The user has to take care of the following formal constraints:

- the input files should be named `molecule.dal` (general input) and `molecule.mol` (geometry and basis set input)
- Z-matrix input is not supported; use Cartesian coordinates instead
- if the `INTGRL` basis set specification is applied, do not use the free format for exponents and coefficients; use either the default or the high precision format (`H` or `h`; see DALTON manual for details)
- do not use symmetry generators; you may either use the automatic symmetry detection routines of DALTON (`SYMTXT` = ' ') or a C_1 input (`SYMTXT`=' 0'). `snfdefine` determines the molecular symmetry through its own symmetry detection routines and sets the symmetry to C_1 in the single point calculations.

5.2.3 ADF

ADF users can use the runscript of an ADF geometry optimization directly. SNFDEFINE will recognize the `GEOMETRY` section and replace it by a section requiring a geometry optimization which is stopped after one cycle. Furthermore, some convergence options may be altered (see below). This runscript will be distributed onto the slave nodes, where

the molecular coordinates will be changed. Apart from this, the settings in the ADF scripts are automatically those which will be used in the single-point calculations on the slaves. The runscript has to fulfill several conditions:

- it must be named `adf.in`
- it must be executable
- it must contain exactly the settings used in the preceding geometry optimization
- the geometry must be specified in Cartesian coordinates in units of Å
- the binary output file must be named `TAPE21`
- it should not contain any blocks requesting other calculations than a geometry optimization

SNFDEFINE will check (and modify, if necessary) the following blocks in the `adf.in` file:

- `GEOMETRY`—delete it (if present) and write instead:

```
Geometry
  GO
  iterations 1
End
```

- `INTEGRATION`—if the grid settings are below `6.0 6.0`, replace them by `6.0 6.0`.
- `SCF`—if the convergence criteria are below `converge 1.0e-6 1.0e-6`, replace them by `converge 1.0e-6 1.0e-6`.
- `OCCUPATIONS`—the options `KEEPORBITALS=1000000 SMEAR=0.0d0` will be set. If not present, this keyword will be added. This prevents ADF from using SCF convergence tools which may lead to results that are only acceptable in a geometric structure optimization, but not in a frequency run.

If SNFDEFINE modifies the integration or convergence parameters, it will print a warning message to the screen remembering the user to redo a geometry optimization with the new settings.

5.2.4 Gaussian

If Gaussian shall be used for the single-point calculations, the user must provide a Gaussian input file which must have the following characteristics:

- it must be named `gaussian.com`
- it must contain the keyword `# Force` in addition to the keywords for the method and the basis set to be used
- the equilibrium geometry must be given in Cartesian coordinates—Z-matrix input is not accepted

For an ethanol molecule, the `gaussian.com` file might look like this (with the last line being a blank line, of course):

```
# Force bp86/TZVP

ethanol

0 1
C -1.225747   -0.225340    0.000002
H -1.296072   -0.865520   -0.891479
H -1.296085   -0.865499    0.891496
H -2.083095    0.464915   -0.000013
C  0.079992    0.551234   -0.000000
H  0.131474    1.203598    0.893871
H  0.131470    1.203596   -0.893875
O  1.161283   -0.399398   -0.000003
H  1.996579    0.098737    0.000020
```

Since this input file will be distributed onto the slave nodes, where nothing will be changed in it but the molecular coordinates, the settings defined there are automatically the settings which are used in the single-point calculations.

Optional additional sections (such as user-provided basis sets) are in general supported. However, sections related to the preceding structure optimization (such as specifications of frozen coordinates) should be deleted in the `gaussian.com` file.

5.2.5 Molpro

The Molprointerface in Snf is not yet fully developed, and using Molprowith Snf therefore requires special caution. Only MCSCF calculations can be carried out with Molproin connection with Snf at the moment. Furthermore, Snf only supports calculations with Molpro version 2002.x; the new input/output format in version 2006.x is not yet supported in all cases. The Molprointerface is suited for calculating IR intensities, but not for Raman or VROA intensities.

The following requirements apply to the input file:

- it must be named `molpro.in`
- it has to start with `***` in the first data line
- the coordinates of the equilibrium structure should be given in Å. Therefore, the keyword `geomtyp=xyz` is necessary and the geometry block (`geometry={ ... }`) should contain the Cartesian coordinates in XYZ format. This format consists of two header lines, the first of them containing the number of atoms, and the second one an optional title. Each of the remaining lines specifies the coordinates of one atom, with the chemical symbol in the first field, followed by the x, y, z coordinates. The coordinates and the chemical symbol have to be separated by a comma. A sequence number may be appended to the chemical symbol without using a comma. No blanks should be used.
- The interface is implemented for MCSCF calculations, therefore the keyword `mcscf` must be included in the input file.
- For gradient calculations, `gradtype=alaska` and `force` must be added.
- The basis is chosen with `BASIS=` and the name of the basis set.

For further information see `http://www.molpro.net`.

5.3 Preparations for excited-state calculations

Excited-state calculations can be performed using the `egrad` module of the TURBOMOLE package. This module allows it to analytically calculate excited state gradients, which can in turn be used for a geometry optimization of the excited state using SCF or DFT calculations. For the particular usage of the `egrad` program, we refer to the TURBOMOLE manual. It should be kept in mind, however, that the frequency calculation is performed in C_1 symmetry, and therefore, the number of the excited state to be calculated must include all lower-lying excited states of the same spin-symmetry. In a calculation with a non-trivial point-group symmetry, the state-numbering refers always to excited states of a particular irrep. Note that no Raman spectra can be calculated for these excited states.

Excited-state calculations and optimizations can also be performed using DALTON's CASSCF module.

CAUTION: SNF assumes that you are calculating frequencies far away from any avoided crossing/conical intersection/near-degeneracy situation. If this is not the case, then there is a potential danger that the electronic character of a certain excited state changes from the equilibrium to the displaced structures. Additionally, the Born–Oppenheimer approximation will not be valid in such a context, upon which the frequency analysis in SNF is based.

5.4 Snfdefine

Using the interactive program Snfdefine you can conviniently set up all necessary input files. You can run Snfdefine simply by typing `snfdefine` (see also Sect. 6 for a detailed description of the most important input options).

First, the program checks whether the mandatory files for running single-point calculations with the desired quantum chemical program package are present in the working directory. Then it tries to read the essential information from them. In case of Turbomole single-point calculations, after reading the Schoenflies symbol of the molecular point group from `control`, Snfdefine checks the symmetry of the molecule in the `coord` file and determines the distortions which are redundant because of symmetry. From version 2.2.1 on, Snfdefine can determine the correct symmetry of the molecule independent of its orientation in space, while former versions could only validate or falsify the specified symmetry[2]. The symmetry detection routines will only be applied if the Schoenflies symbol in the `control` file is inconsistent with the geometrical data in the `coord` file or if their invocation is selected by the user. If Dalton, Adf, Gaussian, or Molpro input files are used, Snfdefine will always determine the symmetry on its own.

For an n-point central differences formula, it is necessary to perform $(n-1)$ distortions in the direction of every Cartesian nuclear coordinate, i.e., $3 \cdot (n-1) \cdot N$ single point calculations are necessary for a molecule with N atoms if no symmetry information is available. However, the number of non-redundant distortions is by far smaller for highly symmetric molecules (compare benzene[3] (8 distortions instead of 72), SF_6 (4 instead of 42) or the Buckminsterfullerene C_{60} (5 instead of 360)). Snfdefine creates the file `restart` which contains entries for each of the $3 \cdot (n-1) \cdot N$ distorted geometries. As an example, the `restart` file for N_2 as created by Snfdefine is given below.

```
snf calculation
(4d20.10)
  410    1    1    1    6    8    2
    0.1000000000D-01
mydir
tempdir
  1  0  1  1  1  0  0
  ------  ------  ------  ------  ------  ------
  2  1  1  1  1  1  1
  ------  ------  ------  ------  ------  ------
 dipoles atom            1
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00
 dipoles atom            2
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00
 gradients atom            1
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00    0.0000000000D+00
```

[2]The symmetry detection routines available in former versions of Snf require Turbomole's standard orientation for a successful determination of the molecular point group (i.e., the z-axis must be the principal axis of rotation etc.). This is still required for cubic and icosahedral groups.

[3]3-point central differences formula assumed

```
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
gradients atom            2
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
polarizabilities atom           1
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
polarizabilities atom           2
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00   0.0000000000D+00
```

The entries have the following meanings:

line No.	meaning
1	title
2	format string for data (dipole moments, gradients, and polarizabilities)
3	`version`, `progno`, `calcno`, `numberfreqs`, `nvar, naus, nummodes`
4	`cstep`
5	working directory
6	scratch directory

with the variables

- `version`: version number of this restart file
- `progno, calcno`: characterizes the type of the calculation (see chapter 15)
- `numberfreqs`: number of sets of dynamic polarizabilities. SNF allows to calculate Raman intensities based on frequency-dependent polarizabilities for up to 10 different frequencies.

- `nvar`: number of nuclear coordinates = number of atoms ×3
- `naus`: number of steps along each coordinate (for a n-point central differences formula for the numerical derivatives, `naus`= $(n-1)$; n must be odd).
- `nummodes`: number of normal modes for which to calculate intensities in "intensities only" mode. In regular (full) SNF mode, `nummodes=0`.
- `cstep`: step size for the numerical derivatives, i.e., distortion of the Cartesian nuclear coordinates, in Bohr.

The names for the working and scratch directory will automatically be written to the restart file by SNFDC.

The next paragraph of the restart file contains two lines for every atom. The first line shows the number of the atom (in order of appearance in the file `coord`), and a step-flag for each distortion along one of the Cartesian coordinates of this atom (or along a normal mode in "intensities only mode"), i.e.,

$$+x, -x, +y, -y, +z, -z$$

for a 3-point formula, and

$$+2x, +x, -x, -2x, +2y, +y, -y, -2y, +2z, +z, -z, -2z$$

for a 5-point formula, etc. (in units of `cstep`). Possible values for the step flags are:

0	single point calculation has not been done yet for this point
−1	calculation for this point is running or aborted
1	calculation for this point is finished (or redundant because of the molecular symmetry)

The second line for each atom contains sets of 3·`naus` dashes for each step (or `naus` dashes per mode in "intensities only mode"), which are replaced by SNFDC by the name of the node that calculated the corresponding step. The control flags for the symmetry-redundant distortions are set equal to 1 by SNFDEFINE (which represents a point on the PES that has already been calculated) such that during the run of SNFDC these points will not be calculated.

Components of the dipole moments, gradients, and polarizabilities are stored in the data fields, which represents the largest part of the restart file. SNFDEFINE fills all these fields with zero entries. If the calculation of Raman intensities is switched off, there are no data fields for the polarizabilities in the restart file. In "intensities only" mode, no gradients are included in the restart file.

In case of TURBOMOLE single-point calculations, SNFDEFINE will modify the `control` file for the TURBOMOLE single point calculations. Be aware that the original `control` file, which contained all entries with the correct point group symmetry will be overwritten by the C_1 settings. This is necessary to prepare the calculations of dipole moments, gradients, and polarizabilities. The first time you run SNFDEFINE in a particular directory, it will create a write-protected backup copy of the original `<inputfile>` (the copy `<inputfile>.bak` is only created if it does not already exist).

Moreover, SNFDEFINE allows to set a lot of options for the calculation and the final output of SNF (see Sect. 6). All SNF-specific settings are stored in the file `snf_control`, which is copied onto every slave node during the SNFDC run.

You can also use SNFDEFINE to modify options. If calculations have already been performed, SNFDEFINE will not overwrite the `restart` file without creating a backup copy.

SNFDEFINE tries to generate the input files `USE_NODES` and `DCINPUT`, which are necessary to run SNFDC. Therefore, the scripts `choose_nodes` and `mkdcinput` must be available in `$PATH`, and the `ELIGIBLE` file must be present in the directory `$CIPROC`. While `DCINPUT` is a mandatory inputfile for SNFDC, `USE_NODES` can be used to construct a hostfile for the MPI version of SNFDC. This will be explained in Sect. 5.5.

Because all following single point calculations have to be performed in C_1 symmetry, the file `mos` (or `alpha` and `beta` in UHF/UKS cases) must contain MO starting vectors in C_1 symmetry. If the original symmetry of the molecule is higher, you can get the C_1 start-vector by using TURBOMOLE's `define`.

Summary:

⇒ If you have performed a geometry optimization with TURBOMOLE, DALTON, ADF, GAUSSIAN, or MOLPROjust type `snfdefine` in your working directory.

⇒ Select the options in the menus (see Sect. 6 for details).

⇒ Note possible warnings in the output of `snfdefine`.

⇒ If the symmetry mentioned in the original `control` file is higher than C_1, create a C_1 MO file (e.g., by using TURBOMOLE's `define` when using TURBOMOLE for single-points).

- Mandatory files for SNFDEFINE:
  `control, coord` (TURBOMOLE), `molecule.mol` and `molecule.dal` (DALTON), `adf.in` (ADF), `gaussian.com` (GAUSSIAN), or `molpro.in` (MOLPRO)

- Mandatory files and scripts for automatical generation of SNFDC input files by SNFDEFINE:
  `$CIPROC/ELIGIBLE`, `choose_nodes`, `mkdcinput`

- Files created by SNFDEFINE:
  `restart`, `snf_control`, `<inputfile>.bak`, `DCINPUT`, `USE_NODES`, `MACHINES`, `mpi.sub`

5.5 Snfdc

Some further preparations are necessary to start the parallelized calculation with Snfdc besides the generation of the `restart` file and the modification of the `control` file.

5.5.1 General input: DCINPUT

First of all, it is necessary to supply the separate input file `DCINPUT`. This file has the following structure:

```
DCPATH /usr/local/bin/snfdc
MYDIR /home/<user>/calculations/example
PREFIX TMP<user>%
TMPDIR /tmp
LOGDIR /home/<user>/calculations/example/log
```

with the variables

- `DCPATH`: path to the `snfdc` executable file (optional) (will be set to `$SNF_PATH/bin/snfdc` if you use `mkdcinput` to generate `DCINPUT`)
- `MYDIR`: your working directory (mandatory)
- `PREFIX`: prefix for logfiles and temporary files (optional)
- `TMPDIR`: directory for temporary files (including all temporary files of the single point calculations); must be available on every machine (optional)
- `LOGDIR`: directory for logfiles (optional)

If the optional variables are not specified in `DCINPUT`, Snfdc will try to use `$MYDIR/tmp` as `TMPDIR` and `$MYDIR/log` as `LOGDIR`. `PREFIX` will be set to "`TMP`".

The variables `DCPATH`, `TMPDIR`, and `LOGDIR` may be specified individually for some (or all) of the slave nodes. This is achieved by entries of the form

```
DCPATH(<nodename>) <pathname>
```

etc. Additional optional variables are

- `PROGDIR`: path to the Turbomole, Dalton, Gaussian, or Molpro executable files. Since Adf always requires to set the directory AdfBIN, Adf binaries will always be started as specified in the input file.
- `CHOOSENOPT`: options for the `choose_nodes` command when executed by Snfdc (or Snfdefine). Examples for options are:

- `-e <ELIGPATH>`: sets path to the `ELIGIBLE` file which shall be used in a calculation to `<ELIGPATH>` (default: `$CIPROC/ELIGIBLE`)
- `-ignore`: fake call that leaves the file `USE_NODES` unchanged; if `USE_NODES` does not exist, it creates an `USE_NODES` file without node entries (no reference to `ruptime` or `ELIGIBLE`).
- `-local`: creates a `USE_NODES` file that just contains the local machine with one CPU (no reference to `ruptime` or `ELIGIBLE`).
- `-nlocal <n>`: creates a `USE_NODES` file that just contains the local machine with `<n>` CPUs (no reference to `ruptime` or `ELIGIBLE`).

If you want to perform calculations with values for the variables `cstep` or `scfconv` which are out of the confidence interval (SNFDEFINE will inform you if this happens; see chapter 6.1.3) the entry

```
CRAP_OK yes
```

must be added to `DCINPUT`.

To create a standard inputfile, you can use the script `mkdcinput`, which is automatically executed by SNFDEFINE. This file should be executed in the working directory. The settings for `DCPATH` and `TMPDIR` in this script must be adapted to the local settings before using it.

5.5.2 Preparations for the MPI versions

The MPI version will run on a static set of nodes. It is therefore particularly suited for queuing systems because in this case no load-balancing is required.

If run inside a queueing system (e.g., PBS), the `mpirun` command supplied with OpenMPI is able to determine the list of nodes and therefore no further setup is required. If, however, the node list is not determined automatically you must write some kind of wrapper script yourself. The number of combinations of existing MPI implementations and queuing systems is too large to handle this in an easy general way.

If the MPI version shall be used outside a queuing system, the list of nodes must be created manually. One can then submit the job using a syntax of the following form:

```
mpirun --hostfile MACHINES /path/to/sndfc &> snfdc.out < /dev/null.
```

In this example command, `MACHINES` denotes the hostfile.

5.5.3 SNFDC calculations

SNFDC will first read the most essential input files specifying the settings for the program to be used in the single-point calculations (that is `control` and `coord` in case of TURBOMOLE calculations, for example), as well as the `restart` file to determine the points which

have to be calculated. These points are then distributed to the processors chosen in the file `USE_NODES`. Temporary directories are created in the `TEMPDIR` directories of the slaves. A logfile is generated for every slave process. These logfiles—and also the file `fort.41`, which is created in the working directory (`MYDIR`)— can be used to get information about errors which occur during the run of SNFDC in the master process and the slave processes.

Most important to observe the progress of the calculation is the file `TMPdcstat` which contains information about the status of each slave process. It consists of 11 columns:

```
 2   1 host1  vbsy   0  90005  1 rdy   1   1 results sent
 3   2 host2         0  90005  1 rdy   2   2 results sent
11  10 host3         0    103  0 run  16  16  968:15 dscf
13  12 host4         0    102  0 run  18  18 1227:31 escf
```

The rows contain the following entries:

column	description
1	task ID
2	continuous numbering
3	node name
4	entry `vbsy` indicates a "very busy" machine
5	process ID (0 for finished processes)
6	cpu-load times 100, or 90005 for finished calculations, or 70005 for trouble in TURBOMOLE programs, or 80005 for wrong results.
7	number of finished single point calculations on this node
8	status: `rdy` = ready for next step, `run` = running calculation, `wait` = node waiting
9	step number
10	step number (redundant)
11	cpu time and name of running program or message

As already indicated by the name of the data file, SNFDC has a restart facility, such that any partly completed `restart` file can be supplied in an SNFDC run if the program

has been (unexpectedly) terminated before.

Summary:

⇒ If the files `restart` and `snf_control` have been prepared by SNFDEFINE, generate the input file `DCINPUT` for SNFDC using the script `mkdcinput` (this is done automatically if SNFDEFINE has been used before).

⇒ Create the file `USE_NODES` using the script `choose_nodes` (this is also done automatically by SNFDEFINE).

⇒ Type `snfdc` to start the parallelized calculation.

⇒ You may check the progress of the calculation via the status file `TMPdcstat`.

- Mandatory files for SNFDC:
  `control`, `coord`, `basis`, `mos` [or `alpha`, `beta`] (TURBOMOLE),
  `molecule.mol` and `molecule.dal` (DALTON),
  `gaussian.com` (GAUSSIAN),
  `adf.in` (ADF), or
  `molpro.in` (MOLPRO)
  `restart`, `DCINPUT`;
  MPI versions: `MACHINES` (for `mpirun`)

- Mandatory scripts for SNFDC:
  `choose_nodes`, `dc_killtask`, `ruptime`

- Files created by SNFDC in the working directory:
  `TMPdcstat, fort.41`

- Files created by SNFDC in the log directory:
  `PREFIXdclog.XXXXXXX`

5.6 SNF

If all required data are available in the `restart` file, the evaluation can be performed by executing `snf`. SNF will then create several output files (not all of these output files are necessarily created, depending on the actual settings in the `snf_control` file; see Sect. 6.3).

- `snf.out`: main output file (see description below)
- `line_spectrum.dat`: list of vibrational frequencies, IR and Raman intensities for a graphical representation (using, for instance, GNUPLOT)
- `gauss/lorentz_spectrum.dat`: data for a model spectrum with peaks in shape of Gauss/Lorentz-functions (can also be plotted by GNUPLOT)
- `ir_line.gpin`: GNUPLOT input files for the generation of EPS output of infrared spectra
- `raman_line.gpin`: GNUPLOT input files for the generation of EPS output of Raman spectra
- `ir_line.eps, ir_gauss/lorentz.eps, raman_line.eps, raman_gauss/lorentz.eps`: Spectra plots (EPS format)
- `g98.out`: GAUSSIAN98 fake output for graphical representation of normal modes using, e.g., MOLDEN (optional)
- `mopac.out`: MOPAC fake output for graphical representation of normal modes using, e.g., MOLDEN (optional)
- `xmol.XYZ.out`: Output readable (as XYZ) by XMOL [63] for graphical representation of normal modes (optional; it seems unclear if XMOL is still maintained)

Note that `gnuplot` must be found in `$PATH`, otherwise the automatical generation of the EPS files by SNF will fail. However, any plotting program can be used to generate plots from the `.dat` files manually.

The main output file `snf.out` briefly summarizes details concerning the calculation, i.e., the options from the `snf_control` file and settings from the `restart` file, and the Cartesian nuclear coordinates of the molecule. The results of a symmetry check for the molecule and for the entries in the Cartesian Hessian matrix are shown. Moreover, a charge decomposition according to Cioslowski [64] may be performed (however, note that it has been found that these charges cannot even qualitatively reproduce the quadrupole moment of certain molecules and are thus not suited to analyze the electron density or calculate intermolecular interaction energies [65]).

The next part of the `snf.out` file contains information about the vibrational modes. Wavenumbers as well as translational and rotational percentages of the modes are shown for each irrep. It follows a second evaluation, in which the translational and rotational contributions are eliminated before diagonalizing the Hessian matrix, such that this evaluation should yield more accurate results. Infrared and (if selected by the user) Raman/VROA intensities are given in this part of the analysis.

The next paragraph contains moments of inertia, the zero point kinetic energy (ZPE) and thermochemical data, followed by the mass-weighted normal coordinates and a list of all output files created. Further information can be obtained by choosing the corresponding output options in the options menu of SNFDEFINE (see Sect. 6).

Summary:

⇒ If all data are collected by SNFDC and written to the `restart` file, run `snf` in your working directory.

⇒ You may modify the output options by SNFDEFINE or by editing the file `snf_control` and rerun SNF.

⇒ Note possible warnings in the output of SNF.

- Mandatory files for SNF:
  `coord`, `control` (TURBOMOLE),
  `molecule.mol`, `molecule.dal` (DALTON),
  `gaussian.com` (GAUSSIAN),
  `adf.in` (ADF),
  `molpro.in` (MOLPRO),
  `restart, snf_control`

- Mandatory program for automatic generation of spectra plots: GNUPLOT.

- Files created by SNFDEFINE:
  `snf.out`, `line_spectrum.dat`, `gauss_spectrum.dat`, `ir_line.gpin`, `ir_gauss.gpin`, `raman_line.gpin`, `raman_gauss.gpin`, `ir_line.eps`, `ir_gauss.eps`, `raman_line.eps`, `raman_gauss.eps`, `g98.out`, `mopac.out`, `xmol.XYZ.out`

6. SNF options

Most of the options for calculations with SNF can be selected using SNFDEFINE. Some additional options, which are less important or only for debugging purposes, can be chosen by special keywords in the file `snf_control`. SNFDEFINE is an interactive program, and short explanations are provided for all menus and options. It works in essentially the same way like DEFINE of the TURBOMOLE package.

The aim of this chapter is to give detailed information about the meaning of these options. The menus are explained in order of appearance in SNFDEFINE.

SNFDEFINE can, of course, be applied to change program options of existing input files. The data from existing `restart` files will be saved. If only program options which have no influence on the electronic data (gradients, dipole moments, and polarizabilities), like output options or non-standard isotopes, are modified, one can rerun SNF without rerunning the data collection process.

6.1 Input Options

6.1.1 Selection of input files

When SNFDEFINE is started, it will check for old `snf_control` files. If such a file is found, you can either accept that file as input for SNF or modify it. In the next menu, you can enter the name of an alternative input file, which will be used as a template for the current calculation.

SNFDEFINE furthermore allows to select the program that should be used for the single point calculations. Depending on whether it finds a mandatory file for one of the supported quantum chemical program packages (see Section 5.5.3), SNFDEFINE will suggest the selection of this program package for the single-point calculations. However, when DALTON is selected for the single point calculations, it is possible to read input files either in TURBOMOLE or in DALTON format (default: DALTON; cf. Sect. 5.2.2). In this case, it is also necessary to specify the type of the single point calculations in an interactive input menu in SNFDEFINE.

Furthermore, the program will read the molecular input files, check the point group given in `control` (if present), and print some information about the molecule.

The following main menu contains several submenus, which allow to set further input options. Moreover, it allows to invoke the automatic symmetry detection routines of SNF.

6.1.2 Selection of isotopes

The isotope menu can be utilized to select non-standard isotopes. By default, SNFDEFINE will choose the most abundant isotopes. However, the electronic data are—within the Born–Oppenheimer-approximation—independent of the nuclear masses, such that only one `restart` file and one SNFDC run are necessary for all evaluations with different isotopes.

The selection of isotopes changes the molecular point group symmetry. Thus, that part of the calculation which depends on the nuclear masses has to be performed using the symmetry of the molecule with modified isotopes, while the generation of the symmetry-redundant electronic data is done using the point group symmetry of the original molecule.

The present version of SNF requires that the orientation of the molecule is the same in both parts of the calculation. This causes the problem that not the full symmetry of the molecule with modified isotopes can be applied. E.g., the standard orientation of benzene is in the xy plane, such that the z axis is the principal axis of rotation. If one ^{12}C atom is replaced by a ^{13}C isotope, the molecular point group is C_{2v}. But the principal axis of rotation is no longer the z-axis, but maybe the y-axis, such that SNF will detect the point group C_s instead of C_{2v}.

However, the effect of the symmetry for this part of the evaluation is of minor importance. It affects the assignment of the symmetry races to the peaks in the spectra and perhaps slightly increases the computational time for the evaluation (which is of the order of some seconds to some minutes). But the full symmetry can still be used for the most time-consuming part, the calculation of the gradients, dipole moments, and polarizabilities.

6.1.3 Options for numerical derivatives

The numerical derivative menu allows to select the number of grid points used for the differentiation, and to set the `cstep` value, which characterizes the step size in Bohr.

For the number of grid points, one may choose 3, 5, or 7. Note that the 5-point central differences formula will double the computational cost of the data collecting process compared to the 3-point formula. It is usually not necessary to use the 5- (or even 7-) point central differences formula, since the accuracy of the 3-point formula is in most cases sufficient. A higher number of points is neither recommended nor implemented in the program, since the gain in accuracy can be neglected in almost all cases. If higher accuracy should be achieved, it is more helpful to apply Richardson extrapolation [66].

As will be shown in chapter 11, the `cstep` value should be $0.01 \leq$ `cstep` ≤ 0.05, since smaller values affect the numerical stability of the differentiation, while larger values lead to a failure of the harmonic approximation. If you would like to use values beyond this range, you have to set the `CRAP_OK yes` option in the SNFDC input file `DCINPUT` (see Sect. 5.5).

6.1.4 Spectrum settings

In the spectrum settings menu one can modify the upper and lower bound of the infrared and Raman spectra. Additionally, one can choose the half-width of the peaks in shape of Gauss- or Lorentz-type functions in the Gauss/Lorentz spectra, and the number of points for their graphical representation. It is also possible to specify a scaling factor for the frequencies (see, e.g., [67] and references cited therein for scaling factors).

The Gauss spectra are plotted using the normalized line shape function

$$g_G(\tilde{\nu}) = \frac{1}{\sqrt{2\pi\gamma^2}} \cdot \exp\left(-\frac{(\tilde{\nu}_0 - \tilde{\nu})^2}{2\gamma^2}\right) \tag{6.1}$$

with the half-width δ_G,

$$\delta_G = 2\gamma\sqrt{2\ln 2}, \tag{6.2}$$

and resembles inhomogeneous line-broadening (e.g., Doppler broadening). For the Lorentz spectra, the normalized line shape function

$$g_L(\tilde{\nu}) = \frac{\delta_L/(2\pi)}{(\tilde{\nu}_0 - \tilde{\nu})^2 + (\delta_L/2)^2}, \tag{6.3}$$

with the half-width δ_L, and should be used if homogeneous line broadening (e.g., natural line broadening) is dominant.

6.1.5 Thermochemical settings

The thermochemistry menu can be used to select temperature and pressure for the thermochemical output.

6.1.6 Excited-state calculation menu

This section allows to change settings for the calculation of excited states employing the TURBOMOLE `dscf/ridft` and `egrad` modules or the DALTON CASSCF module. In case of Hartree–Fock calculations, either the RPA or the CIS method can be chosen for the calculation of excited state frequencies. Furthermore, the number of excited states to be treated in `egrad`, the excited state for which the frequencies shall be calculated, and the spin state of the excited state can be selected. Note, however, the warnings mentioned in Section 5.3.

6.1.7 Intensity-only mode

It is possible to calculate intensities for selected precalculated normal modes, which may have been obtained from a previous SNF run (for example, if the computationally more

demanding Raman intensities shall only be calculated for a few modes), or from a mode-tracking calculation with AKIRA. This option can be switched on by typing `int on` in the appropriate menu, and then reading in normal modes from a SNF, AKIRA, or GAUSSIAN98-style output file. All modes in the file will be read in by default; the desired modes can then be selected or unselected with the (self-explanatory) commands in the menu.

6.1.8 Raman settings

There are several options concerning the Raman spectra, which can be found in the Raman menu. First of all, the calculation of Raman intensities must be switched on by the user. Since this is very time-consuming the option is switched off by default. If the calculation of Raman intensities has been selected, you may choose between the calculation using static polarizabilities (default), or dynamic polarizabilities. If dynamic polarizabilities are chosen, one can select up to 10 different frequencies for the calculation of the Raman intensities. These must be given either as wavelengths (in nm) or as angular frequencies (in a.u., i.e., hartree/$\hbar$). Default is the wavelength of an argon ion laser (514.5 nm).

It is also possible to change the default `rpaconv` parameter for the `escf` run, which sets the convergency criterion for the residual vector (see TURBOMOLE manual). Further options concern the type of data to be plotted in the Raman spectrum output (Raman activities, differential scattering cross sections for different experimental setups) and the scattering angle for the calculation of scattering cross sections.

6.1.9 VROA settings

The VROA menu is analogous to the Raman menu. The only difference is that the output is written to `snf.out` in DALTON format, i.e., intensity differences and CIDs are printed by default for scattering angles of 0° (forward scattering), 90°, and 180° (backscattering). For 90° scattering, intensity differences and CIDs are given for both parallel and perpendicular relative polarization of the incoming and the outgoing laser beam. In order to plot VROA spectra, you may use the program SPECPLOT which is distributed along with the SNF package (see `$SNF_PATH/utilities/specplot`). It reads VROA (and also Raman) intensities from `snf.out` and generates GNUPLOT input files according to the settings chosen by the user

For a summary of the theoretical background of VROA, see Refs. [7] and [68], and for the first accounts on theoretical and experimental aspects, Refs. [69–73].

6.1.10 External matrices settings

To be able to calculate spectra from the property tensor derivatives composed employing the Cartesian Tensor Transferm Method (CTTM) SNF needs to read in external matrices.

The settings are accessible only by manually editing the corresponding keywords in the `snf_control` file (see also section 6.3).

The working directory has to contain only the `control` file from the single point calculations, the `coord` file of the whole molecule, the external matrices and a `snf_control` file. As soon as one necessary entity is not given externally also a restart file is needed. If all keywords are set to `none`, which is the default value, then a normal SNF run is performed. The CTTM is switched on by defining one or more external matrix files. To obtain the frequencies by CTTM the keyword `$hessian_mat` has to be set. For IR or Raman intensities the corresponding keywords are `$dipol_mat` and `$pol_mat`. For ROA intensities `$pol_mat`, `$pol_vel_mat`, `$aten_mat`, `$gtenao_mat`, `$gtenlao_mat` have to be defined.

If the Hessian matrix and also all necessary property tensor derivatives are available as external matrix files then `snf` can be run without a restart file. It is also possible to combine external property tensor derivatives with a hessian matrix from a restart file. Also the other way round is possible. Property tensor derivatives from a restart file can be combined with a external hessian matrix.

All the external matrices have to be in the matrix market file format [74] (`*.mtx`). They are created in the correct format by the CTTM script following the composition rules given in the input file.

6.2 Output options

Many output options are available in SNFDEFINE. Some output options are hidden to keep the output options menu clear. They can be displayed by the command "`all`". Alternatively, these options can be (de-)activated by the corresponding keyword in the file `snf_control` (see Sect. 6.3).

Options available in the standard output menu are:

`iat <on/off>` turn on/off output of information about symmetry-redundant atoms

`idt <on/off>` turn on/off output of information about symmetry-redundant distortions of the non-redundant atoms

`hess <on/off>` turn on/off output of the Cartesian Hessian

`sumdeg <on/off>` turn on/off summation of intensities over degenerate modes

`gau, mopac, xmol <on/off>` turn on/off output of normal modes in format of GAUSSIAN98, MOPAC, or as XYZ format readable by XMOL

`atcontrib <on/off>` turn on/off output of atomic contributions to vibrational intensities (as `.csv` files)

`tmpcl <on/off>` turn on/off removal of temporary directories for single point calculations

`logcl <on/off>` turn on/off removal of logfiles for slave processes

`elig <PATH>` specify an alternative `ELIGIBLE` file

`bak` create backup copy of the original `mos` file (to run SNFDC you need starting MO vectors in C_1 symmetry; if you use TURBOMOLE's `define` to get these MO vectors, the original `mos` file will be overwritten.)

6.3 Additional keywords

`dalmw <val>` sets the scratch memory size in Mwords in case of DALTON calculations. Minimum is `<val>` = 800000.

`dtype [option]` selects the type of numerical differentiation to be used. CAUTION: this is a test option, which works only for two-point formulae. Possible values of `[option]` are: `centr` (central differences, default), `fward` (forward differences), `bward` (backward differences). If forward or backward differences are selected, SNF tries to read the `gradient` file for the equilibrium structure. If the gradient can be read (only TURBOMOLE format at the moment), the numerical precision of forward/backward differences can be expected to be at least one order of magnitude less accurate than central differences (numerical errors up to $20 - 30\ \mathrm{cm}^{-1}$). If the gradient cannot be read, gradient components of `0.0d0` are assumed, which can usually give rise to numerical errors of $> 100\,\mathrm{cm}^{-1}$ (depends on the optimization thresholds). NOTE: FORWARD AND BACKWARD DIFFERENCES SHOULD NEVER BE USED IN PRODUCTION CALCULATIONS. THEY CAN ONLY GIVE A FIRST IMPRESSION OF THE VIBRATIONAL FREQUENCIES OF A MOLECULE.

`logdirclean` if present in the control file the logfiles are deleted after each run. Note that setting this keyword is not recommended since the logfiles usually contain valuable information if a calculation failed for some reason.

`maxsym <on/off>` turns on/off the determination of symmetry-redundant distortions of non-redundant atoms. If `maxsym off` is chosen, only the step flags of distortions of symmetry-redundant atoms will be set equal to 1 in the `restart` file (default: on).

`override_sigma_rot <integer>` Use this value to explicitly specify the symmetry number (σ) used in the thermodynamics calculations. If this keyword is missing (or the value is '0'), the symmetry number is determined by the point group of the molecule.
(default: 0)

`print_cio <on/off>` turn on/off charge decomposition according to Cioslowski [64] (default: on)

`print_dip <on/off>` turn on/off output of dipole moment derivatives (default: off)

`print_force <on/off>` turn on/off output of force constant matrix (default: off)

`print_pol <on/off>` turn on/off output of polarizability derivatives (default: off)

`print_symmetry <on/off>` turn on/off symmetry information (default: off)

`print_thermo <on/off>` turn on/off detailed thermochemical output (default: on)

`print_transrot <on/off>` turn on/off output of translational and vibrational contributions (default: on)

`print_zpe <on/off>` turn on/off output of zero-point kinetic energy and moments of inertia (default: on)

`sleeptime <val>` sets the time to wait for the master process between two subsequent checks of the slave processes. Default: 5. Smaller values decrease the overhead, in particular in fast calculations, larger values can be helpful to decrease the network traffic when running the master job via a network file system, in particular for large calculations with long single-point calculations. The units are milliseconds for the MPI version. For further explanation look at the comments in `parstuff_sleep.c`. The serial version neither needs nor uses this parameter. If you run very large calculations (thousand or more nodes) you should set the sleeptime to 0 for the MPI version to be able to handle all MPI messages in a timely manner.

`tmpdirclean` if present in the control file the temporary directories where the calculations are performed are deleted after each run.

`theta <val>` sets the scattering angle θ (in degree) for the calculation of Raman scattering cross sections, if their calculation has been selected in the Raman menu (default: 90°).

`total electronic energy <val>` sets the total electronic energy for the optimized structure equal to `<val>`. This can be done if the automatical procedure to read this value from the files `job.last` (TURBOMOLE) or `molecule.out` (DALTON) by SNFDEFINE failed. It is only necessary for parts of the thermochemical analysis.

`total spin <val>` analogously for the total spin value

`xmolsc1 <val>` sets the scaling factor for normal modes in the XMOL output file equal to `<val>`

`hessian_mat` defines an external matrix file for the Hessian matrix (default: `none`)

`dipol_mat` defines an external matrix file for the dipole derivatives (default: `none`)

`pol_mat` defines an external matrix file for the polarizability derivatives (default: `none`)

`pol_vel_mat` defines an external matrix file for the polarizability derivatives (velocity representation) (default: `none`)

`aten_mat` defines an external matrix file for the **A** tensor derivatives (default: `none`)

`gtenao_mat` defines an external matrix file for the **G** tensor derivatives (default: `none`)

`gtenlao_mat` defines an external matrix file for the **G** tensor derivatives (London atomic orbitals) (default: `none`)

7. Visualization of results

There are several programs that can be applied to visualize the results of the SNF calculation. SNF only includes our own tool `specplot`[1]. A myriad of other tools are available. Unfortunately, most major Linux distributions are lacking these programs and one therefore has to manually download and install them. For Gentoo however, many such program packages are included and can thus easily be installed.

For the conversion of images generated by the visualization programs the tools from the `imagemagick` package will come in handy. For videos, you may also have a look at `mencoder` which is part of the `mplayer` package. Both packages are included in any major Linux distribution and can also be downloaded seperately from the Internet.

7.1 Spectra plots

`gnuplot` generates the following spectra plots in EPS format:

- `ir_line_spectrum.eps`,
- `ir_gauss_spectrum.eps`,
- `ir_lorentz_spectrum.eps`,
- `raman_line_spectrum.eps`,
- `raman_gauss_spectrum.eps`, and
- `raman_lorentz_spectrum.eps`.

Standard tools like `gv` can be used to display these plots, others—like `convert` from the `imagemagick` package—can convert the format into various others. For Raman and backscattering VROA spectra, the tool `specplot` coming with SNF can be employed in order to generate and visualize `gnuplot` files with the desired settings (see Section 18.2 for further details).

[1] You need to supply `--enable-specplot` to `configure` to build this tool.

7.2 Normal modes

The following list includes a few examples of programs which can read and display normal mode output by SNF:

`molden`: can read the output files `g98.out` or `mopac.out` created by SNF. Normal mode output can be generated as a series of GIF images which can be animated by tools like `animate` or `xanim`. They can also be converted to animated GIF using programs like `whirlgif` or `gifsicle`. Another possible output are Postscript files, in which the normal modes are indicated by vectors. Molden can be obtained from `http://www.caos.kun.nl/~schaft/molden/molden.html`.

`xmol`: can read the `xmol.XYZ.out` files created by SNF. Normal modes can be displayed and exported as vectors (Postscript).

`jmol`: can read the SNF output files `g98.out` and `xmol.XYZ.out`. Output can be generated using vectors. Additionally, it is possible to create a series of pictures (e.g., PNG), which can then be animated using `animate` or `xanim`. `jmol` can be obtained from `http://jmol.sourceforge.net/`.

8. Thermochemistry

Thermodynamical functions are calculated according to the statistical thermodynamics of a canonical ensemble of an ideal gas. Rotational contributions are treated classically. Electronic contributions are taken into account as spin-only values to the entropy in UHF/UKS cases.

The calculations are based on the following relationships[1]:

- **Enthalpy** $H = H^t + H^r + H^{vib}$

  - o Translation: $H^t = 5/2RT$
    R is the ideal gas constant, and T the temperature.
  - o Rotation:
    non-linear molecules: $H^r = 3/2RT$
    linear molecules: $H^r = RT$
  - o Vibration:
    $$H^v = RT \sum_i \frac{h\nu_i}{kT} \left\{ \frac{1}{2} + \left[\exp\left(\frac{h\nu_i}{kT}\right) - 1 \right]^{-1} \right\}$$
    h is Planck's constant, ν_i the frequency of the i-th normal mode, and k the Boltzmann constant. Note that the zero point vibrational energy (ZPVE) is included. The sum runs over all normal modes of the molecule.
  - o Electronical contributions: none

- **Entropy** $S = S^t + S^r + S^{vib}$ **and partition functions**

  - o Translation
    $$S^t = R(ln(z^t) + 5/2) \ , \ z^t = (kT)^{5/2}/(ph^3)\,(2\pi m)^{3/2}$$
    m is the molecular mass, and p the pressure.

[1]See Ref. [75] for details, but note that it contains some typos in the formulas relevant in our context.

- Rotation
  non-linear molecules:

$$S^r = R(ln(z^r) + 3/2)\ ,\ z^r = \frac{\sqrt{\pi}}{\sigma}\sqrt{\frac{T^3}{T_1T_2T_3}}$$

  linear molecules:

$$S^r = R(ln(z^r) + 1)\ ,\ z^r = \frac{\sqrt{\pi}}{\sigma}\sqrt{\frac{T^3}{T_1T_2T_3}}$$

$$T_i = \frac{h^2}{8k\pi^2 J_i}$$

  σ is the symmetry number, which is connected to the point group of the molecule. Symmetry numbers for some common point groups can be found in reference [76]. J_i is the moment of inertia.

- Vibration

$$S^{vib} = R\sum_i \left(\frac{h\nu_i}{kT}\left[\exp\left(\frac{h\nu_i}{kT}\right) - 1\right]^{-1} - \ln\left[1 - \exp\left(-\frac{h\nu_i}{kT}\right)\right]\right)$$

  Partition function with reference to the bottom of the potential well:

$$z^{vib,bot} = \prod_i \exp\left(-\frac{h\nu_i}{2kT}\right)\left[1 - \exp\left(-\frac{h\nu_i}{kT}\right)\right]^{-1}$$

  Partition function with reference to the vibrational ground state energy:

$$z^{vib,v=0} = \prod_i \left[1 - \exp\left(-\frac{h\nu_i}{kT}\right)\right]^{-1}$$

- Electronical contributions:

$$S^e = R\ln(g)$$

  g is the spin degeneracy of the ground state (i.e., only the electronic ground state is assumed to contribute to the entropy; excited electronic states must be sufficiently higher in energy).

- **Heat capacity** $C_v = C_V^t + C_V^r + C_V^{vib}$

  - Translation: $C_V^t = 3/2R$

- o Rotation:
  non-linear molecules: $C_V^r = 3/2R$
  linear molecules: $C_V^r = R$
- o Vibration

$$C_V^{vib} = R \sum_i \left(\frac{h\nu_i}{kT}\right)^2 \exp\left(\frac{h\nu_i}{kT}\right) \left[\exp\left(\frac{h\nu_i}{kT}\right) - 1\right]^{-2}$$

- o Electronical contributions: none

- **Gibbs free energy** $G = H - TS$

9. Cartesian Tensor Transfer Method

Instead of calculating all necessary property tensor derivatives with the program SNF one can also employ the Cartesian Tensor Transfer Method (CTTM) [77] to obtain these quantities. The CTTM is an approximate method to calculate spectra of very large molecules. Although the method yields good results for the frequencies and reasonable spectra in case of IR and Raman, the calculation of ROA spectra by this method has limitations. The CTTM approximates a spectrum by composing the property tensor derivatives from calculations of smaller fragments. It can be employed in any case, where the full calculation is not feasible, keeping in mind that the approximation can introduce severe errors depending on the details of the fragmentation [28].

The CTTM works as follows: Instead of calculating the full Hessian matrix and the full property tensor derivatives as in a normal SNFDC run, the quantatities are calculated only for the smaller fragments. The results of the smaller subsystem calculations are then combined to yield an approximation of the Hessian matrix and the property tensor derivatives of the full molecule. They are then processed by the SNFF routines of the MOVIPAC package as in every other calculation.

Since the choice of the fragments and the way they are combined heavily influences the result of the calculation [28], there is no automatic procedure implemented in the MOVIPAC package. Instead the SNF program allows to read in Hessian matrices and property tensor derivatives from external files (see section 6.1.10). The extraction and composition of the raw data from the subsystem calculations is done by the script `run_cttm`.

To perform a frequency analysis and spectrum calculation employing CTTM the following steps have to be done. Up to now the CTTM is only available for calculations employing the Turbomole package.

1. Calculate the property tensor derivatives for the chosen fragments with SNF. To do so one has to perform a SNF calculation with the appropiate options (IR for the dipoles, Raman for the polarizabilities etc.) and with the print options for the property tensor derivatives switched on. It is important that the necessary quantities are present in the output file of SNF, so that they can be obtained from there in the next step.

2. Extract the property tensor derivatives from the output files of the fragment calculations and compose them to obtain those of the full molecule running the `run_cttm` script.

3. Calculate the desired spectra with SNF from these quantities as is described in section about the usage of external matrices (section 6.1.10).

The second step is the actual application of the CTTM. The parameters are provided to the `run_cttm` script in an `input` file. See the `example/cttm` directory for a sample `input` file (together with all necessary files for a complete run). From the full molecule only the `coord` file and a complete connectivity table is needed. For each fragment a `coord` file and a SNF output file with the printed property tensor derivatives are needed. The last and most important step is to define how the fragments overlap the full molecule. For this purpose the `input` file has an `$overlap` section. In the first column the atom IDs of the full molecule are given. In the second row the corresponding atoms of the first fragment, in the third the ones of the second fragment and so on. If one defines more than one overlap for a certain atom then the script chooses the one which fits best. For algorithmic details we refer to Ref. 28.

The `run_cttm` script has several options in order to define which quantities are desired. They are listed below.

```
Usage: merge_matrices [options] input-file

Options:
        -h, --help      show this message
        -i, --ir        activate creation of matrices necessary for
                        IR spectra (default: on)
        -n, --no-ir     deactivate creation of matrices necessary for
                        IR spectra
        -r, --raman     activate creation of matrices necessary for
                        raman spectra (default: off)
        -v, --vroa      activate VROA creation (default: off)
```

The `run_cttm` scripts generates the desired composed matrices as matrix market format [74] files (`*.mtx`), which can be directly utilized to generate the spectra (see section 6.1.10). It also provides a log file with all the rotation matrices and the RMS values for each atom pair. The log file is named like the `input` file with a trailing `.out`.

Since the `run_cttm` script is written in the Python programming language it is easy to adjust to the user's needs.

10. Localizing Normal Modes

10.1 General

The LocVib tools, which are included in the MoViPac package, provide a number of features for analyzing calculated vibrational spectra in terms of localized modes. For details on the theoretical background, see

Christoph R. Jacob, Markus Reiher. Localizing normal modes in large molecules. *J. Chem. Phys.*, **130** (2009) 084106.

Please also cite this reference in publications using the LocVib tools. The most recent versions of LocVib will be made available at http://www.christophjacob.eu/locvib.php. On this website, a detailed documentation describing all the advanced functionality of LocVib will also be available soon.

10.2 Installation

The LocVib tools are distributed as a Python library. To use them, you have to include the subdirectory "LocVib/VibTools" in your PYTHONPATH environment variable.

LocVib relies on additional Python packages, that have to be installed on your system:

- the Openbabel package (http://openbabel.org), including its Python bindings
- the Matplotlib package (http://matplotlib.sourceforge.net)

These are available as standard packages for most Linux distributions.

The LocVib tools can then be used in Python scripts to perform an analysis of vibrational spectra in terms of localized modes. A few example scripts, that can serve as starting point for more complicated applications, are provided in the subdirectory "LocVib/examples". These will be used to explain the most important features in the following.

10.3 Reading Results from Snf or Akira

For using the LocVib tools in a Python script, one first has to to import them with

```
import VibTools
```

The first step is then to read in the results of a previous SNF or AKIRA calculation; this can be done with the classes SNFResults and AKIRAResults, respectively. For SNF, one can use

```
res = VibTools.SNFResults()
res.read()
```

This requires the files coord (with the molecular coordinates in Turbomole format), the "restart" file produced by SNFDC, and the output file from SNF, snf.out, to the present in the current directory. Alternative names and locations of these files can be passed to SNFResults. For more details and additional options, see the source code in PySNF.py.

Similarly, AKIRA results can be read with

```
res = VibTools.AKIRAResults()
res.read()
```

This requires the files coord and akira_iterations.out. For more details and additional options, see the source code in PyAKIRA.py.

The resulting instances of the result files then give access to the normal modes, their frequencies, and the calculated vibrational intensities. See the source code in Results.py for more details.

10.4 Assigning Normal Modes to Bands

For performing an analysis in terms of localized modes, one has to assign the normal modes to different vibrational bands. To perform this assignment, it is useful to consider the contributions of different groups of atoms to each of the normal modes. An example of a script to assist with such an assignment is included as "1_composition.py".

This script prints a list of the normal modes and for each mode lists the contributions of different atom types. For instance, for the included example of an $(Ala)_{10}$ helix, one obtains the output:

```
                                                NH     CO    CA    HA   CHB
[...]
236 1500.0     54.6596     0.7737    0.0046   76.0   17.7   2.2   1.4   2.7
237 1503.7    189.3569     1.8920    0.0341   74.5   17.7   2.8   2.3   2.7
238 1506.4    217.9075     7.7164   -0.0585   75.1   16.3   2.8   2.7   3.2
239 1508.5    407.3909     6.8705   -0.0143   76.4   18.1   2.3   1.1   2.2
240 1617.6     53.4093     8.3601   -0.0152   98.4    0.3   0.8   0.2   0.3
241 1644.0     22.0081     1.8999    0.0294    3.0   94.1   0.9   1.5   0.5
242 1650.0    322.4753    17.5984    0.0226    3.3   94.2   0.6   1.4   0.5
243 1651.1   1361.5743   112.2062    0.0108    5.1   93.3   0.2   0.8   0.5
244 1656.1    232.1278     2.5841   -0.0374    3.5   94.3   0.5   1.3   0.4
```

```
245 1657.2     41.6081     1.2601   -0.0010    4.4   93.9   0.3   1.0   0.4
246 1663.7     86.5379     1.8897   -0.0026    3.3   94.5   0.6   1.1   0.4
247 1668.3    247.8201     8.6080    0.0088    2.4   96.2   0.5   0.4   0.5
248 1670.0    290.1721    24.3792    0.0023    4.0   94.4   0.3   0.9   0.4
249 1677.3    217.4265     9.0573    0.0112    3.0   95.6   0.4   0.5   0.4
250 1736.0    326.1756    11.9512    0.0017    2.5   96.7   0.4   0.0   0.3
[...]
```

Thus, based on the atomic contributions and the vibrational wavenumbers listed in the second column one can notice that modes 236 to 240 are similar vibrations and form one band (mode 250 appears at significantly higher wavenumber and is, therefore, not included). These are the amide I vibrations, which can be further analyzed now.

The assignment of the atom types is, of course, dependent on the class of molecules considered. For polypeptides, these are assigned with the help of Openbabel, and several different collections of atom types are provided. See the source code in Molecule.py for more details.

10.5 Localization of Normal Modes

After identifying which modes contribute to one band, these can be transformed to localized modes for further analysis. An example of this step given in the script "2_locmodes.py".

First, the relevant subset of the normal modes is selected with

```
modes = res.modes.get_subset(range(241,250))
```

The functionality for localizing normal modes is provided by the class LocVib. An instance of this class is created with

```
lv = VibTools.LocVib(modes, 'PM')
```

where 'PM' selects the atomic-contribution localization criterion (in analogy to the Pikek-Mezey orbital localization). Alternatively, 'Boys' can be used to chose the distance-criterion for the localization instead. Then,

```
lv.localize()
```

performs the iterative localization. If convergence problems are encountered in this step, the set of normal modes has probably been poorly chosen and the assignment should be revisted. Finally, in polypeptides the localized modes can be sorted by residue with

```
lv.sort_by_residue()
```

After the localization has been performed, the localized modes are available in lv.locmodes. For instance, they can be saved to a g98-type file for visual inspection with

```
lv.locmodes.write_g98out(filename="locmodes-amide1.out")
```

The example script also demonstrates how the composition on the localized modes can be analyzed and how intensities of localized modes can be obtained. For more details, see the source code of "2_locmodes.py".

10.6 Coupling Constants

After the localization has been performed for a set of normal modes, coupling constants can also be extracted. An example is provided in the script "3_couplings.py". The sign of the coupling constants depends on the phase of the localized modes. With

```
lv.adjust_signs()
```

the phase of the localized modes is adjusted such that the nearest-neighbor vibrational coupling constants are positive. However, that does not necessarily imply that the phase is chosen consistently for all localized modes. This can at present only be ensured by visually inspecting the localized modes and, if necessary, inverting their signs with `lv.invert_signs([])` (which takes a list of localized mode indices as argument).

The vibrational coupling constants, i.e., the elements of the Hessian matrix in the basis of the localized modes, can be extracted with

```
cmat = lv.get_couplingmat()
```

The intensity coupling matrices are available via the LocModeAnalysis class. See the example and the source code for further details.

10.7 Advanced Features

The LocVib tools provide a number of additional functionalities, for instance for plotting vibrational spectra and coupling matrices. For more details on these tools, we refer to the source code and to the extended documentation that will be made available at http://www.christophjacob.eu/locvib.eu.

11. Parameter studies: step size, basis sets, SCF convergence, and grid type

This chapter presents investigations by C. Kind concerning the accuracy and numerical stability of the calculations for different values of the calculation parameters.

The step size for the numerical differentiation (`cstep`), the energy convergency criterion for the TURBOMOLE calculations (`scfconv`), the quality of the grid for the density functional calculation (`GRID`), the influence of the RI-approximation, and the basis set dependence were taken into account in these analyses.

The results presented here shall help to explain our choices of the default parameters for the calculations, and to provide a basis for error estimations in calculations with SNF.

11.1 Dependence on the step size (`cstep`)

As has already been mentioned in Sect. 6.1.3, it is dangerous to choose `cstep` values which are too small or too large. If the value is too small, there might occur instabilities in performing the central differences during the differentiation procedure. If it is too large, the harmonic approximation might fail. The following calculations have been carried out for *trans*-diazene using DFT/BP86 to analyse the effect of the `cstep` parameter.

- SVP/RI using TURBOMOLE default values (SVP default: SD)
- SVP/RI using `scfconv=8` and `GRID=5` (SVP accurate: SA)
- TZVP/RI using TURBOMOLE default values (TZVP default: TD)
- TZVP/RI using `scfconv=8` and `GRID=5` (TZVP accurate: TA)
- SVP calculation using GAUSSIAN98 (SVP analyt.)
- TZVP calculation using GAUSSIAN98 (TZVP analyt.)

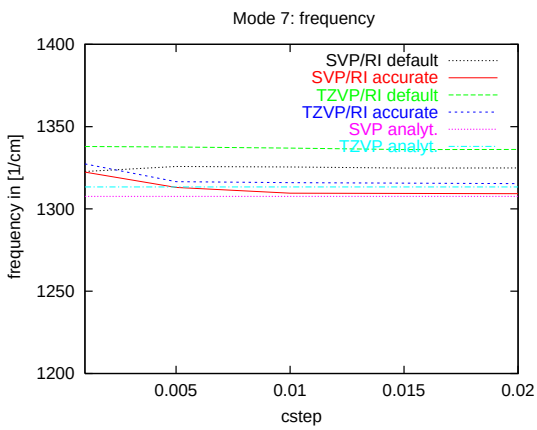
For each type calculations have been performed using the `cstep` values 0.001, 0.005, 0.01, 0.015, and 0.02.

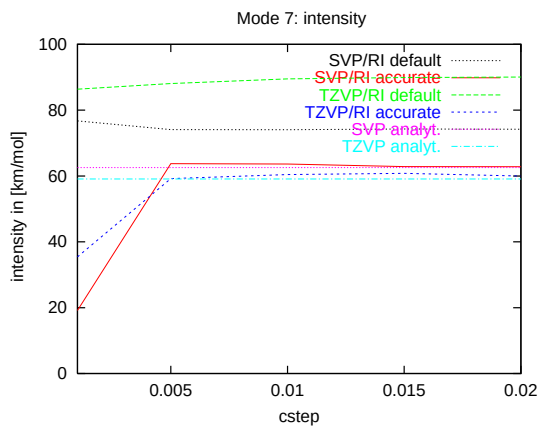
Calculations with GAUSSIAN98 should help to compare our numerical results to analytically determined frequencies, which, of course, are independent of the step size.

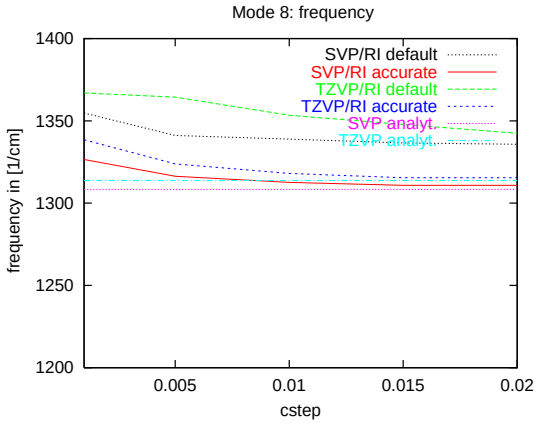
The following tables show the mean and maximum error for different `cstep` values compared to the analytical frequencies. All frequencies are given as wavenumbers in [1/cm], and the infrared intensities, which can also be found in the tables, are given as absorption coefficients in [km/mol].

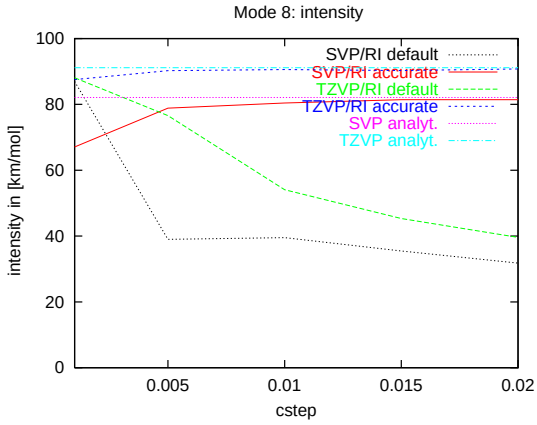
	mean error							
	IR intensities				frequencies			
cstep	SD	SA	TD	TA	SD	SA	TD	TA
0.001	13.5	33.1	12.6	5.9	53.0	25.7	60.5	22.1
0.005	23.7	1.0	13.7	0.3	42.0	7.0	52.9	5.7
0.010	25.7	0.6	28.4	0.4	35.8	4.6	42.7	5.0
0.015	24.1	0.3	29.2	0.5	35.4	4.0	40.7	4.1
0.020	26.0	0.3	32.1	0.3	31.3	4.1	34.3	3.9

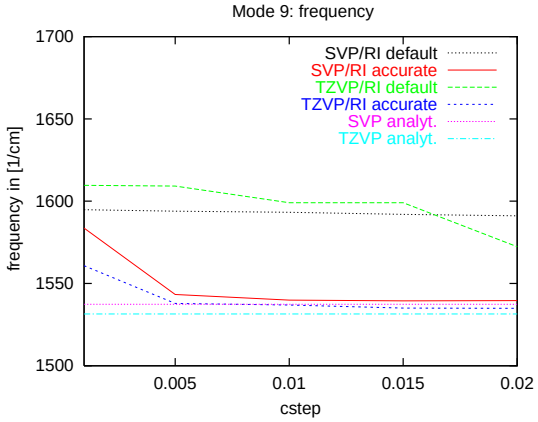
	maximum error							
	IR intensities				frequencies			
cstep	SD	SA	TD	TA	SD	SA	TD	TA
0.001	54.9	72.3	42.1	23.7	88.5	46.3	81.5	39.7
0.005	87.4	3.2	38.6	0.9	86.8	9.0	92.2	10.1
0.010	99.9	1.7	102.9	1.3	84.6	7.2	87.3	8.3
0.015	86.1	1.0	98.4	1.7	82.6	7.3	82.6	8.5
0.020	93.9	1.0	110.3	0.9	79.5	7.4	79.4	7.9

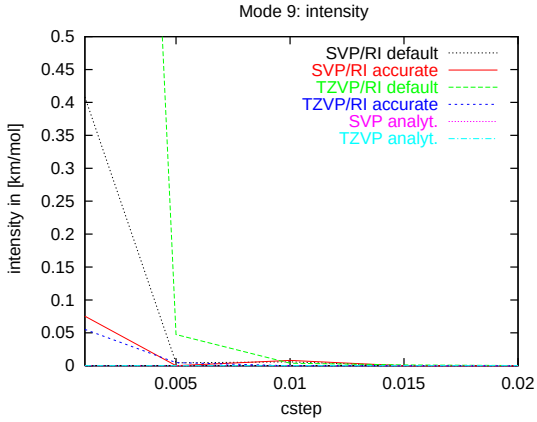


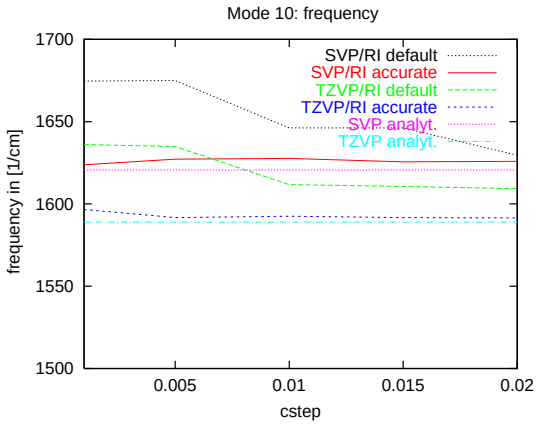


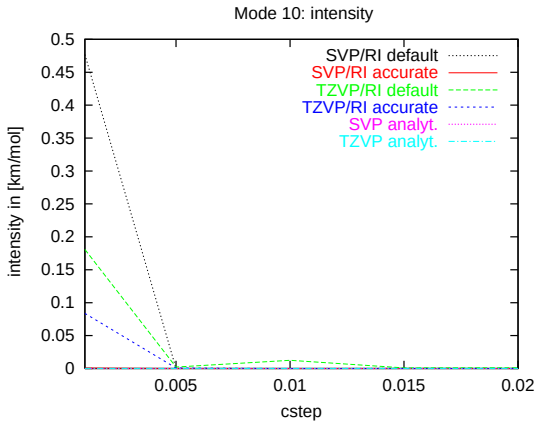

Mode 8: frequency
frequency in [1/cm]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.

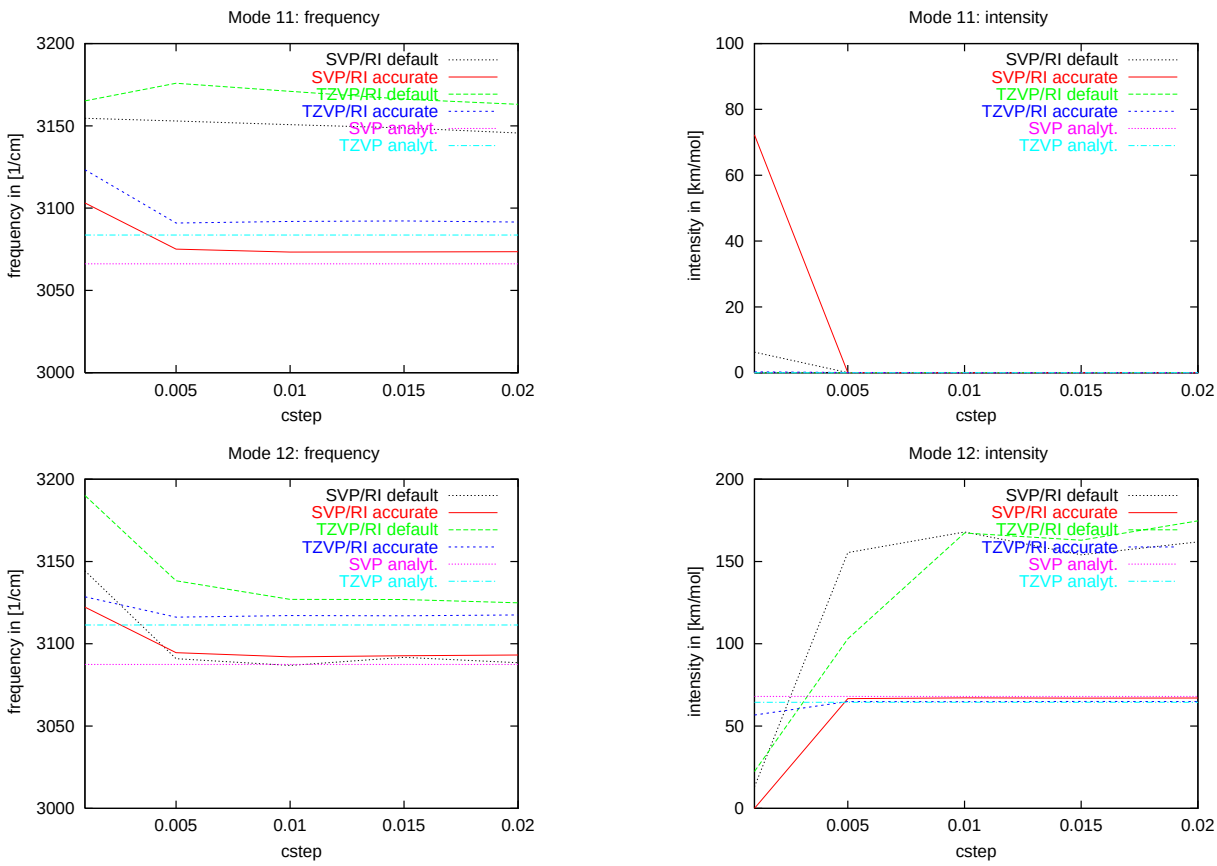

Mode 8: intensity
intensity in [km/mol]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.


Mode 9: frequency
frequency in [1/cm]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.


Mode 9: intensity
intensity in [km/mol]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.


Mode 10: frequency
frequency in [1/cm]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.


Mode 10: intensity
intensity in [km/mol]
cstep
SVP/RI default
SVP/RI accurate
TZVP/RI default
TZVP/RI accurate
SVP analyt.
TZVP analyt.



Since the numerical error of these calculations (3-point central differences formula) is proportional to $(\texttt{cstep})^2$, the step size should in principle be chosen as small as possible. The above results demonstrate, that the smallest possible `cstep`-value which ensures numerical stability is `cstep = 0.01`.

11.2 Influence of the SCF parameters `scfconv` and `GRID`

Of major importance for the numerical derivatives is the accuracy of the single point data. Two of the most important parameters for TURBOMOLE DFT calculations have been investigated to check their influence on the frequencies: the SCF convergency criterion `scfconv` and the grid characterization parameter `GRID`[1].

Calculations have been carried out for *trans*-diazene using a TZVP basis set and `cstep=0.01`.

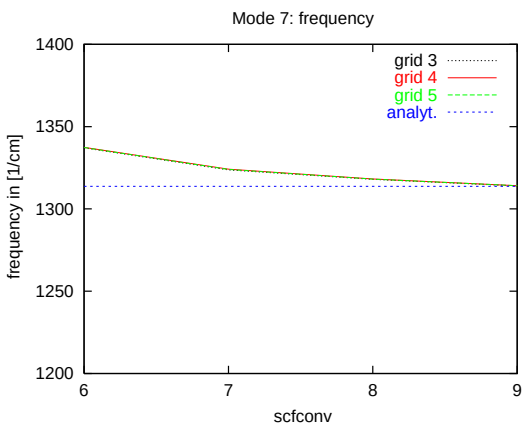
Mean and maximum errors of the numerical frequencies compared to the analytical ones are shown in the following two tables for different values of the parameters `scfconv` and `GRID`.

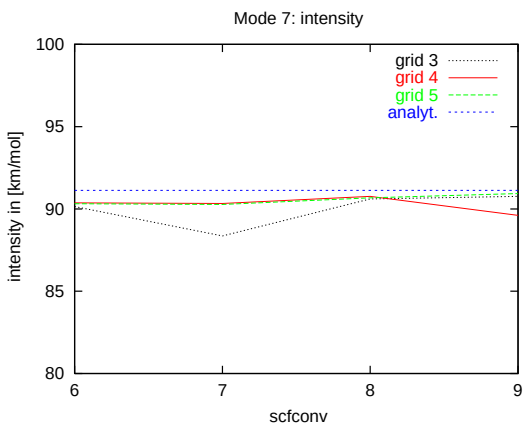
[1]For the meaning of the `GRID` parameter see the TURBOMOLE manual.

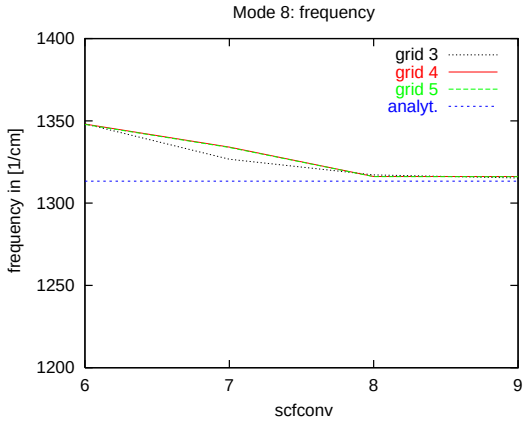
	mean error					
	IR intensities			frequencies		
scfconv	grid 3	grid 4	grid 5	grid 3	grid 4	grid 5
6	23.8	24.3	24.4	40.6	40.4	40.3
7	2.0	1.2	1.1	12.0	14.3	14.2
8	0.4	0.3	0.4	5.2	4.9	4.9
9	0.5	0.9	0.5	2.8	2.7	2.8

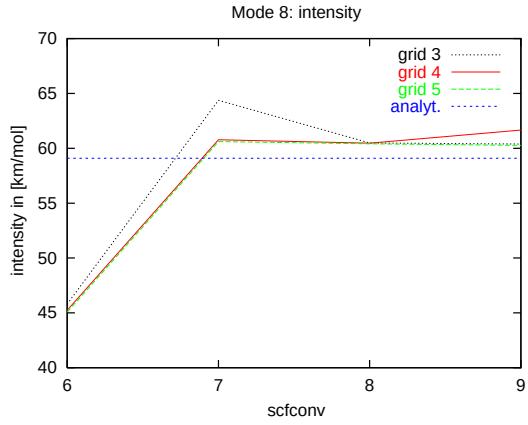
	maximum error					
	IR intensities			frequencies		
scfconv	grid 3	grid 4	grid 5	grid 3	grid 4	grid 5
6	128.6	131.2	131.5	85.4	85.2	85.3
7	5.3	4.6	4.5	16.6	20.7	20.4
8	1.4	1.4	1.3	7.2	7.9	7.7
9	1.6	2.6	1.4	4.9	4.3	4.5

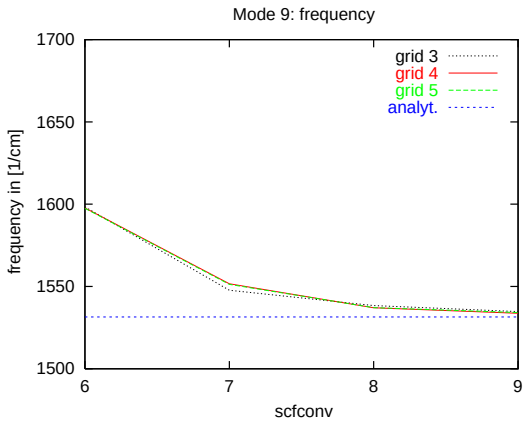
It can be concluded from these calculations, that only a simple grid (`GRID = 3`) is necessary. However, a `scfconv` value of 8 or higher is strongly recommended.

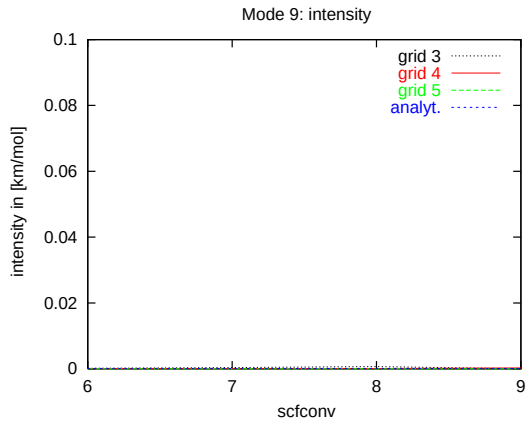


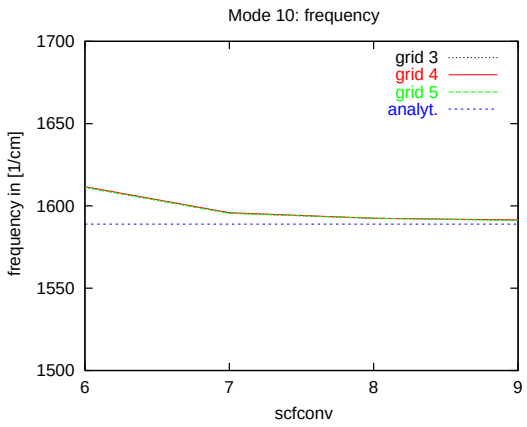


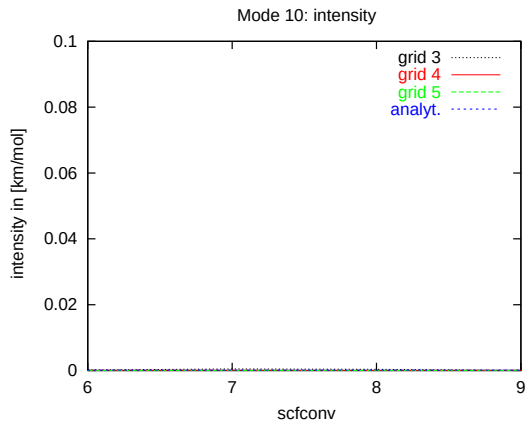

Mode 8: frequency
frequency in [1/cm]
scfconv
grid 3
grid 4
grid 5
analyt.
1400
1350
1300
1250
1200
6
7
8
9

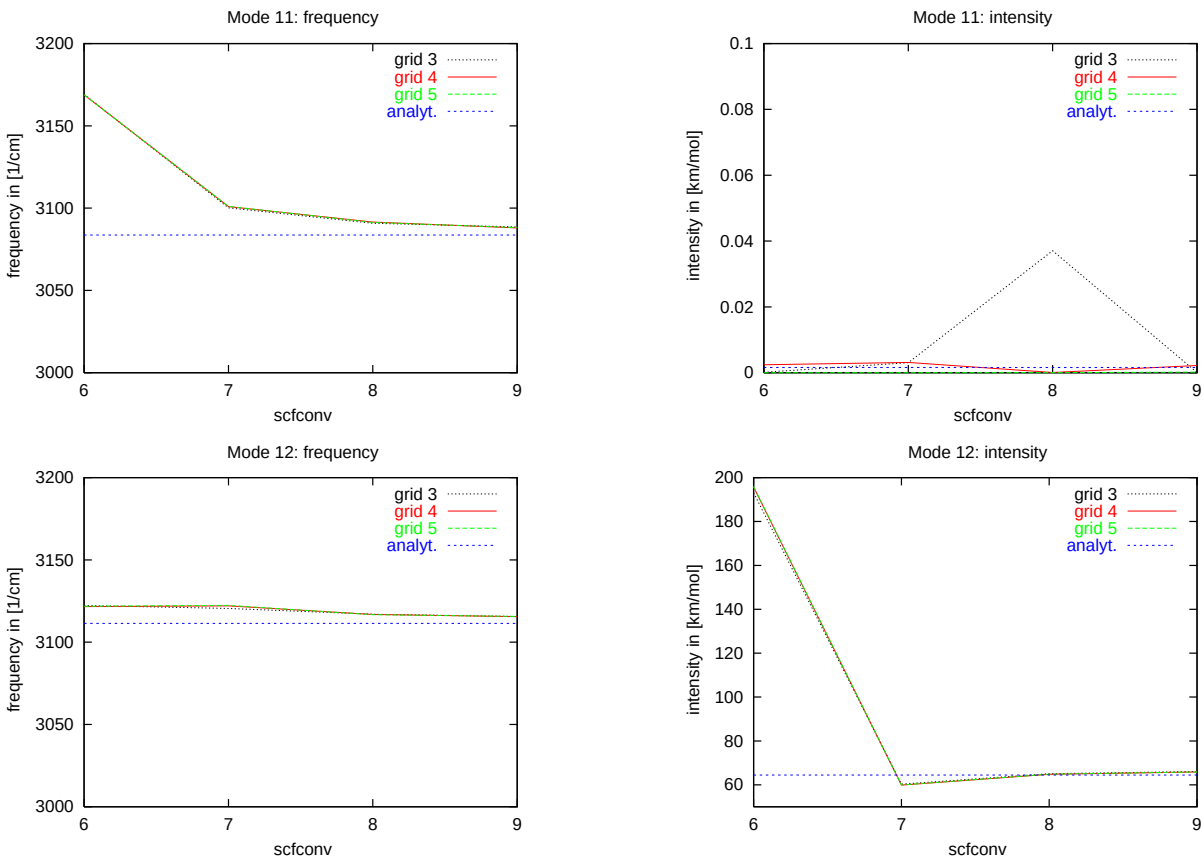

Mode 8: intensity
intensity in [km/mol]
scfconv
grid 3
grid 4
grid 5
analyt.
70
65
60
55
50
45
40
6
7
8
9


Mode 9: frequency
frequency in [1/cm]
scfconv
grid 3
grid 4
grid 5
analyt.
1700
1650
1600
1550
1500
6
7
8
9


Mode 9: intensity
intensity in [km/mol]
scfconv
grid 3
grid 4
grid 5
analyt.
0.1
0.08
0.06
0.04
0.02
0
6
7
8
9


Mode 10: frequency
frequency in [1/cm]
scfconv
grid 3
grid 4
grid 5
analyt.
1700
1650
1600
1550
1500
6
7
8
9


Mode 10: intensity
intensity in [km/mol]
scfconv
grid 3
grid 4
grid 5
analyt.
0.1
0.08
0.06
0.04
0.02
0
6
7
8
9



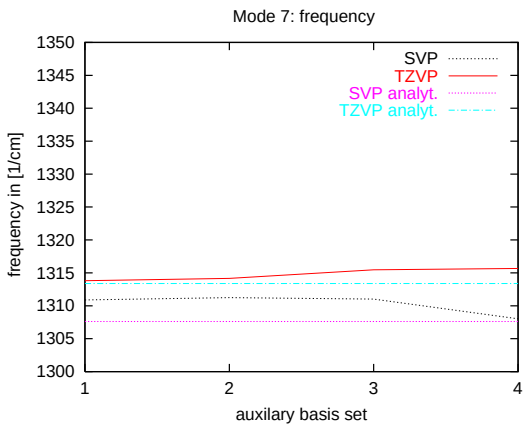
11.3 Influence of the RI-approximation

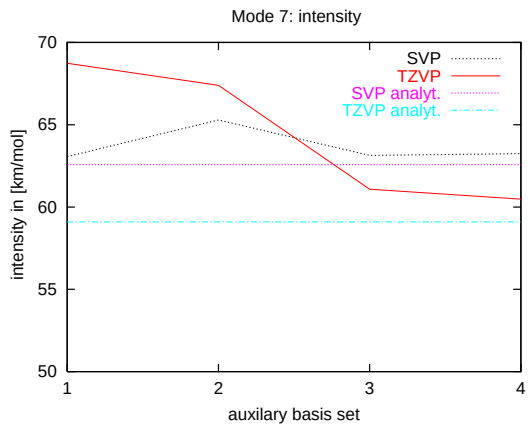
The influence of the RI-approximation [78] has been analyzed by varying the auxiliary basis set. Four different auxiliary basis sets have been investigated; however, for nitrogen the auxiliary basis set SV is the same as SVP and TZVP is the same as TZVPP. For hydrogen, all auxiliary basis sets are different.

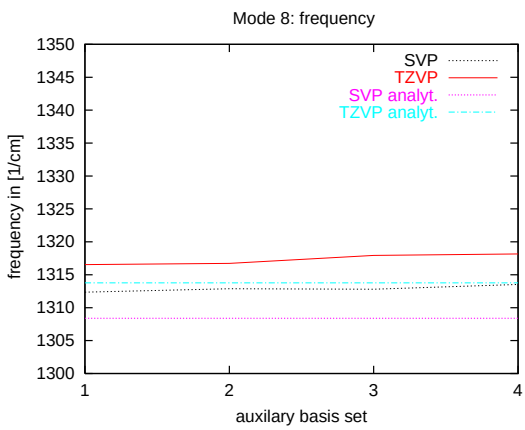
Mean and maximum errors (taking analytical frequencies without RI approximation as references) for the different auxiliary basis sets are listed in the tables below.

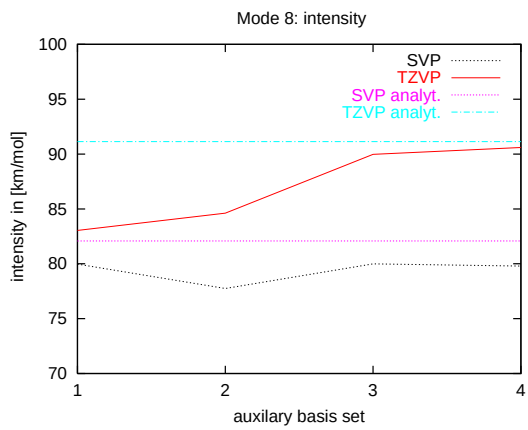
		mean error			
		IR intensities		frequencies	
No.	Auxbasis	SVP	TZVP	SVP	TZVP
1	SV	0.5	3.0	3.9	3.1
2	SVP	1.3	2.6	4.4	3.3
3	TZVP	0.5	0.6	4.5	4.3
4	TZVPP	0.6	0.4	4.8	4.9

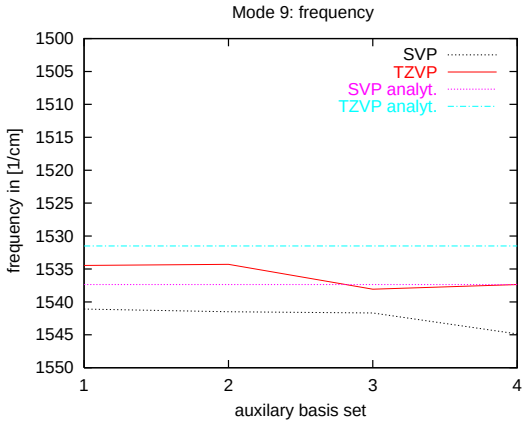
		maximum error			
		IR intensities		frequencies	
No.	Auxbasis	SVP	TZVP	SVP	TZVP
1	SV	2.1	9.6	6.9	10.9
2	SVP	4.3	8.3	8.0	11.6
3	TZVP	2.1	2.0	9.1	8.4
4	TZVPP	2.3	1.4	10.2	12.2

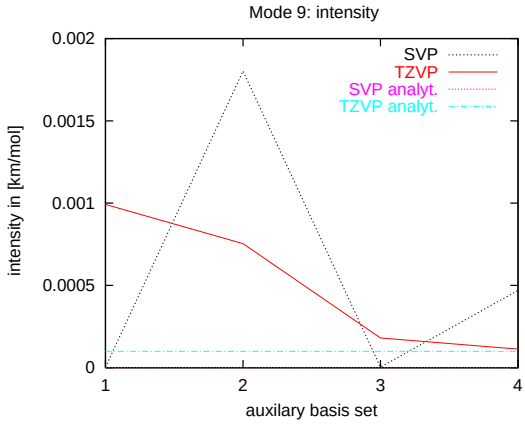


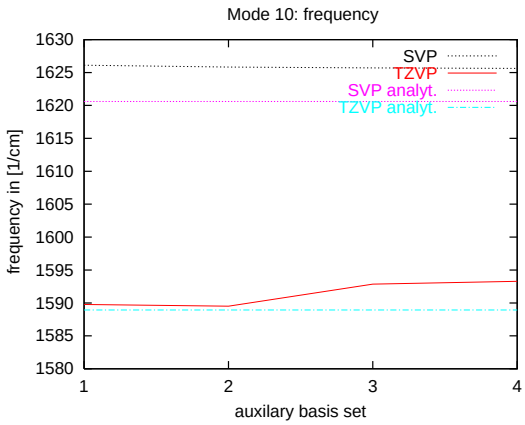


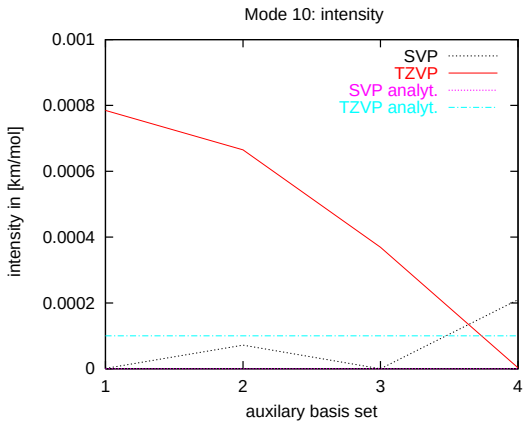


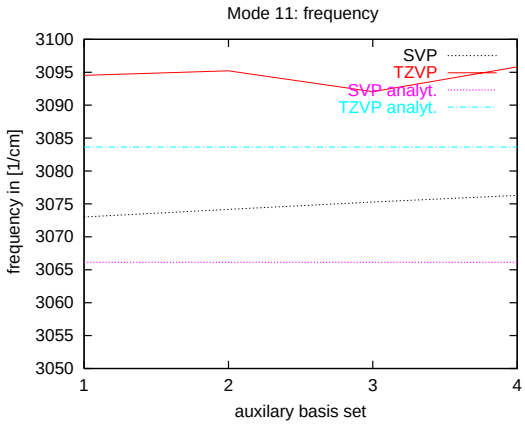


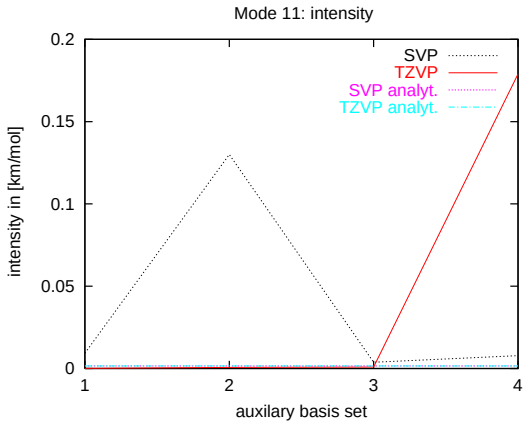

Mode 9: frequency
frequency in [1/cm]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1
2
3
4

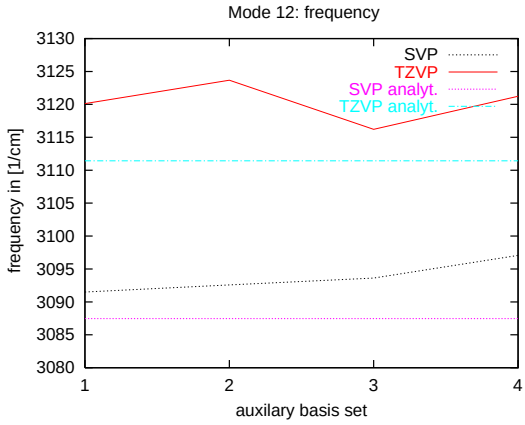

Mode 9: intensity
intensity in [km/mol]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
0.002
0.0015
0.001
0.0005
0
1
2
3
4

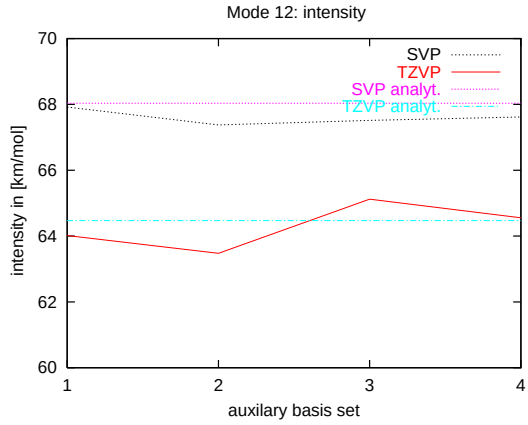

Mode 10: frequency
frequency in [1/cm]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
1630
1625
1620
1615
1610
1605
1600
1595
1590
1585
1580
1
2
3
4


Mode 10: intensity
intensity in [km/mol]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
0.001
0.0008
0.0006
0.0004
0.0002
0
1
2
3
4


Mode 11: frequency
frequency in [1/cm]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
3100
3095
3090
3085
3080
3075
3070
3065
3060
3055
3050
1
2
3
4


Mode 11: intensity
intensity in [km/mol]
auxilary basis set
SVP
TZVP
SVP analyt.
TZVP analyt.
0.2
0.15
0.1
0.05
0
1
2
3
4





As can be seen from these results, the effects of the auxiliary basis sets are very small.

11.4 Basis set dependencies

TURBOMOLE supplies 5 different standard split-valence basis sets. Contraction schemes for hydrogen and nitrogen are as follows:

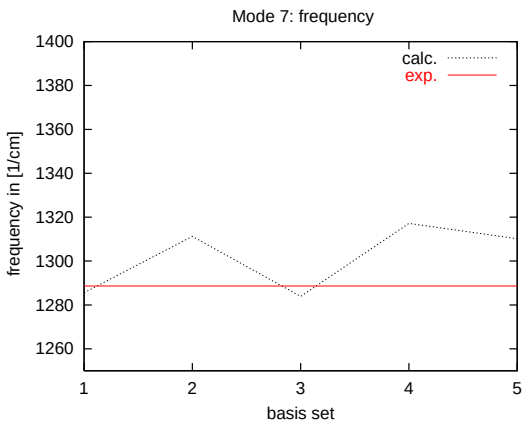
No.	Basis	H	N
1	SV	[4s,2s]	[7s4p,3s2p]
2	SVP	[4s1p,2s1p]	[7s4p1d,3s2p1d]
3	TZV	[5s,3s]	[11s6p,5s,3p]
4	TZVP	[5s1p,3s1p]	[11s6p1d,5s3p1d]
5	TZVPP	[5s3p,3s3p]	[11s6p2d1f,5s3p2d1f]

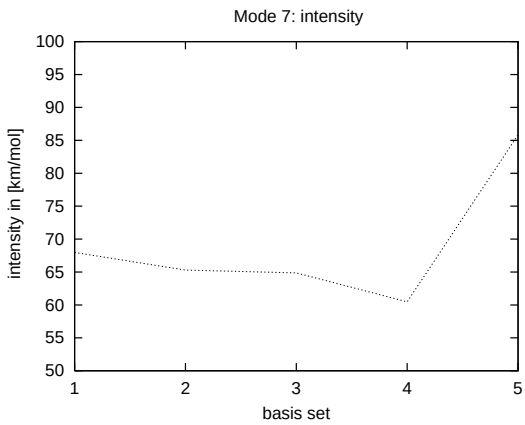
The following table contains maximum and mean errors compared to experimental frequencies (experimental IR intensities could not be obtained). Experimental frequencies for IR active modes have been taken from Hallin *et al.* [79] and from Hegelund and Burger [80] (gas phase). Frequencies of IR inactive modes have been taken from Raman experiments by Bondybey and Nibler [81] (N_2 matrix).

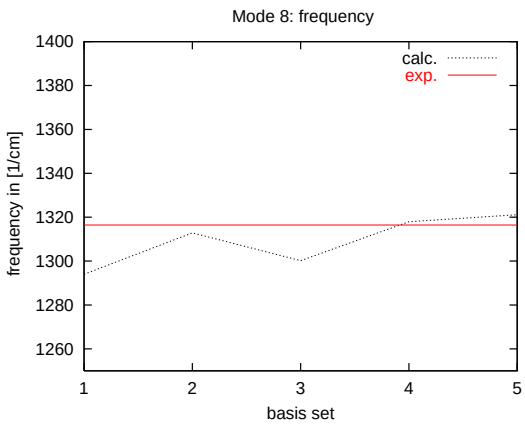
No.	Basis	max. error	mean error
1	SV	115.78	49.02
2	SVP	46.12	27.17
3	TZV	134.52	57.69
4	TZVP	29.47	14.89
5	TZVPP	21.52	10.73

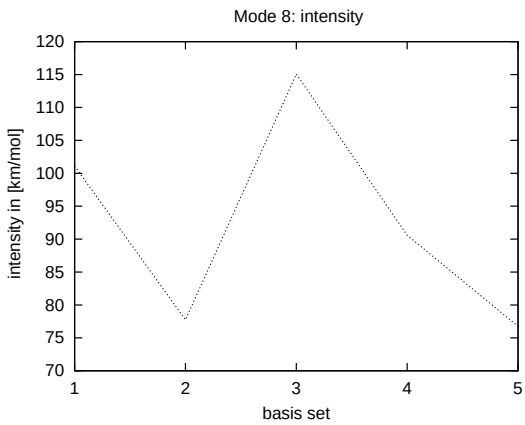
This indicates that polarization functions are mandatory in order to obtain satisfactory results; otherwise, errors up to 150 cm^{-1} may occur for the frequencies (and up to 20%

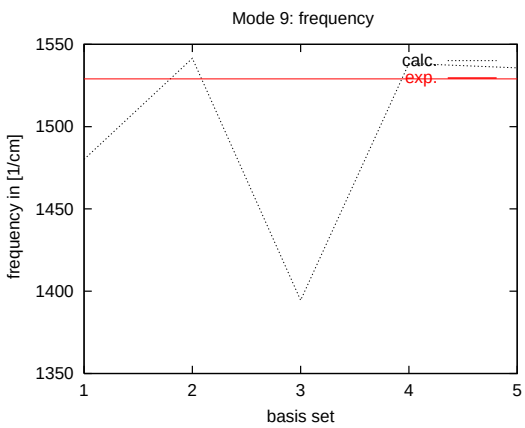
for the intensities). The mean error for a TZVP basis set is smaller than 15 cm^{-1}, such that this basis set should be preferred to obtain accurate results.

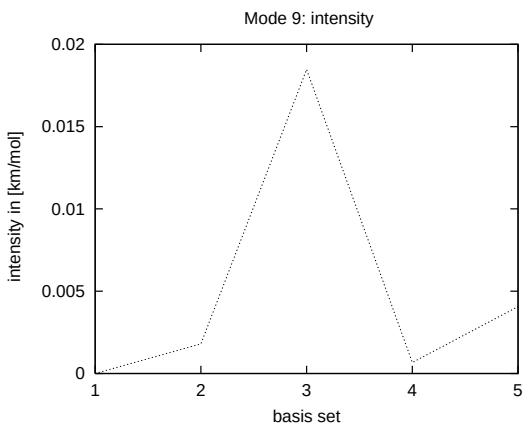


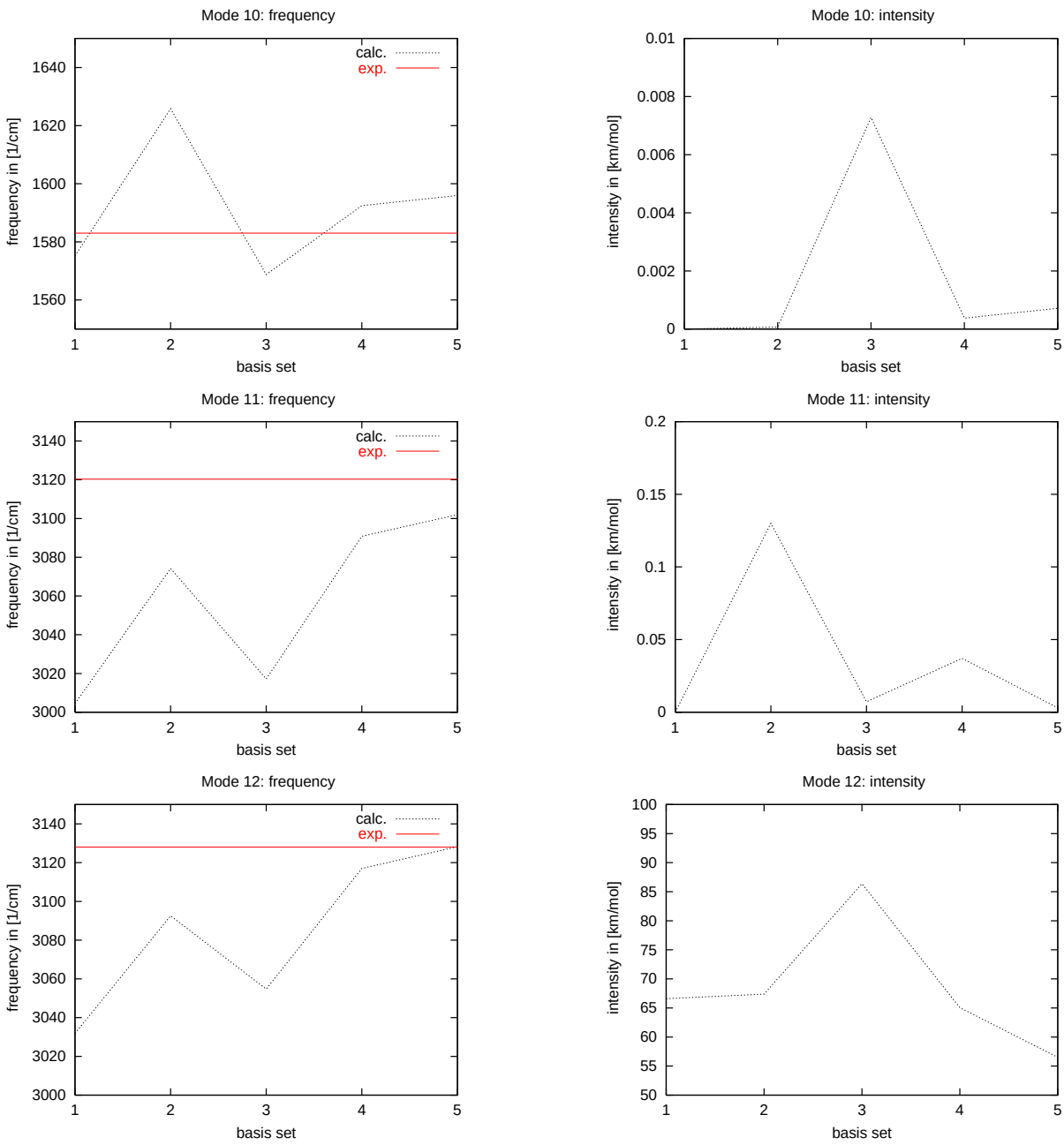












11.5 Raman intensities

For an analysis of the influence of the choice of basis set size and quantum chemical method on Raman intensities we refer to the extensive analyses given in [3] and [47].

12. Programmer's guide

12.1 New interfaces to quantum chemistry programs

SNF can be combined with any quantum chemical program which provides molecular gradients and dipole moments. These data are sufficient to calculate vibrational wavenumbers, normal modes and infared intensities. If furthermore electric-dipole–electric dipole (and electric-dipole–magnetic dipole and electric-dipole–electric quadrupole) polarizability tensors are available, calculations of Raman (and Raman optical activity) intensities are possible.

12.1.1 General issues

In the SNF source code, the programs are identified by the integer variable `progno`. At present, its values are defined as follows:

`progno`	Program
1	TURBOMOLE
2	DALTON
3	GAUSSIAN
4	ADF
5	MOLPRO

Since the in- and output of the single-point calculations may differ within a given program if different quantum chemical methods are employed, the types of calculations are distinguishable by the value of the integer variable `calcno`. The `progno` and `calcno` variables are managed by the subroutines `calc_type` and `prog_type`. *In order to check whether all necessary steps have been taken for programming the interface, it might be helpful to check the code for these two variables.*

12.1.2 Steps to take

1. Construct an input file for the quantum chemical program which yields the desired information, i.e., energy gradients, dipole moments, and, if neccessary, (generalized) polarizability tensors. If there is no input keyword avaliable which provides the gradients directly, it may be helpful to request a molecular structure optimization

("geometry optimization") which is stopped as soon as possible, for example after the first cycle.

2. Check where in the output the gradient, dipole moment etc. can be found, and for which output keywords you have to search in order to identify this information uniquely.

3. Change the source code in `src/snfdc/` and `src/snf/` such that the new `progno` and `calcno` are accepted, input files for the displaced structures are written, the correct quantum chemical program is started on the slave nodes, and the output is read in correctly when a single-point calculation is finished. The subroutines which have to be modified or added are listed in detail below. All new files must be added to `Makefile.am`. Do not modify either `Makfile.in` or `Makefile` since these are autogenerated files and your changes are lost upon the next invocation of `make` .

4. Compile the code with all possible combinations of parallelization type, file copying, and compilers (see `INSTALL`), and check whether all are still working.

12.1.3 Changes in the code

The following subroutines have to be added (first part of each table) or modified (second part). Replace `<np>` by a suitable name. *Just to be sure, you should check the code for* progno *and* calcno *and, if the routine was not in the list above, extend these sections as needed.*

		SNFDC
File	Subroutine	Purpose
`<np>rslave.F`	`<np>rslave`	Performs a single-point calculation on slave node, collects the results, and sends them to master.
`rw<np>.F`	as you like	Contains routines for writing input files for the displaced structures and reading gradients (i.e., copying the input file while replacing the equilibrium coordinates by the distorted ones), dipole moments, and, if neccessary, polarizabilities from the output files of the single-point calculations.
`getgrad.F`	`get<np>grad`	Reads gradient components from ouput files of slave calculation.

File	Subroutine	Purpose
		SNFDC
getpol.F	get<np>pol	Reads polarizability components from ouput files of slave calculation.
...?	...?	Get further information (dipole moments, energies, if these cannot be read within one of the routines above) from ouput files of slave calculation.
rslave.F	rslave	Driver for preparing input files for electronic structure programs for displaced coordinates
islave.F	islave	Initializes slave (child) processes
cslave.F	cslave	Driver for calling appropriate nprslave.
spawn.F	spawn	Controls file copying through MPI.
util.F	nfex	Returns the number of files to be copied to the slaves.
	calc_type	Returns type of calculation (as a character string) to be done for a given progno and calcno.
	prog_type	Returns type of program to be used (as a character string) for a given progno.
basic_input.F	basic_input	Driver for reading basic input files for SNFDC and electronic structure programs.

File	Subroutine	Purpose
		SNF
snf_define.f90	snf_define	Set up SNFDC run.
snf_control.f90	various	Initialize and control the frequency calculation.
snf_tools.f90	ramanpos	.true., if a Raman calculation is possible in a given program mode.

	SNF	
File	Subroutine	Purpose
	`vroapos`	`.true.`, if a VROA calculation is possible in a given program mode.
	...?	
`snf.f90`	`snf`	Evaluate results from SNFDC run.
`snf_<np>.f90`	`init_<np>_ctrl`	Initialize some variables for new program (can probably be shifted to `snf_<np>mol.f90`).
`snf_<np>mol.f90`	as you like	Contains routines for assuring that the right keywords for the gradients etc. calculations are present in the input files, and for reading the essential input information such as molecular equilibrium coordinates (should be unified with `rw<np>.F` one day).

13. Program history of SNF

The first version of the Fortran 90 program SNF was developed by C. Kind and M. Reiher at the University of Erlangen-Nuremberg in 1999. It used the program NUMFREQ [82] as the data collector.

The original Fortran 77 version of NUMFREQ was written by S. Grimme at the University of Bonn and contains contributions from C. Marian and M. Gastreich. This PVM parallelized version of the program has been largely extended in 1999 by B. A. Hess who also added the MPI, LAM-MPI, and the MPICH versions of the program. This extensive rewriting of the code, which yielded a program that can now solely be used for the general parallelized collection of data (gradients, etc.), led to the introduction of SNFDC (DC = data collector).

Substantial parts of the program SNF are based on subroutines for the utilization of projection operators and representation matrices, which have been written by C. Kind at the University of Cologne (1995/96). In winter 2000/2001 M. Reiher and C. Kind modified the program and included thermochemistry and Raman intensities based on static polarizabilities. In 2001 J. Neugebauer revised the existing code and included the cubic and icosahedral point groups as well as the calculation of Raman intensities based on dynamic polarizabilities. Further extensions by J. Neugebauer concern the treatment of dummy atoms and linear molecules, the calculation of isotope effects, and several output options.

Modifications after version 1.4.0 are listed below according to the version numbers.

5.0.1 Increase size of work arrays such that even larger molecules can be treated. Corrected some mistake in the manual regarding usage of Gaussian. General update of manual. This version is integrated into MOVIPAC 1.0.1.

5.0.0 Completely revised release for MOVIPAC 1.0.0.

4.2.1 Several minor bugfixes. Employ a tighter grid in Turbomole DFT calculations.

4.2.0 VROA calculations are now also possible with a modified version of TURBOMOLE. Note that this modified version is not officially distributed and you need the source code of TURBOMOLE to actually perform such calculations. The code now generally assumes that the operation system conforms to Posix.1-2001 and the C compiler fully supports C99. This is generally true for all systems shipped in 2003 or later.

Furthermore, the code was cleaned up and improved, and many smaller bugs were fixed (e.g., sleeptime issues with the MPI version).

4.1.0 The build-system was completely revamped and now makes full use of the autotools (`autoconf`, `automake`). The code was cleaned up and the output improved. Support for systems from the 1990s is dropped now. Support for newer ADF versions which use three digits for the exponent has been added and reading of polarizabilities from the `escf` module of different TURBOMOLE versions has been fixed. The MPI version had many deadlocks in the communication which where removed. Race conditions in the parallel code were fixed. The DALTON code was improved. The `$rij` flag used by newer TURBOMOLE versions is now recognized by SNF. Finally, a possibility to explicitly specify a user-defined symmetry number has been added.

4.0.3 This is a bugfix release fixing compiler warnings and problems with the creation of temporary files.

4.0.2 Many smaller bugs have been fixed. It is now possible to specify the sleeptime (i.e., the time the master process waits between two subsequent checks of the slave processes).

4.0.1 The configuration machinery was updated to allow different compilers for the F90 and F77 routines. IBM-AIX support has been added. Hessian-output in TURBOMOLE-style is now possible. Frequency analyses are now possible with ADF's QM/MM-interface.

4.0.0 SNF now features an RICC2 interface to TURBOMOLE. A serial version of SNFDC is available. Several changes have been made to update the configuration machinery and to simplify the installation process. A test suite and a tool for plotting in particular VROA spectra (`specplot`) have been added. Several smaller bugs were fixed.

3.5.6 Some minor bugs were fixed.

3.5.5 This is a bugfix release to fix some compiler warnings.

3.5.4 Conventional CIDs can now be calculated for an arbitrary list of normal modes. Initial blank lines in GAUSSIAN input files are now supported.

3.5.3 The values of the A tensor can now be set to zero for the calculations done for `snf.out`.

3.5.1 The GAUSSIAN interface was improved (e.g., user-provided basis sets are now supported).

3.5.0 A test interface to MOLPRO was added (for MCSCF calculations) and some minor bugs were fixed.

3.4.2 Normal modes can now be read in from the file `tm\_normmodes`.

3.4.1 Several modifications were done concerning the ADF interface and some smaller bugs in the intensity-only mode were fixed.

3.4.0 The GAUSSIAN input file is now automatically modified such that any symmetry keyword is removed for the single point calculations.

3.3.0 Interfaces to the quantum chemical program packages GAUSSIAN and ADF have been included.

3.2.0 This is a bugfix release.

3.1.0 The program was extended to allow for the calculation of Vibrational Raman Optical activity (VROA) spectra. These are now possible using the DALTON package. For the evaluation of the data (derivatives of London and non-London G tensor and of A tensors), parts of a subroutine from the ABACUS part of DALTON have been used with permission of K. Ruud.

3.0.0 The entire program has been restructured in order to change its former TURBOMOLE-specific character to a general structure that allows for an easy interfacing of new quantum chemical programs.

2.3.1 Output of the `escf` module of TURBOMOLE 5.6 is now supported.

2.3.0 SNF can calculate vibrational frequencies and infrared spectra for excited states via the `egrad` module of the TURBOMOLE 5.6 package, using either RPA or CIS (only for HF) calculations for the excited state. SNFDEFINE was extended accordingly to allow an easy setup of excited state calculations. Gauss- and Lorentz-type spectra can be plotted.

2.2.2 Individual `ELIGIBLE` files can be specified for the parallel calculation. Improved error handling for SNFDC. New options for the Raman spectrum output are available: all kinds of differential cross sections calculated by the program can be plotted (keyword `rptype <str>`, can be set via SNFDEFINE). Empirical scaling factors can be applied for the evaluation of the data.

Forward and backward differences are implemented to check the precision of the numerical differentiation.

2.2.1 Implementation of orientation independent symmetry detection routines. SNF now allows to use DALTON input files as well as TURBOMOLE input files for the preparation of DALTON single point calculations. BASED is no longer invoked, neither by SNFDEFINE, by SNF, nor by SNFDC.

2.2.0 SNF contains routines for the preparation of DALTON input files, which are invoked if BASED is not available. The molecular charge is determined via occupation numbers in `control`. SNF supports the calculation of HF polarizabilities using DALTON's `RESPONSE` module instead of the `ABACUS` module. SNF supports the calculation of Raman intensities from SOPPA polarizabilities in combination with MP2 frequencies and infrared intensities. `plotcperu` keyword available, replaced by `rptype <str>` keyword in V2.2.2.

2.1.6 Output of atomic contributions to vibrational intensities is possible via keyword `$atcontrib`.

2.1.5 Setting of `rpaconv` parameter possible in SNFDEFINE; 7-point central differences formula implemented. The code was cleaned up and improved.

2.1.4 Minor corrections in the LAM-MPI version of SNFDC and in module `snf_control`. SNF is now distributed under the terms of the GNU General Public License; see file `LICENSE` in the SNF installation directory.

2.1.3 MCSCF Raman and infrared intensities are available using DALTON. Data from old restart files can now be used if calculations shall be restarted using a higher number of grid points for the numerical derivatives.

2.1.2 Configuration script makes use of TURBOMOLE configuration files; optional configuration using the `PARM` file.

2.1.1 Minor error corrections in the MPICH version of SNFDC. SNFDC now prepares input files and runscripts for the MPI versions of SNFDC. SNF and SNFDEFINE fulfill Fortran 95 standard. Infrared intensities are available for TURBOMOLE MP2/RI-MP2-calculations. Logfiles for slave processes and temporary directories for single point calculations may be removed automatically.

2.1.0 SNFDC now supports the DALTON program modes.

2.0.1 The scripts `choose_nodes` and `mkdcinput` are executed by `snf_define` for easier preparation of SNFDC input files.

2.0.0 The TURBOMOLE version of SNF is available as a package containing the interactive input program SNFDEFINE, the data collector SNFDC, the evaluation program SNF, and all scripts which are necessary to run these programs.

1.8.1 Support for new `escf` module has been added.

1.8.0 A new directory hierarchy has been introduced.

1.7.9 Dynamic polarizabilities are now supported. Some minor bugs were fixed.

1.7.8 This is a bugfix release.

1.7.7 The internode communication has been improved.

1.7.6 An archive facility has been added.

1.7.5 A few minor bugs were fixed.

1.7.3 The internode comunication has been simplified and some small bugs were fixed.

1.7.2 The code was changed to to fully support open shell cases.

1.7.1 All buffers have been increased to 2500 characters such that log filenames do not cause problems anymore.

1.6.7 An MPI version of SNFDC is now available.

1.6.6 The nodes are now dynamically scheduled.

1.6.5 Some minor bugs were fixed and preparations for an MPI version have been made.

1.6.4 Issues with the load balancing were fixed.

1.6.3 The code was improved.

1.6.1 Load balancing have been introduced.

1.6.0 The `restart` file format has been adjusted.

1.5.4 Calculation of Raman intensities is now possible.

1.5.2 The code was improved for a better node managing.

1.4.4 The data for the 5-point central-differences formula can now be collected with SNFDC.

1.4.3 New ESCF versions (binaries denoted as `escf.intel` and `escf.athlon`) can be applied for the calculation of polarizabilities. They allow to use the RI-approximation in DFT calculations.

1.4.2 Symmetry redundant distortions of non-redundant atoms can be omitted.

1.4.1 In open-shell cases, total spin expectation values are read to calculate electronic (spin-only) contributions to the entropy etc.; scattering cross sections can be calculated for any scattering angle.

1.4.0 Implementation of the DALTON module. SNFDEFINE can now prepare DALTON calculations using the methods HF(SCF), MP2, CCS, CCSD, and CC2 if TURBOMOLE input files are available.

14. Supported point groups

SNF supports almost every molecular point group, with the exceptions of $C_{\infty v}$ and $D_{\infty h}$. In version 1.0 and all later versions, the following 45 point groups are implemented:

$C_1, C_i, C_s,$
$C_2, C_3, C_4, C_5, C_6, C_7, C_8$
$C_{2v}, C_{3v}, C_{4v}, C_{5v}, C_{6v},$
$C_{2h}, C_{3h}, C_{4h}, C_{5h}, C_{6h},$
$D_2, D_3, D_4, D_5, D_6,$
$D_{2h}, D_{3h}, D_{4h}, D_{5h}, D_{6h},$
$D_{2d}, D_{3d}, D_{4d}, D_{5d}, D_{6d},$
$S_4, S_6, S_8,$
$O, O_h, T, T_d, T_h, I, I_h$

15. Program modes

The program modes available in SNF are given in the table below, which also shows the corresponding program number (`progno`) and calculation number (`calcno`) which are stored in the `restart` file.

`progno`	`calcno`	program package	type of calc.	note
1	1	TURBOMOLE	SCF/DFT	
1	2	TURBOMOLE	RI-DFT	
1	3	TURBOMOLE	RI-MP2	no Raman
1	4	TURBOMOLE	MP2	no Raman
1	5	TURBOMOLE	EGRAD	no Raman
1	6	TURBOMOLE	EGRAD/RI	no Raman
1	7	TURBOMOLE	DSCF/RICC2	no Raman
2	1	DALTON	HF	
2	2	DALTON	MP2	no Raman
2	3	DALTON	CC2	
2	4	DALTON	CCS	
2	5	DALTON	CCSD	
2	6	DALTON	MCSCF	
2	7	DALTON	SCF/TDHF	
2	8	DALTON	MP2/SOPPA	
2	9	DALTON	DFT	
3	1	GAUSSIAN	GAUSSIAN98	no frequency-dependent Raman intensities
3	2	GAUSSIAN	GAUSSIAN03	
3	3	GAUSSIAN	GAUSSIAN09	
4	1	ADF	(all)	
5	1	MOLPRO	MCSCF	no Raman

16. Supported compilers

The program SNFDC is written in Fortran 77 while SNF is written in Fortran 90. In principle, any compiler supporting these language standards can be used. We test the program with the GNU compiler collection (`gfortran`), the Intel Fortran compiler (`ifort`) and the Portland Group compilers (`pgf77`, `pgf90`).

Since neither SNFDC nor SNF perform any time-consuming operations (in comparison to the required single-point calculations), there is in general no need for aggressive optimization flags. The `configure` script tries to determine the compiler automatically but prefers the GNU compilers `gfortran` and `gcc` above all others.

Note that the GNU Compiler Collection (GCC) has some issues with creating 64 bit code prior to version 4. We therefore neither recommend nor support usage of gcc-3.x. The current code was tested with `gcc-4.4.6`.

For the MPI version, a wrapper script called `mpif77` is required. This wrapper is usually provided by the MPI implementation.

17. Elimination of translational and rotational modes

In this section, two methods will be explained to eliminate the contributions of translational and rotational modes to the Hessian matrix.

The first method determines the translational and rotational fractions of each mode after diagonalization of the Hessian, such that the six modes with the largest percentages of non-vibrational contributions are neglected. The second method explicitly removes the contributions of pure non-vibrational modes before diagonalizing the Hessian. For details of the notation utilized here, see [3].

17.1 Determination of translational and rotational fractions

Every pure vibrational mode must preserve the center of mass, i.e.,

$$\Delta M = \left| \sum_{i=1}^{N} \mathbf{R}_i^{(c)} m_i \right| = 0 \ , \tag{17.1}$$

where $\mathbf{R}_i^{(c)}$ is the displacement vector of atom i in Cartesian coordinates in this normal mode. The change ΔM_{trans} for a pure translational normal mode may be defined by a normalized motion of all atoms along a selected direction $\mathbf{R}_{\text{trans}}^{(m)}$, where the superscript indicates the mass-weighting. The normalization condition is

$$\sum_{i=1}^{N} |\mathbf{R}_{i,\text{trans}}^{(m)}|^2 = \sum_{i=1}^{N} |\mathbf{R}_{i,\text{trans}}^{(c)}|^2 m_i = |\mathbf{R}_{\text{trans}}^{(c)}|^2 M_{\text{total}} = 1 \ , \tag{17.2}$$

where we made use of the fact that all displacements $\mathbf{R}_{i,\text{trans}}^{(c)}$ are equal by definition. M_{total} is the total mass of the molecule. The translational amplitude is thus

$$|\mathbf{R}_{\text{trans}}^{(c)}| = \frac{1}{\sqrt{M_{\text{total}}}} \quad . \tag{17.3}$$

With the above equation, we obtain for ΔM_{trans}

$$\Delta M_{\text{trans}} \quad = \quad \left| \sum_{i=1}^{N} \mathbf{R}_{\text{trans}}^{(c)} m_i \right| = \sum_{i=1}^{N} \frac{m_i}{\sqrt{M_{\text{total}}}} = \sqrt{M_{\text{total}}} \ , \tag{17.4}$$

such that we get for the translational percentage

$$c_{trans} = \frac{\Delta M}{\Delta M_{trans}} = \frac{\Delta M}{\sqrt{M_{total}}} \ . \tag{17.5}$$

For the rotational percentage, an analogous procedure can be applied. In this case, the resulting angular momenta $\Delta\mathbf{L}$ and $\Delta\mathbf{L}_{\mathrm{rot}}$ have to be considered,

$$\Delta\mathbf{L} = \sum_{i=1}^{N} m_i(\mathbf{r}_i \times \mathbf{R}_i^{(c)}) \ , \tag{17.6}$$

where $\mathbf{r}_i$ are the Cartesian coordinates of atom i in the center-of-mass coordinate system, as they may be obtained from the input file. The axes of rotation are per definition the x-, y-, and z-axis in this "laboratory coordinate system". Note that the above formula only describes angular momenta resulting from differential displacements of the atoms. But since this also holds for the determination of the normal modes, i.e., since rotational normal modes are only pure rotational for differential motions, it is legitimate to use the above definition in this framework.

The resulting angular momentum $\Delta\mathbf{L}_{\mathrm{rot}}$ for a pure rotation about the axis specified by the vector $\Delta\mathbf{L}$ for the normal mode under consideration is normalized as follows:

$$\sum_{i=1}^{N} |\sqrt{m_i}(\mathbf{r}_i \times \vec{\omega}_{\mathrm{rot}})|^2 = \sum_{i=1}^{N} m_i |\mathbf{r}_i^{\perp}| \cdot |\vec{\omega}_{\mathrm{rot}}|^2 = 1 \ . \tag{17.7}$$

Here, $\vec{\omega}_{\mathrm{rot}}$ is the angular velocity around the specified axis, which is identical for all atoms in this pure rotation, and $\mathbf{r}_i^{\perp}$ is the component of $\mathbf{r}_i$ perpendicular to this axis. The absolute value $|\vec{\omega}_{\mathrm{rot}}|$ is then given as

$$|\vec{\omega}_{\mathrm{rot}}| = \frac{1}{\sqrt{\sum_{i=1}^{N} |\mathbf{r}_i^{\perp}|^2 m_i}} \ . \tag{17.8}$$

The angular velocity is related to the angular momentum by the moment of inertia I,

$$\vec{\omega}_i = I^{-1}\Delta\mathbf{L}_i \ . \tag{17.9}$$

The final result for the rotational percentage to a normal mode is

$$c_{\mathrm{rot}} = \frac{\Delta\mathbf{L}_i}{\Delta\mathbf{L}_{\mathrm{rot}}} = \frac{|\vec{\omega}_i|}{|\vec{\omega}_{rot}|} \ . \tag{17.10}$$

The values c_{trans} and c_{rot} may then be used to eliminate the modes with the three largest percentages of rotational and translational contributions.

17.2 Elimination of non-vibrational contributions by means of projection operators

In the second method, pure translational and rotational modes, i.e., motions along the coordinates $\mathbf{R}^{(m)}_{\text{trans}}$ and $\mathbf{R}^{(m)}_{\text{rot}}$, are constructed and projected off the Hessian.

For this purpose, the projection operators

$$\mathbf{P}^{(m)}_{\text{trans}} = 1^{(m)} - \sum_{\text{trans}}^{3} \mathbf{R}^{(m)}_{\text{trans}} \mathbf{R}^{(m)\dagger}_{\text{trans}} , \tag{17.11}$$

$$\mathbf{P}^{(m)}_{\text{rot}} = 1^{(m)} - \sum_{\text{rot}}^{3} \mathbf{R}^{(m)}_{\text{rot}} \mathbf{R}^{(m)\dagger}_{\text{rot}} , \tag{17.12}$$

are applied on the matrices[1] $\mathbf{S}^{(m)}_{\mu}$ and the resulting matrix is orthonormalized with an appropriate matrix $\mathbf{A}$[2]. The matrix now obtained,

$$\mathbf{V}^{(m)}_{\mu} = \mathbf{A}\mathbf{P}^{(m)}_{\text{trans}} \mathbf{P}^{(m)}_{\text{rot}} \mathbf{S}^{(m)}_{\mu} , \tag{17.13}$$

is a transformation matrix in the basis of pure vibrational eigenvectors of the Hessian matrix. Applying this matrix on the mass-weighted Hessian yields

$$\mathbf{V}^{(m)\dagger}_{\mu} \mathbf{F}^{(m)} \mathbf{V}^{(m)}_{\mu} = \mathbf{F}^{(v)}_{\mu} \quad . \tag{17.14}$$

The dimensions of matrix $\mathbf{F}^{(v)}_{\mu}$ are reduced by the number of removed translational and rotational modes, which correspond to the irrep Γ_{μ} (compared to the matrix $\mathbf{F}^{(s)}_{\mu}$, which would be obtained without projecting off the non-vibrational contributions).

The pure translational and rotational modes are generated as follows:

- As far as the translational modes are concerned, these are assumed as motions along the axes of the laboratory coordinate system. The absolute value of the normalized distortion is known from Eq. (17.3). After mass-weighting, we obtain for the entry corresponding to atom i and the rotational mode j

$$R^{(m)}_{\text{trans},ji} = \sqrt{m_i}\mathbf{R}^{(c)}_{j} = \frac{\sqrt{m_i}}{\sqrt{M_{\text{total}}}}\mathbf{e}_j \quad . \tag{17.15}$$

- The treatment of the rotational modes is more complicated. They are only orthogonal if the axis of rotation are the principal axis of the moment of inertia. This can be achieved by diagonalizing the matrix of the moment of inertia, $\mathbf{I}$,

$$\omega^{\dagger}_{\text{rot}} \mathbf{I} \omega_{\text{rot}} = \mathbf{I}^{(\text{diag})} \quad . \tag{17.16}$$

[1] These matrices contain the eigenvectors of the projection operators that block-diagonalize the mass-weighted Hessian; see [3] for details.

[2] This matrix is constructed via *Gram–Schmidt* orthogonalization, such that $\mathbf{V}^{(m)}_{\mu}$ is orthonormal and does not contain any zero vectors.

The absolute value of ω_{rot} is given by Eq. (17.8). After mass-weighting we obtain for the entry corresponding to atom i and the rotational mode j

$$R^{(m)}_{\mathrm{rot},ji} = \sqrt{m_i} R^{(c)}_{ji} \tag{17.17}$$

$$= \sqrt{m_i}(\mathbf{x}^{\perp}_{ij} \times \omega_{\mathrm{rot},j}) \tag{17.18}$$

$$\mathbf{x}^{\perp}_{ij} = \mathbf{x}_i - \frac{\mathbf{x}_i \cdot \omega_{\mathrm{rot},j}}{\omega^2_{\mathrm{rot},j}} \omega_{\mathrm{rot},j} \quad . \tag{17.19}$$

The modes generated with this treatment correspond to circular motions, i.e., to exact rotations only in case of infinitesimal distortions. However, this is also the case for exact rotational normal modes, since they can also describe curvilinear motions only in the limit of infinitesimal steps.

18. Tools and scripts

18.1 Scripts helpful for running snfdc

In the subdirectory `scripts` of the installation directory, you can find several scripts, which are either mandatory or useful for running the programs of the SNF package. Therefore, these scripts should be available in your `$PATH`. The scripts have the following purposes:

`choose_nodes`: selects nodes from `$CIPROC/ELIGIBLE` file to be added to the virtual parallel machine; see Secs. 5.4 and 5.5

`dc_killtask`: mandatory script for SNFDC; see Sec. 5.5

`mkdcinput`: prepares the `DCINPUT` file; see Secs. 5.4 and 5.5

`dcmore`: SNFDC status monitor; displays the file `TMPdcstat` in time intervals of 5 seconds. The update time can be set via `dcmore <val>`, `<val>` = update time in seconds.

18.2 Graphical tools

For the graphical visualization of the spectra obtained with SNF, the Fortran 77 program SPECPLOT is available under `src/utilities/specplot_1.4/`. While IR and Raman spectra are plotted by `Snf` by default, SPECPLOT is particularly designed for the following special purposes:

- plot VROA spectra (only backscattering intensities implemented so far),
- plot Raman spectra which are obtained as a "byproduct" of a VROA calculation (only backscattering intensities implemented so far),
- plot selected peaks of Raman and VROA spectra only,
- plot several IR spectra, which are read from `g98.out` files, into one figure.

The first three points mentioned above require a control file `vroaplot_control`. Examples may be found in `examples/specplot`. SPECPLOT is based on GNUPLOT. A

special feature of the program is that it provides the spectrum as a postscript file, which is automatically shown via GHOSTVIEW, and which is updated after every command you enter (provided the "watch file" option is selected in your GHOSTVIEW settings). Such commands might request, for example, adjusting axis ranges, selecting and deselecting individual peaks, or some other features, which can be chosen in the main SPECPLOT menu:

```
Generate spectra plots
 ----------------------

  The current plot of your spectra is just being shown.
  The following settings can be changed:
  [d]ist      :  distance between baselines
  [xr]ange    :  minimum and maximum frequency
  [ymin] <num>:  minimum intensity = <num>
  [ymax] <num>:  maximum intensity = <num>
  [tit]le     :  title
  [xl]<string>:  label of x-axis = <string>
  [yl]<string>:  label of y-axis = <string>
  [l]egends   :  legends of individual plots
  [gnu]plot   :  enter gnuplot commands
  [th] <int>  :  set thickness of lines to <int>
  [pg/plo/pli]:  plot {\sc Gaussian} / Lorentzian /line data
  [ali]       :  add line spectrum to
              :  {\sc Gaussian} / Lorentzian plot
  [nali]      :  remove line spectrum
  [scli <num>]:  scale original line spectrum
                 by factor <num>
  [pw <num>]  :  use peakwidth = <num>
  [spw]       :  adjust peakwidth for selected region
  [rm] <a-b>  :  extinguish wavenumber range <a> to <b>
  [re] <fname>  <a-b> : reread peaks in wavenumber
                 range <a> to <b> from file <fname>
  [wi] <num>  :  width of plot  = <num>
  [he] <num>  :  height of plot  = <num>
  [gr] <a-b>  :  group peaks in wavenumber range
                 <a> to <b>
  [sqgr]      :  group peaks in wavenumber range
                 <a> to <b> and plot them as rectangles
```

```
Further options:
! <command> :  execute shell command <command>
[pr] <int>  :  print list of freq.s and intensities
               for <int>th file (default: file 1)
[sa <file>] :  save spectrum as file.eps, file.gpin
               and file.dat
[su] <a>-<b>:  print sum of intensities of all peaks
               in wavenumber range <a> to <b>
<quit>     :  quit without changes
```

Please note that SPECPLOT is work in progress — it should work for all the above-mentioned purposes, but in some cases, you may need to adjust some lines in the code to get exactly the result you want. The recommended compiler is Intel's `ifort`.

Part II

Akira

19. Introduction

This manual covers the program AKIRA [21] and is organized as follows:

Sect. 21 contains help on how to get AKIRA binaries running on your system. A description of the installation of AKIRA is given and the requirements for the parallelization are explained.

In Sect. 22, the theoretical background of the subspace iteration methods employed in AKIRA is illustrated and the numerical derivative methods are explained in detail. Sect. 23 deals with how to prepare and to run modetracking calculations using the commands AKIRADEFINE and AKIRA, respectively. An analysis of the influence of the step size used in numerical differentiation on the calculation of vibrational frequencies is given at the very end.

Any use of this program that results in published material should cite the following:

M. Reiher, J. Neugebauer. A mode-selective quantum chemical method for tracking molecular vibrations applied to functionalized carbon nanotubes. *J. Chem. Phys.*, **118**(4) (2003) 1634–1641.

This article introduces the MODE-TRACKING method and describes the implementation of the algorithm; comparisons of some initial guesses and preconditioners are also given. We should, however, emphasize that we cannot warrant that the method is indeed suitable for your particular purpose.

Further details of the program capabilities and examples for mode-tracking calculations are given in the following articles:

- *extraction of couplings between different parts of a molecule, applications to large gold clusters:*
  J. Neugebauer, M. Reiher *J. Comput. Chem.* **25** (2004), 587 – 597.

- *local vibrations in large molecules, applications to molecular-wires type transition metal complexes:*
  J. Neugebauer, M. Reiher *J. Phys. Chem. A* **108** (2004), 2053 – 2061.

- *systematic comparison of initial guesses, preconditioners, convergence characteristics, preconditioning schemes, near-degeneracies:*
  M. Reiher, J. Neugebauer *Phys. Chem. Chem. Phys.* **6** (2004), 4621 – 4629.

- *applications to molecules adsorbed at surfaces:*
  C. Herrmann, M. Reiher *Surf. Sci.* **600** (2006), 1891 – 1900.

- *QM/MM calculations:*
  C. Herrmann, J. Neugebauer, M. Reiher *J. Comput. Chem.* **29** (2008), 2460 – 2470.

For further information on the semi-numerical implementation in our vibrational spectroscopy programs, see the following articles, which do also give references to important work by other groups:

- general review of theoretical vibrational spectroscopy for large molecules:
  - C. Herrmann, M. Reiher *Top. Curr. Chem.*, **168** (2007), 85 – 132.
- parallelized, semi-numerical calculation of vibrational frequencies and intensities (in particular, Raman intensities [2nd paper]):
  - J. Neugebauer, M. Reiher, C. Kind, B. A. Hess. *J. Comput. Chem.*, **23** (2002), 895 – 910.
  - J. Neugebauer, M. Reiher, B. A. Hess. *J. Chem. Phys.*, **117** (2002), 8623 – 8633.
- anharmonicity effects in the density functional framework (1st paper) and error cancellation of harmonic BP86/TZVP frequencies in comparison with fundamental frequencies (both papers below):
  - J. Neugebauer, B. A. Hess. *J. Chem. Phys.*, **118** (2003), 7215 – 7225.
  - M. Reiher, G. Brehm, S. Schneider. *J. Phys. Chem. A*, **108** (2004), 734 – 742
- mode-wise calculation of intensities for vibrational spectra:
  - M. Reiher, J. Neugebauer, B. A. Hess. *Z. Phys. Chem.*, **217** (2003), 91 – 103.
  - J. Neugebauer, M. Reiher, B. A. Hess, in: S. Wagner, W. Hanke, A. Bode, F. Durst (Eds.), *High-Performance Computing in Science and Engineering 2000-2002*, Springer-Verlag, Berlin **2002**, pp. 157 – 169.

20. Quickstart

The program package AKIRA has been developed to implement the Mode-Tracking idea. Selected vibrations (normal modes and wavenumbers) can be obtained using the harmonic approximation. The vibrational frequencies are determined using numerical differentiation of analytic gradients of the total electronic energy with respect to collective Cartesian nuclear coordinates. AKIRA requires single-point calculations with either DALTON, TURBOMOLE, ADF, or GAUSSIAN, which can be performed using coarse-grained parallelization (MPI). Note that you need to possess an official licence for any of these quantum chemistry packages! AKIRA does not intermingle with any of these programs but only scans the output of them in order to extract all relevant raw data for the Mode-Tracking protocol. AKIRA will automatically start DALTON, TURBOMOLE, ADF, or GAUSSIAN single-point jobs on slave nodes (if no computer cluster is available it is possible to run AKIRA in a single-processor mode). For the easy set up and handling of the calculations you may start the set-up tool AKIRADEFINE. Normal modes may be tracked for any electronic structure method implemented in DALTON, TURBOMOLE, ADF, or GAUSSIAN for which analytic energy gradients are available.

AKIRA comes as part of the MOVIPAC program package. By default, only the program SNF is installed. The installation of AKIRA can be enabled by using the flag `--enable-akira` in the configure script, i.e., you need the following steps:

To install Akira, you need the following steps: To install SNF, you need the following steps:

- read this manual
- unpack: `tar -xvjf snf-4.2.1.tar.bz2`
- change to subdirectory: `cd snf-4.2.1`
- set up build environment: `autoreconf -i -v`
- configure package: `./configure --prefix=PREFIX_DIR --with-mpi --enable-akira`
- compile package: `make`
- install binaries to `PREFIX_DIR`: `make install`
- clean up installation process: `make clean`

`PREFIX_DIR` denotes the directory where the SNF and AKIRA binaries are installed to. The path to this directory needs to be assigned to the environment variable `SNF_PATH`. Moreover, `SNF_PATH/bin` needs to be added to the environment variable `PATH`.

More detailed information about installing AKIRA can be found in section 4.2.

21. Installation and technical issues

21.1 Getting started

21.1.1 General: the commands in the bin and scripts directory

In the bin directory of the installation directory, you will find two binaries, `akiradefine` and `akira`. You will call `akiradefine` in order to set up the modetracking calculation, and `akira` in order to perform the calculation.

In the same directory, the following scripts are available:

script	use of script	needs ...	produces ...
`mkrdcinput`	specifies paths and directories for `akira`	—	`DCINPUT`
`choose_nodes`	checks cluster for free nodes and writes them to file `USE_NODES`	`ELIGIBLE`, `ruptime`	`USE_NODES`
`dcmore`	monitors Akira calculation by displaying `TMPdcstat` file, updating it by default every 5 seconds. The update time can be set via `dcmore <val>`, `<val>` = update time in seconds.	—	—

In order to assure that `akiradefine` finds these scripts, you can add the scripts directory to the `$PATH` in your `.bashrc` or copy the scripts into the bin directory in your home.

`mkrdcinput` and `choose_nodes` are called automatically by `akiradefine`. For further information on how to monitor a calculation with `dcmore`, see section 21.4.

Examples for all in- and output files mentioned here can be found in the `examples/files` subdirectory in the source directory. Furthermore, a detailed explanation of the `DCINPUT` file and the `choose_nodes` command is given in the subsections 21.1.2 to 21.1.3.

As far as Akira is concerned, all paths necessary for MPI and for the quantum chemical package you want to use, that is Turbomole, Dalton, Adf or Gaussian, must be set correctly. In the `examples/files` subdirectory, you will find a `bashrc` file which contains an example of all paths that need to be set for Akira.

In the following three subsections, more detailed information will be given on the DCINPUT file and the preparations necessary for MPI.

21.1.2 The DCINPUT file

This file has the following structure:

```
DCPATH /usr/bin/akira
MYDIR /home/<usrname>/calculations/example
PREFIX TMP<usrname>%
TMPDIR /tmp
LOGDIR /home/<usrname>/calculations/example/log
```

with the variables

- DCPATH: path to the Akira executable file (optional)
- MYDIR: your working directory (mandatory)
- PREFIX: prefix for logfiles and temporary files (optional)
- TMPDIR: directory for temporary files (including all temporary files of the single point calculations); must be available on every machine (optional)
- LOGDIR: directory for logfiles (optional)

If the optional variables are not specified in DCINPUT, Akira will try to use $MYDIR/tmp as TMPDIR and $MYDIR/log as LOGDIR. PREFIX will be set to "TMP".
The variables DCPATH,TMPDIR, and LOGDIR may be specified individually for some (or all) of the slave nodes. This is achieved by the entry
DCPATH(<nodename>) <pathname>
etc. Additional optional variables are

- PROGDIR: path to the Turbomole, Dalton or Gaussian executable files (note that Adf commands will always be taken from the path which is set in your shell).
- ARPATH: path for archiving of mos files etc. This option has been disabled.

If you want to perform calculations with values for the variables cstep or scfconv, which are out of the confidence interval (Akiradefine will inform you if this happens), the entry

CRAP_OK yes

must be added to DCINPUT.

To create a standard inputfile, you can use the script `mkrdcinput`, which is automatically executed by AKIRADEFINE.

This file should be executed in the working directory. The settings for `DCPATH` and `TMPDIR` in this script must be adopted to the local settings before using it.

21.1.3 Preparations for the MPI versions

If one of the MPI versions shall be used, this can be done by the command

```
mpirun -machinefile MACHINES -np <number of CPU's> akira
```

The file `MACHINES` must contain the names of all computers available for the parallel calculation. You may create this file by running

```
choose_nodes -mpi
```

and then renaming the file `USE_NODES`, which is created by `choose_nodes`, to `MACHINES`. This is done automatically by AKIRADEFINE, which also creates the runscript `mpi.sub`. The runscript contains the `mpirun` command mentioned above; the number of CPU's is extracted from the files `USE_NODES`. Hence, after running AKIRADEFINE you may immediately run the MPI version of AKIRA using the command `mpi.sub`. Note that it might be necessary to modify the `mpi.sub` runscript, since different MPI installations may use different options.

Note that the exact syntax of the `MACHINES` file and the `mpirun` command depends on the MPI version. The syntax above corresponds to that of MPICH. OPENMPI, for instance, follows a slightly different syntax.

21.2 Parallelization standards

AKIRA uses raw data from several structures displaced from a reference structure along a set of basis modes, which start with a (or a couple of) user-defined basis vector(s). Each basis vector represents a collective distortion of the cartesian coordinates of the molecular (minimum) structure. The program executes two single-point calculations of electronic energies and their gradients for each basis vector employed in each iteration. Although parallelization is not as vital as in normal frequency analyses [3], the total wall clock time can be considerably reduced if these calculations are executed simultaneously in a coarse-grained parallel manner. In order to use the parallelized version of the data collection in AKIRA, it is necessary to provide the software for the parallelization standard MPI. Besides these, note that a *serial version of the program is also available.*

21.2.1 MPI

Different implementations of the MPI standard are available. OpenMpi, MPI, Mpich, and Lam/Mpi, resp., may be obtained from
`http://www.open-mpi.org`,
`http://www.erc.msstate.edu/misc/mpi/`,
`http://www-unix.mcs.anl.gov/mpi/mpich`, or
`http://www.lam-mpi.org`.
We use OpenMpi for development and, therefore, recommend its usage. Akira will look for a MPI Fortran compiler (like `mpif90`) and will use it for compilation.

21.3 Memory management and pre-processing

In the latest Akira version, program-specific setup tools and memory management constructs were reduced to a minimum. In particular, dynamic memory allocation is now handled by default Fortran90 `allocate/deallocate` calls.

Earlier versions of Akira used a memory managment that relied on the Memmgr module by B. A. Hess [83], which was also used in Snf [3]. The memmgr allows dynamical allocation of memory within Fortran routines which are written entirely in Fortran77. These versions also employed the preprocessor Delrem by B. A. Hess [84].

21.4 Monitoring Akira calculations

Most important to observe the progress of the calculation is the file `TMPdcstat` which contains information about the status of each slave process. The file can be watched during the calculation by executing the script `dcmore` which can be found in the scripts directory. It consists of 11 columns:

```
2   1 computer1   vbsy        0   90005   1 rdy      1    1 results sent
3   2 computer2               0   90005   1 rdy      2    2 results sent
4   3 computer3               0     103   0 run      3    3  968:15 dscf
5   4 computer4               0   90005   1 rdy      4    4 results sent
```

The rows contain the following entries:

1 task ID
2 continous numbering
3 node name
4 entry `vbsy` indicates a "very busy" machine
5 process ID (0 for finished processes)
6 cpu-load times 100, or
90005 for finished calculations, or
70005 for trouble in TURBOMOLE programs, or
80005 for wrong results.
7 number of finished single point calculations on this node
8 status:
`rdy` = ready for next step,
`run` = running calculation,
`wait` = node waiting
9 step number
10 step number (redundant)
11 cpu time and name of running program or message

As already indicated by the name of the data file, AKIRA has got a restart facility, such that any partly completed `restart.akira` file can be supplied in a AKIRA run, if the program has been aborted.

21.5 Further programs and scripts needed by AKIRA

As a meta-program, AKIRA starts standard quantum chemical packages on slave nodes in order to produce the raw data. The output produced on the slave nodes is then scanned in order to pick up all necessary information needed by AKIRA. Thus, AKIRA does not intermingle with any of these programs and you do need a license to run these programs, which you have to acquire separately according to the conditions of the particular vendor or theoretical chemistry groups, respectively!

For the performance of the single point calculations, the programs of the ADF [14], DALTON [85], GAUSSIAN [86] or TURBOMOLE [11] package, respectively, must be available on every node in `$PATH`. AKIRADEFINE allows to perform semiempirical MOPAC [18] calculations automatically as an initial guess for normal modes and Hessians. If this feature shall be used, MOPAC must be installed on your system. AKIRADEFINE furthermore requires MOLDEN [19] if the initial guesses selected for the mode-tracking calculations shall be visualized.

Furthermore, the scripts from the `scripts` subdirectory of the AKIRA installation directory must be available in `$PATH`. The script `choose_nodes` also requires the file `ELIGIBLE` in the directory `$CIPROC`, which contains the names of computers available and their numbers of processors (see Chapt. 23). An example file is provided with the AKIRA package. The variable `$CIPROC` must be set by the user. `choose_nodes` also requires the shell-command `ruptime` which comes with the `rwho` package. This is only necessary for the parallel version of AKIRA.

22. Methodology

The first section of this chapter explains the theoretical background of the subspace iteration methods employed in AKIRA. The next section deals with details of the numerical derivative methods.

22.1 Subspace iteration techniques

Two different subspace iteration techniques are available in AKIRA, namely the Davidson and Lanczos algorithms, which will be explained in the next subsections (compare also the reference to the original work given above).

22.1.1 Davidson algorithm

In order to calculate the vibrational frequencies, we have to solve the eigenvalue equation

$$\mathbf{H}^{(m)}\mathbf{Q}_k = \lambda_k \mathbf{Q}_k, \tag{22.1}$$

where $\mathbf{H}^{(m)}$ is the mass-weighted Cartesian Hessian, which contains the (mass-weighted) second derivatives of the total electronic energy with respect to nuclear Cartesian coordinates, and $\{\lambda_k, \mathbf{Q}_k\}$ is the eigensystem to be determined (with $\lambda_k \sim \omega_k^2$ and ω_k being the kth vibrational frequency; see [5, 87]).

The conventional procedure is to calculate *all* elements of the matrix $\mathbf{H}^{(m)}$ (either analytically or numerically) and to diagonalize this matrix to obtain all $3N$ eigenvalues and eigenvectors for a molecule containing N atoms. If only selected vibrations are of interest, one can apply subspace iteration methods like those by Lanczos [25] or by Davidson [23]. This has the major advantage that the full Hessian need not be calculated, which is the time limiting step in the standard procedure.

Our Davidson-type method starts with a collective displacement $\mathbf{b}$ of all atoms

$$\mathbf{b} = \sum_{j=1}^{3N} b_j \mathbf{e}_j^{(m)}, \tag{22.2}$$

where $\mathbf{e}_j^{(m)}$ are the $3N$ (mass-weighted) nuclear Cartesian basis vectors, and b_j are the components of the displacement. The kth elemof the vector $\boldsymbol{\sigma} = \mathbf{H}^{(m)} \cdot \mathbf{b}$, which is the

first approximation to the left-hand side of Eq. (22.1), is then given as

$$\sigma_k = \{\mathbf{H}^{(m)} \cdot \mathbf{b}\}_k = \sum_l \frac{\partial^2 E}{\partial R_l^{(m)} \partial R_k^{(m)}} b_l = \frac{\partial^2 E}{\partial R_k^{(m)} \partial \mathbf{b}}. \tag{22.3}$$

$\partial^2 E/[\partial R_l^{(m)} \partial R_k^{(m)}]$ is the second derivative of the total electronic energy with respect to (mass-weighted) nuclear Cartesian coordinates. This relation allows us to calculate the vector $\boldsymbol{\sigma}$ as a numerical directional derivative of the gradient of the total electronic energy E with respect to the collective displacement $\mathbf{b}$,

$$\boldsymbol{\sigma} = \mathbf{H}^{(m)} \cdot \mathbf{b} = \begin{pmatrix} \sum_l \frac{\partial^2 E}{\partial R_1^{(m)} \partial R_l^{(m)}} b_l \\ \sum_l \frac{\partial^2 E}{\partial R_2^{(m)} \partial R_l^{(m)}} b_l \\ \vdots \\ \sum_l \frac{\partial^2 E}{\partial R_{3N}^{(m)} \partial R_l^{(m)}} b_l \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 E}{\partial R_1^{(m)} \partial \mathbf{b}} \\ \frac{\partial^2 E}{\partial R_2^{(m)} \partial \mathbf{b}} \\ \vdots \\ \frac{\partial^2 E}{\partial R_{3N}^{(m)} \partial \mathbf{b}} \end{pmatrix}. \tag{22.4}$$

The vector $\boldsymbol{\sigma}$ can thus be calculated as the numerical derivative of the analytic gradients of the total energy. For this numerical differentiation it is necessary to carry out single point calculations for the along-$\mathbf{b}$ distorted structures such that n-point central difference formulae [88] for the numerical finite-difference approximation of the second derivative can be applied. For the generation of these distorted structures, we use displacements which result in a preselected norm of the corresponding (non-mass-weighted) Cartesian displacement vector; in general, a step size of 0.01 bohr proved to yield reliable and numerically stable derivatives [3, 22].

In the ith subspace iteration we build the Davidson matrix $\tilde{\mathbf{H}}^{(m),i}$ as

$$\tilde{\mathbf{H}}^{(m),i} = \mathbf{B}^{i\,T} \mathbf{H}^{(m)} \mathbf{B}^i = \mathbf{B}^{i\,T} \boldsymbol{\Sigma}^i \tag{22.5}$$

where all vectors $\mathbf{b}^l$ and $\boldsymbol{\sigma}^l$ (with $l = 1, \ldots, i$ and i being the actual iteration step) are collected in the matrices $\mathbf{B}^i$ and $\boldsymbol{\Sigma}^i$, respectively. We then solve the eigenvalue problem for the small Davidson matrix,

$$\tilde{\mathbf{H}}^{(m),i} \mathbf{c}_\mu^{(i)} = \lambda_\mu^{(i)} \mathbf{c}_\mu^{(i)}, \tag{22.6}$$

where $\lambda_\mu^{(i)}$ is the ith approximation for eigenvalue λ_μ, from which we can calculate approximate wavenumber in every iteration step. The desired eigenvector $\mathbf{c}_\mu^i$ is selected from the set of vectors obtained from Eq. (22.6) and the residuum vector reads

$$\mathbf{r}_\mu^{(i)} = \sum_{l=1}^{i} c_{\mu,l}^{(i)} \left[\boldsymbol{\sigma}^l - \lambda_\mu^{(i)} \mathbf{b}^l \right], \tag{22.7}$$

(note that i always denotes the actual ith iteration and μ marks the selected vector). The sum is over all basis vectors $\mathbf{b}^l$, and the number of basis vectors is increased in each iteration. In the standard Davidson method, the number of basis vectors is equal to the number of iterations, since in each iteration one new basis vector is introduced. For each new basis vector $\mathbf{b}^{i+1}$, we obtain a new vector $\boldsymbol{\sigma}^{i+1}$ as the numerical derivative of the gradient with respect to the collective displacement $\mathbf{b}^{i+1}$. The ith approximation $\mathbf{v}_s^i$ to the exact eigenvector $\mathbf{q}_s$ in Eq. (22.1) is obtained as

$$\mathbf{Q}_\mu^{(i)} = \sum_{j=1}^{i} c_{\mu,j}^{(i)} \mathbf{b}^j, \tag{22.8}$$

22.1.2 Preconditioning: The Davidson algorithm

The new basis vectors are generated from the residuum vectors,

$$\mathbf{b}^{i+1} = \mathbf{D}^{-1,(i)} \mathbf{r}_\mu^{(i)}, \tag{22.9}$$

where $\mathbf{X}^i = \mathbf{D}^{-1,(i)}$ is a preconditioner, which should ideally be as close as possible to $[\mathbf{H}^{(m)} - \lambda_\mu^{(i)} \mathbf{1}]^{-1}$. The simplest approximation for the inverse matrix of $[\mathbf{H}^{(m)} - \rho_s^i \mathbf{1}]$ is to use a diagonal preconditioner with diagonal elements $X_{jj} = 1/(H_{jj}^{(m)} - \rho_s^i)$. However, this is only a good approximation for diagonally dominant matrices, a condition which is fulfilled for configuration interaction matrices, but not for the Hessian matrices investigated here. This procedure is repeated until the convergence criterion drops below a predefined threshold. Convergence criteria are: i) the maximum element of the residuum vector, ii) the norm of the residuum vector, iii) the contribution $u_{s,i}^i$ of the latest basis vector i in Eq. (22.8) to the selected eigenvector and iv) the change in the wavenumber.

The convergence characteristics of this algorithm strongly depend on the reliability of i) the initial guess of the first basis vector $\mathbf{b}^1$, which is the first approximation to the desired exact eigenvector $\mathbf{q}_s$ and of ii) the preconditioner. The latter problem is delicate since we do not have any information about the matrix $\mathbf{H}^{(m)}$; only matrix–vector products $\boldsymbol{\sigma}^l = \mathbf{H}^{(m)} \mathbf{b}^l$ are known.

The Hessian may be approximated using the inverse transformation of Eq. (22.5)

$$\mathbf{H}^{(m)} = \mathbf{B} \tilde{\mathbf{H}}^{(m)} \mathbf{B}^T, \tag{22.10}$$

with $\mathbf{B} := \mathbf{B}^{3N}$. This transformation would thus only be exact if we used a complete set of $3N$ basis vectors. If the basis set is not complete, we may use the approximation

$$H_{nj,\text{appr.}}^{(m)} = \sum_{kl} \tilde{H}_{kl}^{(m),i} B_{kn}^i B_{lj}^i \tag{22.11}$$

for the default preconditioner, where the sum is over all basis vectors $\mathbf{b}^l$ $(l = 1, \ldots, i)$ stored in the matrix $\mathbf{B}^i$. But this is usually a poor approximation and yields only as many

approximate diagonal elements as basis functions are used in the current iteration (for the other diagonal elements, one could use either unit entries or the last diagonal element determined in this way for all other diagonal entries). However, the more iterations are needed, the better becomes the preconditioner in this default preconditioning scheme. Furthermore, $1/(H_{jj}^{(m)} - \lambda_\mu^{(i)})$ is a poor approximation to the inverse of a matrix $(\mathbf{H}^{(m)} - \lambda_\mu^{(i)}\mathbf{1})$ if $\mathbf{H}^{(m)}$ is not diagonally dominant. Consequently, this approach is in most cases not better than using a unit matrix as a preconditioner at the very beginning of the procedure, when only very few basis vectors are available.

Both problems mentioned above in connection with the convergence criteria can be overcome by using a semi-empirical calculation as an initial approximation: We calculate an estimate for the Hessian and approximate normal modes using the PM3 model (of course, other semi-empirical models can also be utilized). An initial guess for the eigenvector can be chosen from the set of semi-empirical normal modes, while the semi-empirical Hessian can be used for the preconditioning procedure. Since the Hessian matrices under investigation are of dimensions of about a few hundred rows and columns, it is — in contrast with configuration interaction matrices — possible to explicitly calculate the inverse preconditioner matrix

$$\mathbf{X}^i = \left[\mathbf{H}_{\mathrm{PM3}}^{(m)} - \rho_s^i\right]^{-1} \tag{22.12}$$

in each iteration. It should be emphasized that the bottleneck of the calculation is not this matrix inversion, which takes only a couple of seconds, but the single point calculations of electronic energies and gradients for the displaced structures. Using a 3-point central differences formula [88] for the numerical differentiation, we need two single-point calculations for structures distorted along each basis vector, which are performed in a coarse-grained parallelized way using standard parallelization techniques as provided by MPI. Unfortunately, it is not possible to perform *all* single-point calculations at once as the basis vectors of iteration i depend on the results of all $(i-1)$ former iterations. Therefore, the little computational effort for the generation of more accurate preconditioners is easily compensated by the resulting reduction of the number of iterations.

In course of the calculation of $\mathbf{H}_{\mathrm{PM3}}^{(m)}$, we also obtain the PM3 normal modes, which we use as the first approximation $\mathbf{b}^1$. Note that this 'guessing of normal modes' is different from the standard projection operator technique, which always requires a certain point group in order to set up the projector from the irreducible representations of this point group. Instead, we project out a selected mode and do not rely on any group theoretical tools. Consequently, our approach is applicable also in C_1-symmetric cases. Nevertheless, these projection operator techniques can be used to determine an initial guess for the desired normal modes.

22.1.3 The Jacobi–Davidson algorithm

Let us take another look at the preconditioning problem: In a mode-tracking calculation however, we iteratively solve the equation

$$(\mathbf{H} - \lambda_\mu^{(i)})\mathbf{Q}_\mu^{(i)} = \mathbf{r}_\mu^{(i)} \tag{22.13}$$

With the residuum vector, we want to construct the correction $\Delta\mathbf{Q}$ to the current eigenvector approximation with

$$(\mathbf{H} - \lambda_\mu)(\mathbf{Q}_\mu^{(i)} - \Delta\mathbf{Q}) = \mathbf{r}_\mu^{(i)} - (\mathbf{H} - \lambda_\mu)\Delta\mathbf{Q} = \mathbf{0}. \tag{22.14}$$

This suggest that

$$\Delta\mathbf{Q} = (\mathbf{H} - \lambda_\mu)^{-1}\mathbf{r}_\mu^{(i)} = \mathbf{D}^{-1}\mathbf{r}_\mu^{(i)}, \tag{22.15}$$

where we used the definition $\mathbf{D} = (\mathbf{H} - \lambda_\mu)$. Since the purpose of applying subspace iteration techniques is to avoid the calculation and/or storage of the full Hessian matrix, the matrix $\mathbf{D}$ is usually not known. The knowledge of this correction vector would allow us to use $\Delta\mathbf{Q}$ as the next basis vector, which should immediatly reduce the residual vector to zero.

The original Davidson procedure is mainly used in CI-type problems, where electronic energies are identified as eigenvalues of the CI-matrix. These matrices are strongly diagonally dominant, which means that an appropriate guess for them can be constructed by using the diagonal elements of the CI-matrix. Such guesses for $\mathbf{D}$ are very successful for preconditioning and usually lead to rapid convergence. A guess for the eigenvalue λ_μ is readily available from the last iteration.

As described in earlier work, the Hessian matrix of a system is usually not diagonally dominant, and furthermore, a calculation of all diagonal elements of the Hessian is not much less work than the calculation of the full Hessian. But as mentioned above in many cases it is possible to get a guess for the Hessian of the system from simpler calculations, like semi-empirical, force-field or small basis set calculations. In these cases it is possible to construct the matrix $\mathbf{D}^{-1}$ by direct inversion of the guess Hessian (minus the approximate eigenvalue),

$$\mathbf{D}^{-1} = (\mathbf{H}^{\text{guess}} - \lambda_\mu^{(i)}). \tag{22.16}$$

Note that here and in the following, we use the notation $\mathbf{D}$ also for guesses of the exact definition given above.

As pointed out by Sleijpen and Van der Vorst [24], the Davidson diagonalization scheme has great difficulties if the guess for the Hessian becomes too good. Imagine that we apply the exact Hessian for preconditioning; in that case, the new basis vector would be obtained as

$$\mathbf{b}^{(i+1)} := (\mathbf{H} - \lambda_\mu^{(i)})^{-1}\mathbf{r}_\mu^{(i)} = \mathbf{Q}_\mu^{(i)}, \tag{22.17}$$

where we used Eq. (22.13) for the second equality. This means that the better the approximation for the Hessian is which we employ for preconditioning purposes, the smaller

will be the angle between the new basis vector and the old eigenvector approximation. In the limit of the exact Hessian, no improvement at all will be obtained.

It is, however, difficult to decide what will happen in a practical mode-tracking calculation, since the new basis vectors are always orthogonalized to all preceeding basis vectors, which should eliminate the problem of linear dependencies. And even for an exact Hessian, the numerical differentiation of the electronic gradient will introduce some numerical noise, so that always a component orthogonal to the current basis vectors will be present. This component might, however, be very small, and therefore we employ a cyclic procedure of Gram–Schmidt-orthogonalizations and orthogonality checks in our program in order to ensure orthogonality even in problematic cases. Whether the preconditioning is still efficient in those cases is a question which is investigated in the next section, since the orthogonal component of our (non-orthogonalized) new basis vector might just consist of numerical noise.

Sleijpen and Van der Vorst [24] proposed a Jacobi–Davidson diagonalization scheme, which automatically restricts the new basis vector to the subspace orthogonal to the current approximation $\mathbf{Q}_\mu^{(i)}$. This is achieved by using the orthogonal projection of the matrix to be diagonalized onto this subspace. Again, usually neither the matrix itself nor this orthogonal projection is available. Therefore, they suggest a one-step approximation to find a new basis vector if a guess for the matrix is available,

$$\mathbf{b}^{(i+1)} := \epsilon \mathbf{D}^{-1}\mathbf{Q}_\mu^{(i)} + \mathbf{D}^{-1}\mathbf{r}_\mu^{(i)}, \tag{22.18}$$

where

$$\epsilon = \frac{\mathbf{Q}_\mu^{(i)}\mathbf{D}^{-1}\mathbf{r}_\mu^{(i)}}{\mathbf{Q}_\mu^{(i)}\mathbf{D}^{-1}\mathbf{Q}_\mu^{(i)}} \tag{22.19}$$

The last equation is determined by the requirement that $\mathbf{b}^{(i+1)}$ is orthogonal to $\mathbf{Q}_\mu^{(i)}$. This should definitely fix the problem in Eq. (22.17), which might occur for the Davidson algorithm.

22.1.4 The Lanczos algorithm

The implementation of the Lanczos-type algorithm is very similar to the Davidson-diagonalization scheme. The first step is again the calculation of the vector $\boldsymbol{\sigma}^i$, Eq. (22.4), by numerical differentiation of elements of the gradient vector, calculated for structures perturbed along the basis vector $\mathbf{b}^i$. In the next step, the diagonal elements of the small Hessian matrix for the subspace are calculated,

$$d_i = \boldsymbol{\sigma}^{\boldsymbol{i},\boldsymbol{T}}\mathbf{b}^i, \tag{22.20}$$

(note that the Lanczos algorithm is essentially a method to create a tridiagonal matrix

$$\mathbf{H}^{(m),\text{tridiag}} = \mathbf{B}^T\mathbf{H}^{(m)}\mathbf{B} = \begin{pmatrix} d_1 & t_1 & 0 & 0 & \cdots \\ t_1 & d_2 & t_2 & 0 & \cdots \\ 0 & t_2 & d_3 & t_3 & \cdots \\ 0 & 0 & t_3 & d_3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{22.21}$$

from the original matrix $\mathbf{H}^{(m)}$). With these quantities, the vector

$$\mathbf{x}_{i+1} = \boldsymbol{\sigma}_i - d_i\mathbf{b}_i - t_{i-1}\mathbf{b}_{i-1}, \tag{22.22}$$

is calculated, which in turn determines the elements t_i via

$$t_i = |\mathbf{x}_{i+1}|, \tag{22.23}$$

and the new basis vector,

$$\mathbf{b}_{i+1} = \frac{\mathbf{x}_{i+1}}{|\mathbf{x}_{i+1}|}. \tag{22.24}$$

Note that $t_0 = 0$ in the first iteration. Furthermore, the new basis vector $\mathbf{b}_{i+1}$ is usually explicitly orthonormalized to the set of all previous basis vectors to avoid (near-)linear dependencies. The disadvantage of the Lanczos algorithm is that the new basis vector is in no way preconditioned for better convergence of the eigenvector selected for optimization. Therefore, the convergence characteristics is usually better for the Davidson diagonalization, unless only very poor preconditioners are available (see below).

Calculation of approximate force constants, wavenumbers and normal modes in each iteration are carried out in exactly the same manner as for the Davidson diagonalization. This also holds for the residuum vectors, which are only necessary for convergence control in the case of the Lanczos method.

22.1.5 Details of the implementation

We have implemented the above-described subspace iteration with a Davidson, a Jacobi–Davidson, as well as a Lanczos solver in the AKIRA program. A comparison of both diagonalization schemes shows that they perform equally well if the preconditioning is not well chosen (see below). But, in case of a good preconditioning through a PM3 or similarly sophisticated guess, we obtain a significantly better convergence of the Davidson-type algorithm. Our implementation allows one to optimize several eigenvectors simultaneously, which is known as the *Davidson–Liu* or *block-Davidson* method [89, 90]. Root homing is also guaranteed [91]. For root homing, there exist two promising protocols in the case of normal modes as eigenvectors: i) selection of the eigenvector with the largest overlap with

the initial guess vector; ii) selection of the eigenvector with the largest overlap with the approximate eigenvector chosen in the last iteration. Both methods are implemented in our program. While the first method can cause convergence problems if only a poor initial guess vector is available, the second method usually shows better convergence characteristics; however, it may converge to a different, non-desired eigenvector due to poor initial vectors in combination with some preconditioners (see [21] for examples) Furthermore, it is possible to select the vector with minimal residuum or to optimize the lowest root.

22.2 Generation of displaced structures

We use displacements along the mass-weighted basis vectors $\mathbf{b}^i$, for which the energies and gradients have to be calculated. A displacement along the mass-weighted basis vector of $\Delta\mathbf{b}^i$ gives rise to a displacement in Cartesian coordinates of

$$\Delta\mathbf{R}^i = \Delta\mathbf{b}^i\mathbf{M}^{-1/2}, \tag{22.25}$$

where $\mathbf{M}^{-1}$ is a diagonal matrix with elements $M^{-1}_{ij} = \delta_{ij}/m_i$; m_i is the mass of the atom corresponding to the Cartesian coordinate R^i_j.

22.2.1 Displacements in units of length

Test calculations have shown that the step size s_R should be chosen such that the norm of the Cartesian displacement vector is $s_R|\Delta\mathbf{R}^i| \approx 0.01$ bohr. The displacement may be re-written in terms of normalized displacement vectors for both the Cartesian and the mass-weighted basis modes,

$$s_{Q_k}\Delta\mathbf{b}^{i,\mathrm{norm}} \mathrel{\hat{=}} s_{Q_k}\Delta\mathbf{R}^i = s_{Q_k}|\Delta\mathbf{R}^i|\Delta\mathbf{R}^{i,\mathrm{norm}} = s_R\Delta\mathbf{R}^{i,\mathrm{norm}} \tag{22.26}$$

which leads to a step size s_{b^i} for the numerical differentiation of

$$s_{b^i} = s_R/|\Delta\mathbf{R}^i| = s_R\left(\sum_{j=1}^{3N}(b^{i,\mathrm{norm}}_j)^2/m_j\right)^{-1/2}\left(\frac{[\text{unit of length}]}{[\text{unit of mass}]^{1/2}}\right). \tag{22.27}$$

The inclusion of the units is necessary in order to keep both coefficients dimensionless (compare also Ref. [51]). Note that for a given value of s_R, the coefficient s_{b^i} may have different values for different normal coordinates b^i.

22.2.2 Calculation of second derivatives

The calculation of the elements of the $\boldsymbol{\sigma}$ vector, Eq. (22.3), can be accomplished by calculating numerical derivatives of the components of the analytic gradient w.r.t. the

collective displacements along the basis vectors $\mathbf{b}^i$,

$$\frac{\partial^2 E}{\partial R_k^{(m)} \partial \mathbf{b}}. = \frac{1}{2 s_{b^i}^2 |\Delta \mathbf{b}^{i,\mathrm{norm}}|} \left[g_k^{(m)}(+s_R \Delta \mathbf{R}^{i,\mathrm{norm}}) - g_k^{(m)}(-s_R \Delta \mathbf{R}^{i,\mathrm{norm}}) \right] \tag{22.28}$$

where the $g^{(m)}$ are the mass-weighted components of the gradient vector for the along-$\mathbf{b}^i$ displaced structures.

23. Running the calculation

In this chapter, the steps necessary to carry out a mode-tracking calculation using Akira will be described in detail. Starting from the preparations, we will discuss the setup of the calculation using Akiradefine, the mode-tracking calculation itself, and possible restart and re-evaluation runs.

Running a mode-tracking calculation with Akira requires two separate steps: (I) choosing the initial guess vector(s) with Akiradefine and (II) running the mode-tracking calculation with Akira.

23.1 Preparations

23.1.1 Structure optimization

Since Akira tries to determine the vibrational frequencies and normal modes from the eigenvalues and eigenvectors of the Hessian matrix, and since these quantities have, in a strict sense, a well-defined meaning only for structures for which the electronic energy gradient is zero, it is necessary to perform a geometry optimization first. The mode-tracking calculation must then start from the optimized structure. A large number of test calculations with Snf (see Introduction) has shown that the maximum component of the gradient should be smaller than 0.0001 a.u. in order to obtain accurate results for the vibrational frequencies.

23.1.2 Electronically excited states

From Akira 2.1.0 on, mode-tracking calculations can be performed for electronically excited states, provided Turbomole (version 5.6 or higher) is used for the single-point calculations. Of course, for the reasons mentioned in Section 23.1.1, the molecular structure has to be optimized in the chosen excited state before starting the Akira program. In this case, no further Akira-specific settings have to be made, since Akira will recognize the excited-state keywords in the Turbomole control file and use these settings automatically for the single-point calculations of the distorted structures on the slave nodes.

23.1.3 Molecular symmetry

AKIRA uses a semi-numerical algorithm in which second derivatives of the electronic energy are calculated as numerical first derivatives of analytical energy gradients. Therefore, the algorithm will create structures which are displaced from the equilibrium structure along some basis vectors. In almost all cases, these displacements will lower the molecular symmetry, which can cause problems if the original calculation explicitly used the higher symmetry, since the MOs of the originial equilibrium structure are in some cases used by AKIRA as an initial guess for the MOs of the displaced structures.

For TURBOMOLE users: If the molecule under investigation is of higher symmetry than the trivial C_1 point group symmetry, the user should run a single-point energy calculation without taking advantage of the symmetry in order to provide C_1-symmetric MOs as start-MOs for the perturbed structures of the molecule. As an alternative, you may provide any other initial guess for the MOs (which can, e.g., be generated by TURBOMOLE's `define` program).

There are indeed a few special cases in which the calculations for the displaced structures can be performed using a higher symmetry: If the basis vectors given in the setup preserve a higher symmetry than C_1, and if also all new vectors generated in the mode-tracking calculation are guaranteed to keep this symmetry, then also each single-point calculation can indeed use this higher symmetry. To give the most important example: If you use AKIRA not to track a specific vibration, but to calculate all vibrations in a certain irrep, then it is possible to restrict the calculation to basis vectors of this irrep. If you want to calculate, e.g., all vibrations in the totally symmetric irrep of the molecule, than all displaced structures will also exhibit the original symmetry. In that case, the original MOs can be used as a guess. Another example is the totally symmetric breathing mode of the buckminsterfullerene C_{60}, for which the calculations for the perturbed structures can also be done in I_h symmetry. In this case, only one basis vector is needed to achieve convergence (see [21]).

It is *not necessary* to provide MOs in the case of ADF, DALTON or GAUSSIAN single point calculations.

23.2 AKIRADEFINE: setting up the calculation

For an easy preparation of the AKIRA calculation, most of the input/output and program options can be controlled and set via the interactive setup tool AKIRADEFINE. All steps done by this program are explained here in detail. If your `PATH` variable contains the instalation directory, you can start AKIRADEFINE simply by typing:

```
akiradefine
```

The information which is collected by AKIRADEFINE is written to the file `akira_control`
.

23.2.1 Program-specific input

Depending on whether TURBOMOLE, ADF or GAUSSIAN has been used for the molecular structure optimization, AKIRA can perform the single-point calculations for the displaced structures with different programs. If the geometry has been optimized with TURBOMOLE, it is possible for the user to request either a TURBOMOLE calculation (HF, DFT, RI-DFT, MP2, RI-MP2) or a DALTON calculation (HF, MP2, CC2, CCSD, CASSCF) for the perturbed structures. In case of a GAUSSIAN geometry optimization, GAUSSIAN is always employed in the single-point calculations. and similarly, in case of ADF optimizations, AKIRA will automatically use ADF for the perturbed structures.

TURBOMOLE input files

If TURBOMOLE or DALTON are to be used for the single-points subsequent to a TURBOMOLE structure optimization, all files which are necessary for TURBOMOLE single-point calculations must be provided (i. e., `control`, `coord`, `mos` or `alpha` and `beta`, `basis` and, for RI-accelerated calculations, `auxbasis`).

GAUSSIAN input file

If GAUSSIAN shall be used for the single-point calculations, the user must provide a GAUSSIAN input file named `akira.com` which contains the keyword `#p force`, the keywords for the method and the basis set to be used, information on molecular charge and spin state and the equilibrium geometry in cartesian coordinates. For an ethanol molecule, the `akira.com` file might look like this (with the last line being a blank line, of course):

```
#p force  bp86/TZVP

ethanol

0 1
C -1.225747   -0.225340    0.000002
H -1.296072   -0.865520   -0.891479
H -1.296085   -0.865499    0.891496
H -2.083095    0.464915   -0.000013
C  0.079992    0.551234   -0.000000
H  0.131474    1.203598    0.893871
H  0.131470    1.203596   -0.893875
O  1.161283   -0.399398   -0.000003
H  1.996579    0.098737    0.000020
```

This file is also contained in the directory `examples/files`.

ADF input file

Since AKIRA will normally be run after performing a geometry optimization, it is designed to reuse simply the ADF script which has been used to run the geometry optimization. This script has to fulfill several conditions:

1. It must be named `adf.in`.
2. It be executable.
3. It must contain exactly the settings used in the preceding geometry optimization.
4. The geometry must be specified in cartesian coordinates *using Angstrøm*
5. The binary output file must be named `TAPE21`.
6. It should not contain any blocks requesting other calculations than a geometry optimization.

AKIRADEFINE will check (and modify, if necessary) the following blocks in the `adf.in` file:

- `GEOMETRY` — delete it (if present) and write instead:

  ```
  Geometry
    GO
    iterations 1
  End
  ```

- `INTEGRATION` — if the grid settings are below `6.0 6.0`, replace them by `6.0 6.0`.
- `SCF` — if the convergence criteria are below `converge 1.0e-6 1.0e-6`, replace them by `converge 1.0e-6 1.0e-6`.
- Sort the list if cartesian coordinates according to their nuclear charge. (This will facilitate reading in the data of the distorted structure-single point calculations in the AKIRA run.) **Attention: the reordering of atoms will affect the choice of internal coordinates in AKIRADEFINE as well as the usability of ADF restart files!**

If AKIRADEFINE modifies the integration or convergence parameters, it will print a warning message to the screen remembering the user to redo a geometry optimization with the new settings.

Starting from version 3.1.0, AKIRA can deal with QM/MM gradients. Furthermore, it is now possible to provide an ADF restart file, which is distributed onto the slave nodes in

order to accelerate the single-point calculation for the displaced structures. Based on the keywords in the `adf.in` input file, AKIRAEFINE will recognize automatically whether a QM/MM calculation is requested or whether a restart file shall be copied onto the slaves. In case an ADF restart file shall be used, the user must necessarily specify the option `nogeo` (like in the example below), since otherwise, all displaced structure calculations will be performed for the geometry given in the restart file, thus leading to nonsense results. It is not possible to use ADF restart files in combination with QM/MM calculations.

The `adf.in` input file you provide might look like this:

```
#! /bin/sh

$ADFBIN/adf -n1 << eor
Create O   $ADFRESOURCES/TZP/O.1s
End Input
eor
mv TAPE21 t21.O

$ADFBIN/adf -n1 << eor
Create H   $ADFRESOURCES/TZP/H
End Input
eor
mv TAPE21 t21.H

$ADFBIN/adf << eor
Title  H2O

INTEGRATION 6.0  6.0

Atoms
    O        -0.004404    0.000000   -0.003115
    H         0.038319    0.000000    0.969951
    H         0.927252    0.000000   -0.287190
End

Fragments
  O t21.O
  H t21.H
End

Geometry
  GO
```

```
End

STOPAFTER GGRADS

RESTART restartfile &
  nogeo
END

symmetry nosym

XC
 LDA VWN
End

savefile TAPE21

SCF
 converge   1.0e-6  1.0e-6
End

end input
eor
```

This example can also be found in the `examples/files` directory.

23.2.2 Program selection

AKIRADEFINE first of all will ask for the user's preferences concerning the quantum chemical program package which shall be used in the single-point calculations of the distorted structures:

```
 Which program would you like to use ?

tm        : use TURBOMOLE for single points
d[alton]  : use DALTON for single points
g98       : use GAUSSIAN 98 for single points
g03       : use GAUSSIAN 03 for single points
a[df]     : use ADF for single points

(<return> = default = tm)
```

The default program for the single-points is TURBOMOLE. You can select another program package by typing the abbreviation for this package. For example, `g03` will

select GAUSSIAN03 for the single-point-calculations.

AKIRADEFINE then checks the existence of he TURBOMOLE input files named `control` in case of TURBOMOLE or DALTON single-point calculations, `akira.com` in case of GAUSSIAN single-point calculations and `adf.in` if ADF shall be used. If the molecule under study has been optimized with GAUSSIAN or ADF and thus no TURBOMOLE `control` and `coord` files are present, AKIRADEFINE will generate a fake TURBOMOLE `control` and `coord` file from the parameters found in `akira.com` or `adf.in`, respectively.

AKIRADEFINE then reads `control` and `coord` and prints the general molecule information as well as information about the thresholds employed in the electronic structure calculations.

In the next menu, an overview of the AKIRA input parameters is given, which may be changed by the user:

```
Program settings menu:
----------------------
Current settings:
cstep    =     0.01000000
numderiv =     3
scfconv  =     8
maxnbm   =    21
mtype    =     1
tmpcl    = off
logcl    = off

Choose one of the following commands:
-------------------------------------
 cstep <real>          : set cstep [<real> in bohr]
 maxnbm <int>          : choose max. no. of basis modes
 numderiv <int>        : choose <int> point central differences bickley formula
 scfconv <int>         : choose scfconv parameter
 tmpcl                 : switch on/off removal of temporary directories
 logcl                 : switch on/off removal of log directories
 mtype <int>           : choose type of masses to be used
                         (1: most abundant isotopes, 2: average atomic masses,
                          3: most abundant isotopes, but deuterium mass for hydrogen)
 <return>              : leave this menu
```

The parameters which can be modified in this menu are the following:

`cstep`: step size for the displacements from the equilibrium structure (given as the norm of the displacement vector in [bohr]; default = 0.01, larger values are recommended for low-frequency modes).

`maxnbm`: maximum number of basis vectors for this calculation. This number cannot be chosen larger than $3N$, the number of degrees of freedom for this molecule (which would correspond to a full harmonic force field calculation).

`numderiv`: number of grid points used for the numerical differentiation (default = 3).

`scfconv`: SCF convergence threshold parameter (cf. TURBOMOLE manual); `scfconv` ≥ 8 is strongly recommended.

`tmpcl`: temporary directories on slave nodes are removed after the calculation (default = off).

`logcl`: log files for slave node calculations are removed after the calculation (default = off).

`mtype`: Type of masses to be used in the calculation (1 = masses of most abundant isotopes, 2 = average atomic masses (not recommended), 3 = masses of most abundant isotopes but deuterium mass for hydrogen atoms). Note that it is not yet possible in AKIRA to specify individual atomic masses for all atoms in the molecule.

23.2.3 Convergence criteria menu

In this menu, the thresholds for the convergence checks can be set:

```
Convergence critera menu:
-------------------------
current thresholds:
|max. component of residuum vector| :     0.00050000
change of max. component of r       : 0.50000000E-07
orthonormalization parameter        :              8

choose one of the following commands:
 rthres <real>     : set threshold for max. component of residuum vector
 rabsthres <real>  : set threshold for norm of residuum vector/(3*natoms)
 dvthres <real>    : set threshold for change in wavenumber [1/cm]
 ecthres <real>    : set threshold for expansion coef. of last basis vector
 rchthres <real>   : set threshold for change of max. component of r
 iortho <int>      : set orthogonalization parameter
 d                 : default settings
 &                 : go back to program settings menu
 <return>          : leave this menu
```

There are several different thresholds to control the convergence of the results:

`rthres`: maximum component of the residuum vector (default = 0.0005)

`rabsthres`: norm of the residuum vector divided by the number of degrees of freedom of the molecule, i.e., $3N$.

`dvthres`: change in wavenumber between two subsequent calculations

`ecthres:` expansion coefficient for last basis vector

`rchthres:` change in the maximum component of the residuum vector

`iortho:` set orthogonalization parameter

While the first four criteria are direct measures for the quality of the eigenvectors or eigenvalues of the Hessian, the last value, `rchthres` is a measure for the improvement of the eigenvector between two subsequent iterations. If this improvement is smaller than this threshold, the algorithm assumes that no further improvement for this vector is possible and stops the iterations for this vector. This can often be traced back to wrong setups in the calculation and does *not* mean that the vector is converged. If a very tight threshold for the residuum has been selected, it may, however, happen that this convergence criterion cannot be fulfilled: Due to numerical noise, it will never be possible to reduce the residuum vector to zero. Besides the convergence criteria, it is also possible to select a threshold for the orthonormality check in AKIRA, the parameter `iortho`. This parameter determines the threshold for the orthonormality of the basis vectors employed in the subspace iteration ($10^{-\texttt{iortho}}$). AKIRA uses a Gram–Schmidt orthogonalization to keep new basis vectors orthogonal to previous ones. For some preconditioners, the new basis vectors determined by the Davidson procedure will be almost parallel to the set of basis vectors in use. In those cases, up to `iortho` subsequent orthonormalization cycles and orthonormality checks will be performed to fulfill the orthonormality threshold. If this cannot be achieved, the program will stop with an error message since it is not possible to find a new, linearly independent basis vector.

23.2.4 Output menu

In this menu, you can modify the output of the normal mode calculated by AKIRA and switch on/off a more detailed output on the subspace iterations:

```
Output menu:
------------
choose one of the following commands:
 g98it off         : switch off g98 normal mode output
 g98bs off         : switch off g98 basis vector output
 tmout on          : switch on  TM output of final normal modes
 xmout on          : switch on  xmol output of final normal modes
 prall on          : switch on  detailed output of subspace iteration
 d                 : default settings
 &                 : go back to program settings menu
 <return>          : leave this menu
```

It contains the following options:

`g98it on/off` switch output of normal mode approximations in G98 format after each iteration on/off. Creates files `g98.out.it`n for each iteration n.

`g98bs on/off` switch output of basis modes (vectors) in G98 format after each iteration on/off[1]. Creates files `g98.out.bs`n for each iteration n.

`tmout on/off` switch output of converged normal modes in TURBOMOLE format on/off. Creates file `tm_normmodes` after convergence.

`xmout on/off` switch output of converged normal modes in XMOL format on/off. Creates file `xmol.XYZ.out` after convergence.

`prall on/off` switch detailed output of subspace iteration on/off.

The following menu will be discussed in a separate section (Sec. 23.3), since it is the most important preparation step in a mode-tracking calculation: the selection of the initial basis vectors. In this section, we proceed with a description of the following menus.

23.2.5 Hessian guess and rigid modes

After the selection of initial basis modes, which is described in Sec. 23.3, you have to decide whether the basis vectors should be kept orthogonal to translational and rotational ("rigid") modes of the molecule:

```
Do you want to orthogonalize your guess vectors
to translational and rotational modes ?
(y/n, default = yes)
```

This is always recommended, since it reduces the computational cost and increases the accuracy of the calculation. However, it might be desired for test purposes to omit the orthogonalization on these rigid modes, or if, e.g., the full cartesian Hessian shall be calculated explicitly.

If a Hessian guess is available from the "Initial guess selection", AKIRADEFINE will suggest to use this guess for preconditioning purposes.

23.2.6 Root homing and preconditioning

After this input step follows the "Root homing and preconditioning menu"ndexroot homing menu:

[1] Wave numbers do not have a physical meaning in this case.

```
Root homing and preconditioning menu:
-------------------------------------
current settings:
root homing by overlap with selected vector of previous iteration
preconditioning scheme: backtra
   nbt                :       6

choose one of the following commands:
 root <string>     : set root homing scheme; possible values:
                     lastsel = default; overlap with last eigenvector
                     trackgv = overlap with guess vector
                     minresi = choose vector with minimal residuum
                     testmin = test for minimum; optimize lowest root
 prec <string>     : set preconditioner; possible values:
                     backtra = backtransformation of Davidson Matrix
                     unitmat = Davidson with unit matrix
                     lanczos = Lanczos type diagonalization
 nbt  <int>        : min. no. of vectors for backtransformation
 genstart on       : only one iteration for generation of start vectors
 d                 : default settings
 &                 : go back to convergence criteria menu
 <return>          : leave this menu
```

For root homing, the following choices can be made:

`lastsel`: default; the program selects as a new vector to be optimized that eigenvector approximation which has the largest overlap with the last vector selected.

`trackgv`: the vector with the largest overlap with the initial guess vector will be chosen for further optimization.

`minresi`: the vector with the lowest residuum will be further optimized, irrespectively of its nature, i.e., the type of vibration involved.

`testmin`: the vector with the lowest eigenvalue will be selected in order to test for a minimum or saddle point.

Note that for each basis vector given in the first iteration, one vector for further optimization will be selected.

For preconditioning, the following options are available:

`backtra`: backtransformation of the Davidson Matrix according to Eq. (22.11). Default if no approximate Hessian is available.

`unitmat`: a unit matrix preconditioner will be applied.

`lanczos:` test option: Lanczos algorithm. Only one vector should be selected in this case, since the block-Lanczos algorithm, which allows to optimize several vectors simultaneously, is not yet tested.

If an approximate Hessian matrix is available, three additional options will appear on the screen:

`jacobid:` use Jacobi–Davidson diagonalization scheme. Default if approximate Hessian is available.

`aphsinv:` Davidson algorithm in which the approximate Hessian will be explicitly inverted to get the preconditioner. Yields similar performance as Jacobi–Davidson.

`aphsdia:` test option: original Davidson algorithm in which the inverse diagonal elements are used as a preconditioner. Cannot be recommended for calculations, since this assumes that the matrix to be diagonalized is diagonally dominant, which is in general not the case for Hessian matrices.

Note that the first three options can be selected irrespectively of the availability of an approximate Hessian, while the latter three require such a guess (and will thus not appear on the screen if no approximate Hessian is present). The `jacobid` and `aphsinv` options usually lead to the best convergence, if appropriate guess Hessians are available. The `backtra` option sometimes shows bad performace during the first iterations, even worse than the most simple unit matrix preconditioner. Therefore, it is possible to use a combined approach in which the first n iterations will employ a unit matrix preconditioner, while in the following iterations the backtransformation will be used. The number n can be set with the keyword `nbt`.

Note that experience with these options is still somewhat limited. It appears, however, that no big differences occur between the `unitmat` and `backtra` preconditioning schemes.

One last option, `genstart on/off`, can be selected to do exactly one iteration. After this iteration, you can use one of the vectors created in this cycle as a better approximation for the desired normal mode.

23.2.7 Additional keywords

Some keywords can only be introduced manually into the file `akira_control`. At the moment, the following options are available:

`sleeptime <time-in-seconds>:` sets the time in seconds to wait for the master process between two subsequent checks of the slave processes. Default: 5. Smaller values decrease the overhead, in particular in fast calculations, larger values can be helpful to decrease the network traffic when running the master job via a network file system, in particular for large calculations with long single-point calculations.

23.2.8 Data collector options

The next and last step of AKIRADEFINE creates the files necessary for the parallel execution of the data collection:

```
  ===================================
  Input completed, calling prepdc ...
  ===================================

  ELIGIBLE file to be used for choose_nodes call:
 $CIPROC/ELIGIBLE

  Specify name of alternative ELIGIBLE file or press <return>
```

The file `DCINPUT` will be created by calling the script `mkrdcinput`. The `DCINPUT` file can be further modified manually afterwards, see Section 21.1.2 for details. A directory for logfiles, named `log`, will be created. The logfiles contain valuable information on what is going on on the slave nodes.

23.3 AKIRADEFINE: Setting up the initial guess

While AKIRA itself can be used mainly as a black-box program, the setup of the initial guess must be done by the user, and it requires some idea of the specific problem to be analyzed. If there is no such specific problem, it is probably not useful to run a mode-tracking calculation, since either all normal modes are wanted to get an overview, or no modes at all are of interest.

If, however, the problem is identified, AKIRADEFINE offers a lot of help to construct an appropriate guess for the modes to be studied in the initial basis vector menu:

```
 Selection of initial basis vectors
  ----------------------------------
  No. of basis modes selected:  0
  choose one of the following commands:

   ---------------------------------------------------------------
   pint menu  :  show internal coordinate options
   ---------------------------------------------------------------
   psym menu  :  show symmetry coordinate options
   ---------------------------------------------------------------
   pcart menu :  show cartesian coordinate options
   ---------------------------------------------------------------
   pfile menu :  show basis-vector-from-file options
   ---------------------------------------------------------------
   pmop menu  :  show basis-vectors-from-MOPAC options
```

```
-------------------------------------------------------------
trarot     :  create translational and rotational modes
submenu    :  enter subsystem menu
<return>   :  quit this menu
```

By typing `xxx menu`, you can display the menu `xxx`, and by typing `xxx hide`, you can hide it again. In the following, the individual menus are explained in detail.

23.3.1 Internal coordinates

In many cases, the user might be interested in a local vibration of the molecule, like a stretching mode of a particular bond between two atoms, or a valence angle bending. In the internal coordinate submenu which can be displayed by typing `pint menu`, AKIRADEFINE offers the following possibilities to select such modes:

`str i j` select a stretching mode between two atoms

`bd i j k` select a bending mode between three atoms

`tor i j k l` select a torsional mode between four atoms

`oop i j k l` select an out-of-plane mode, characterized by four atoms

`br` select a totally symmetric breathing mode of the molecule

`brm` select a mass-weighted totally symmetric breathing mode of the molecule

23.3.2 Symmetry coordinates

In some cases, also symmetry coordinates may be a good starting point for the subspace iteration: Indeed, they sometimes completely determine the vibrational modes.

```
sym        :  generate symmetry coordinates as basis vectors
spesym     :  generate normal modes for special symmetries
psym hide  :  hide symmetry coordinate options
```

Some special cases are included with the `spesym` option, while general symmetry coordinates for practically every point-group symmetry can be constructed with the `sym` option. Note that in AKIRADEFINE does not check the symmetry, which the user has to specify manually. However, giving a wrong input for the symmetry of your molecule will only result in an empty set of symmetry coordinates.

All symmetry coordinates for the chosen irrep of a particular symmetry will be put in a buffer, and may be selected from there as basis vectors in the mode-tracking calculation (see section 23.3.6).

23.3.3 Cartesian coordinates

A very simple way to construct a guess for a normal coordinate is simply to select a Cartesian nuclear coordinate. This might be useful if you want to investigate *all* vibrations of a molecule in which a particular atom or a set of atoms is involved. Since this is usually not very specific, especially when a large number of cartesian coordinates is selected, we recommend to use the genstart option (Sect. 23.2.6) or a similar procedure: With that option, the calculation stops automatically after the first iteration (while it has to be stopped manually after several iterations). After *every* iteration, an output of the approximate normal modes is performed[2]. If you specified several cartesian basis vectors as initial guesses, then the first iteration will produce linear combinations of these basis vectors, which are already to a certain extend adapted to the specific molecule. These might be much better approximations for the normal modes you are looking for than the cartesian basis vectors themselves, and the interesting linear combinations can be selected as basis vectors in a second mode-tracking calculation.

```
at <i>     :  select all cartesian basis vectors for atom i
atx <i>    :  select cartesian x-basis vector for atom i
aty <i>    :  select cartesian y-basis vector for atom i
atz <i>    :  select cartesian z-basis vector for atom i
aatz       :  select all cartesian z-basis vectors
cfull      :  select all cartesian basis vectors
              (full frequency analysis)
pcart hide :  hide cartesian coordinate options
```

The command `cfull` can be used to specify a full frequency analysis. In that case, all $3N$ cartesian normal modes will be taken as basis vectors. The rigid motions will be removed by orthogonalization, so that only $3N-6$ (for non-linear molecules) basis vectors are employed.

23.3.4 Normal coordinates from file

As mentioned earlier in this manual, results from cheap low-level calculations, like force-field, semiempiric, or small-basis set calculations might already be available for your molecule. For such cases, it is possible to simply read the output files of some molecular mechanics or quantum chemistry programs as an input for AKIRA.

```
re  <i>    :  read <i> basis vectors from file
resub <i>  :  read <i> basis vectors for subsystem from file
pfile hide :  hide basis-vector-from-file options
```

`re` can be used to read normal modes from a file which has to be specified later. It is also possible to use the `resub` command and read normal modes for a subsystem. In that case, it is necessary to specify the number n of atoms in this subsystem:

[2]If not explicitly suppressed by the user.

```
Enter No. of atoms in subsystem:
(the first <No.> atoms will be taken !!!)
```

It is always assumed that the first n atoms belong to the chosen subsystem. General subsystems can be handled by the subsystem menu submenu, Sect. 23.3.5. Next, AKIRADEFINE will ask the user for the name of the file which contains the normal modes to be read in as initial basis vectors,

```
Enter filename:
```

Then, the format of these basis vectors can be selected:

```
Select format of basis vectors
------------------------------

  snf      : SNF normal modes (massweighted MOPAC)
  tmn      : TURBOMOLE normal modes
  tmb      : Basis vectors in TURBOMOLE format
  tmi      : Internal coordinates in TURBOMOLE format
  g98      : Gaussian98 normal modes
  mop      : MOPAC normal modes (not massweighted)
  pcm      : PCM normal modes (massweighted)
  adf      : ADF normal modes (not massweighted)
  free     : free format modes (not massweighted)
  <return> : return to main menu
```

AKIRADEFINE contains routines to read the following input formats:

- SNF - normal modes (massweighted MOPAC style)
- TURBOMOLE - normal modes
- GAUSSIAN98 normal modes (Gaussian03 is also supported)
- MOPAC - normal modes
- PCMODEL - normal modes
- ADF - normal modes
- free format normal modes (one number per line, modes separated by wavenumbers)

The modes read from file are stored in a buffer, from which they can be selected as basis vectors for the AKIRA calculation (see section 23.3.6).

23.3.5 Subsystem menu

One of the possible applications of mode-tracking is to study vibrations of a large molecule which are local in that sense that they are more or less restricted to a certain subunit of the system. There may still be couplings with motions of the rest of the molecule, but as a first approximation, only the atoms of the subunit may be involved. In those cases, it might actually be possible to find a small model for the full (super-)molecule, which only consists of the subunit (plus a model for the rest of the molecule), and for which a full frequency analysis is possible. Then these subsystem normal modes can be used as a guess for the full molecule. Indeed, it might be possible to identify several subunits. The subsystem menu allows to define a number of subunits, and to read normal coordinates and wavenumbers for them, which then are used as basis vectors for the full molecule.

Additionally, it is possible to combine the Hessians for these subsystems and copy them to the corresponding blocks of the full Hessian, so that a guess for the full molecule is obtained, which can be used for preconditioning purposes. By typing `submenu` in the main basis vector menu, you get to the subsystem menu:

```
=====================================================
 No. of subsystems specified so far:   0
=====================================================

 Select one of the following commands:

  ns <i> <j>   : create new subsystem, ranging from
                 atom <i> to <j>
  rnm <i>      : read normal modes for subsystem <i>
  chs <i>      : create Hessian for subsystem <i>
  trarot <i>   : create trarot-modes for subsystem <i>
  n2fs <i>     : copy modes for subsystem <i> to full system
  showpr       : show print options
  c2fh <i>     : copy Hessian for subsystem <i> to full Hessian
  ints <i j r> : set interaction strength between blocks <i> and <j> to <r>
  inta <r>     : set interaction strength for all blocks to <r>
  whs          : write (appr.) hessian to akira_control
                 for preconditioning purposes
  cnm          : create normal modes by diagonalization
                 of current full system (appr.) Hessian
  cl           : clear all subsystem entries
  *,q          : return to basis vector menu
```

In that menu, the following commands are possible:

`ns <i j>` define a new subsystem, ranging from atom `i` to atom `j`.

`rnm <i>` read normal modes from file for subsystem no. `i`. For possible input formats, see Sect. 23.3.4.

`chs <i>` create Hessian for subsystem no. `i` from normal modes and wavenumbers. Only possible if normal modes and wavenumbers have already been read.

`trarot <i>` create translational and rotational modes for subsystem no. `i`. This option is useful when it can be assumed that modes of a supermolecule can be described by out-of-phase translations or rotations of different molecules within the supermolecule. These modes can be constructed automatically, no preceeding subsystem calculation is necessary.

`n2fs <i>` copy normal modes for subsystem `i` to set of normal modes for full system.

`showpr` show print options (for normal modes, wavenumbers, and Hessians)

`c2fh <i>` copy Hessian for subsystem `i` to full Hessian. Only possible if subsystem Hessian has been constructed before.

`ints <i j r>` artificially set the interaction strength, i.e., the off-diagonal elements between blocks for subsystems `i` and `j` to the constant value `r`. These blocks are otherwise always zero in the model Hessian for the full molecule, so that no coupling exists between different subunits.

`inta <r>` artificially set the interaction strength for all off-diagonal blocks (blocks between subsystems) to the constant value `r`. Like `ints`, this command allows to empirically correct the model Hessian.

`whs` write the model Hessian for the full molecule to file `akira_control` in order to be able to use it for preconditioning purposes.

`cnm` create model normal coordinates by diagonalization of the model Hessian for the full system.

`cl` clear all subsystem entries

To keep an better overview over the subsystems, a table is shown with the relevant atom numbers, the number of modes read, as well as information about whether the normal modes have been copied to the full system modes, whether the Hessian is created, and whether the Hessian is copied to the full system. After leaving this menu, the modes which are either copied from a subsystem to the full system, or which are created by diagonalization of the model system hessian, will be stored in a buffer. From there, they may be selected as basis vectors.

23.3.6 Other options

Further options which are available in the basis vector menu, are the following:

`trarot` create translational and rotational modes. For test purposes, they might be included in the mode-tracking calculation.

`cbv <i>` copy vector `i` from buffer to set of selected basis vectors. Only available if there are modes in buffer.

`acbv` copy all vectors in buffer to set of selected basis vectors. Only available if there are modes in buffer.

`ort` orthogonalize basis vectors to translational and rotational modes. Only available if basis vectors have been selected. Orthogonalization is carried out automatically afterwards anyway, unless suppressed by the user.

The following options are available for the set of selected basis vectors (without preceeding "`b`") or for the set of buffer vectors (with preceeding "`b`"):

`[b]disp` display modes using MOLDEN [19]

`[b]c` clear set of modes

`[b]p` print set of modes

`[b]pm` print set of mass-weighted modes

`[b]g98` write G98 fake output of modes to file `g98.out.it0` or `g98.out.buf`. Is done automatically when using command `[b]disp`.

`[b]xmol` write XMOL fake output of modes to file `xmol.XYZ.it0` or `xmol.XYZ.buf`.

23.3.7 Restricting the Subspace

Under certain circumstances it may be helpful to restrict the search space for the subspace iteration. Typical examples could be the restriction to basis vectors within a certain irreducible representation or the exclusion of exact normal modes found in a previous run. One restriction that is by default always employed is that the basis vectors are kept orthogonal to translational and rotational motions.

Other restrictions can be applied by specifying a set of orthogonal vectors in the guess vector menu. Any vector that is available in the buffer (see above) can be copied into the set of orthogonal vectors with the commands:

`cov <i>` copy vector `i` from buffer to set of orthogonal vectors. Only available if there are modes in buffer.

`acov` copy all vectors in buffer to set of orthogonal vectors. Only available if there are modes in buffer.

If some vectors have been chosen for the orthogonal complement, their number will be displayed in the menu. Since it is sometimes easier to specify the space in which the possible solutions should lie (instead of its orthogonal complement), there is an additional menu appearing after quitting the basis vector selection that allows the user to either keep the solution vectors within the set of vectors specified in the "orthogonal subspace" selection, or to keep them orthogonal to that set,

```
You have selected an additional set of vectors
to restrict the search space for the soluation
vectors. Do you want to ...
 sub - keep the solution vectors within this set
       (default)
 ort - keep the solution orthogonal to this set
 q   - quit without using this set of vectors
```

If `sub` is chosen, AKIRADEFINE constructs the complementary space to the one selected before and keeps the solution vectors orthogonal to that set. Note that the orthogonal subset vectors are also orthogonalized to translational and rotational modes, as all basis vectors are per default kept orthogonal to these rigid modes. The orthogonal modes are written to the file `akira_control`.

23.4 Subspace iteration with AKIRA

The general procedure for a subspace iteration is outlined in the flow-chart in Fig. 23.1. To request a subspace iteration with AKIRA simply call
`akira`
without any further program options. These must be specified in the AKIRA control file `akira_control` and can be selected interactively in AKIRADEFINE.

23.5 Output files

- Akira writes an output file named `akira.out`, which contains intermediate data for every iteration. In particular, the following quantities are reported:
  - vector $\boldsymbol{\sigma}^i$, Eq. (22.3)
  - approximate Hessian matrix for subspace (Davidson matrix), $\tilde{\mathbf{H}}^{(m),i}$, Eq. (22.5)
  - approximate eigenvalues and eigenvectors of the Davidson matrix, ρ^i and $\mathbf{u}^i$, Eq. (22.6)

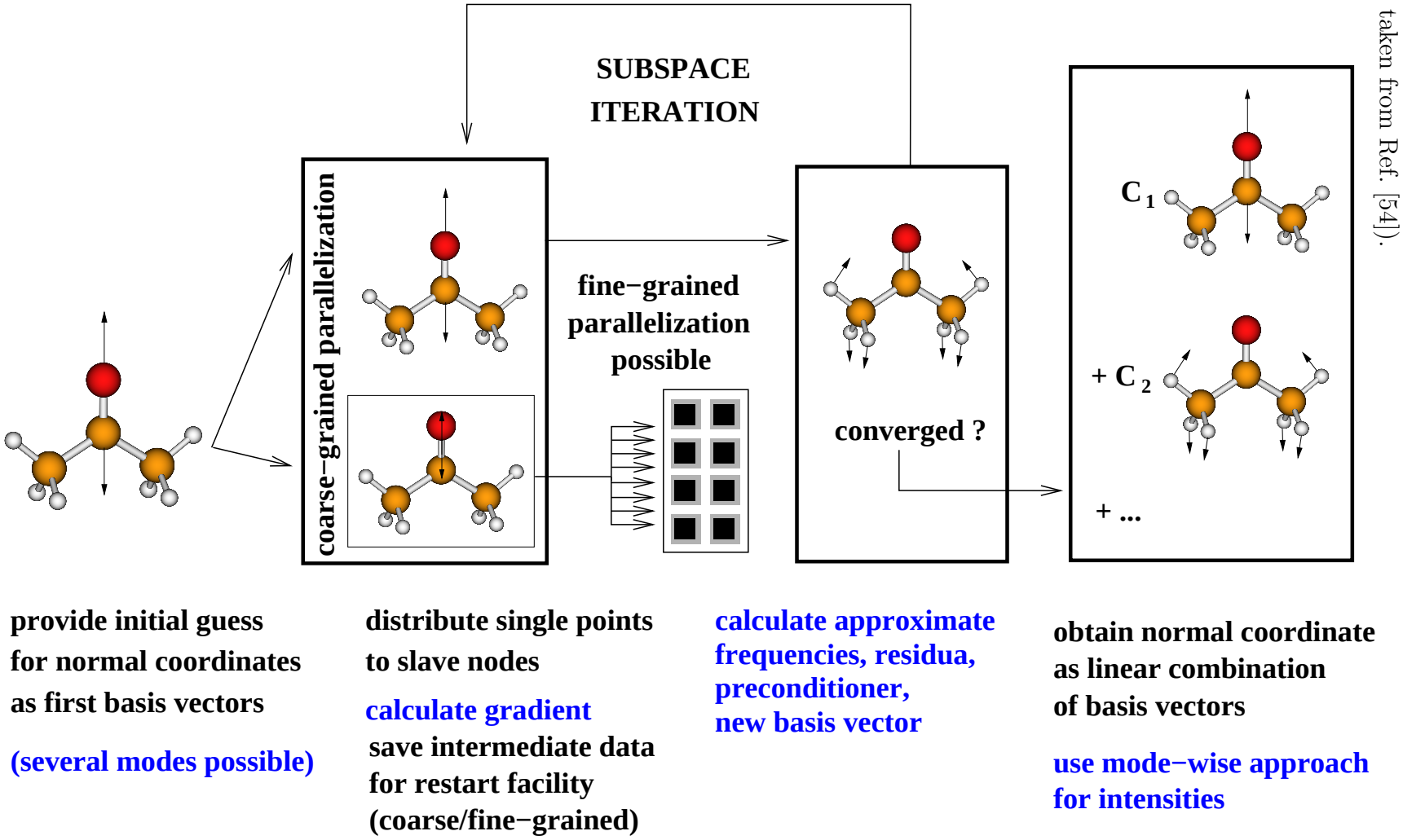


Figure 23.1: Outline of the parallelized mode-tracking algorithm as implemented in AKIRA (figure taken from Ref. [54]).

- approximate wavenumbers calculated from the eigenvalues
- approximate normal modes, $\mathbf{v}_s^i$, Eq. (22.8)
- information about the root-homing procedure; three quantities are calculated in order to select the eigenvector to be optimized from the set of the $\mathbf{v}_s^i$:
  * the unsigned difference between the eigenvalues of current iteration and the eigenvalue(s) of the eigenvector(s) selected in the last iteration
  * the squared norm of the difference vector between the eigenvectors in the current iteration and the eigenvector(s) selected in the previous iteration.
  * the unsigned overlap of the current eigenvectors and the eigenvector(s) selected in the previous iteration.

  Note that the root-homing is necessary for all eigenvectors which shall be optimized simultaneously. If the three criterions lead to different eigenvectors, the overlap criterion will be used to determine the new eigenvector(s) to be optimized.
- residuum vectors $\mathbf{r}_s^i$ for all approximate eigenvectors $\mathbf{v}_s^i$, Eq. (22.7)
- convergence control parameters:
  * norm of residuum vector(s) for selected eigenvector(s)
  * change of residuum vector norm
  * maximum component of the residuum vector
  * contribution of the last basis vector to the new approximate normal mode vector
- new Cartesian basis vector(s) $\mathbf{b}^{i+1}$, Eq. (22.9)
- intermediate results: wavenumbers, $|r|$, $|r_{\max}|$ and convergence status for *all* approximate normal modes in the current iteration

- `akira_iterations.out`: Contains information on the input settings, the molecule, and the results of the subspace iterations such as basis vector coefficients, convergence control, root homing, intermediate and final vibrational normal modes and wave numbers. When choosing `$printall on` in the `akira_control` file, more detailed output will be written to `akira_iterations.out`.

- `g98.out.itn`: Vibrational normal modes and wave numbers in iteration `n` in GAUSSIAN98 format.

- `g98.out.bsn`: Basis vectors in iteration `n` in GAUSSIAN98 format. Wave numbers are meaningless here and are therefore set to 10000.0 for all basis vectors.

- Furthermore, the file `wavenumbers` is updated after each iteration, which contains the wavenumber of the residuum vector for each approximate normal mode.

- Also, the information in `restart.akira` and the basis modes in `akira_control` will be updated during the AKIRA run.

23.6 Restart facilities

The single-point data obtained for structures perturbed along the basis vectors are stored in the file `restart.akira`. Besides the gradients for all displaced structures, which enter Eq. (22.3), the energies, the equilibrium structure and all perturbed structures are written to this file. The general structure of the restart file is essentially the same as for the program package SNF (see introduction), with the difference that gradients and energies are collected in groups for every basis vector, not in groups for Cartesian displacements of a particular atom (as in SNF) or in groups for the normal modes of the molecule. In earlier AKIRA versions, the number of basis-vector entries in the restart file was equal to the number of degrees of freedom of the molecule for convenience. In version 3.1.2, this was changed into the number of basis modes present up to the current iteration. This was necessary for the QM/MM interface of AKIRA, which allows to treat systems with a huge number of degrees of freedom, in order to keep the size of the restart file small.

The stepflag entries in the restart file determine which calculations have to be done in a particular iteration step. Further restart information is written in the program control file `akira_control` in the following form:

```
$nnbm   3
   1
   1
   1
```

This means that the current iteration is the third, and the three "`1`" entries specify the number of basis vectors added in each iteration, i.e., there are now three basis vectors, one of which was generated per iteration step. This information allows — in combination with the stepflags — to restart the calculation at every point of the subspace iteration, irrespectively of the current process (single point calculations for perturbed structures or solution of the Davidson matrix eigenvalue problem/generation of new basis vectors).

It is sometimes necessary or convenient to restart the calculation at an earlier iteration. This is possible by introducing the additional keyword

```
$restartit n
```

in the file `akira_control`, where `n` is the iteration number after which the calculation should be restarted, i.e., restart information will only be used up to iteration `n`. **CAUTION: This will remove all previous raw data from iteration `n+1` on your restart file !!!**

Summary AKIRADEFINE**:**

⇒ If you have performed a geometry optimization with TURBOMOLE, just type `akiradefine` in your working directory.

⇒ If you have used GAUSSIAN for the optimization, create a `akira.com` file (see Sect. 23.2.1) and then type `akiradefine` in your working directory.

⇒ For ADF users: assure that you have a runscript named `adf.in` in your working directory which contains all settings used in the preceding geometry optimization.

⇒ Select the options in the menus (see Sect. 23.2 for details).

⇒ Note possible warnings in the output of AKIRADEFINE.

⇒ For TURBOMOLE users: If the symmetry mentioned in the original `control` file is higher than C_1, create a C_1 MO file by using TURBOMOLE'S DEFINE.

- Mandatory files for AKIRADEFINE:
  `control, coord` (TURBOMOLE, DALTON) or `akira.com` (GAUSSIAN) or `adf.in` (ADF).

- Mandatory files and scripts for automatical generation of AKIRA input files by AKIRADEFINE:
  `$CIPROC/ELIGIBLE, choose_nodes, mkrdcinput` or `mkparainput`

- Files created by AKIRADEFINE:
  `restart, akira_control, control.bak, DCINPUT, USE_NODES, MACHINES, mpi.sub`

Summary AKIRA**:**

⇒ Create the files `USE_NODES` using the script `choose_nodes` (this is also done automatically by AKIRADEFINE).

⇒ Start the AKIRA calculation by typing `akira` in your working directory, or, in order to prevent the AKIRA from crashing when you log out, to send the command to the background and to pipe the screen output to a file:
`nohup akira > filename &`

⇒ You may check the progress of the calculation via the status file `TMPdcstat`, which can be constantly viewed by the command `dcmore`.

- Mandatory files for AKIRA:
  `control, coord, restart, DCINPUT, akira_control`;
  TURBOMOLE, DALTON: `basis`[and `auxbasis`], `mos` [or `alpha, beta`];
  GAUSSIAN: `akira.com`
  ADF: `adf.in`
  MPI versions: `MACHINES` (for `mpirun`)

- Mandatory scripts for AKIRA:
  `choose_nodes, ruptime`

- Files created by AKIRA in the working directory:
  `TMPdcstat, fort.41, g98.out.it`n, `g98.out.bs`n

- Files created by AKIRA in the log directory:
  `PREFIXdclog.XXXXXXX`

⇒ Note possible warnings in the output of AKIRA.

23.7 Double-parallel runs

The maximum number of processors in the parallel machine is restricted by the number of tracked vibrations (2 per vibration, if a 3-point central-difference formula is used for numerical differentiation). Thus, it may be desirable to run also the single-point calculations in parallel to exploit the full capacity of a computer cluster. At present, this has only been tested for the Akira—Adf interface on a PC cluster built from machines with 2 dual-core AMD Opteron processors (⇒ 4 processors) each, using the MPI version of Akira. The Adf calculations were parallized employing PVM, so any interference of the two parallel processes was avoided. To control the parallel execution of the Adf jobs, the following lines were inserted at the very beginning of the Adf input file `adf.in`:

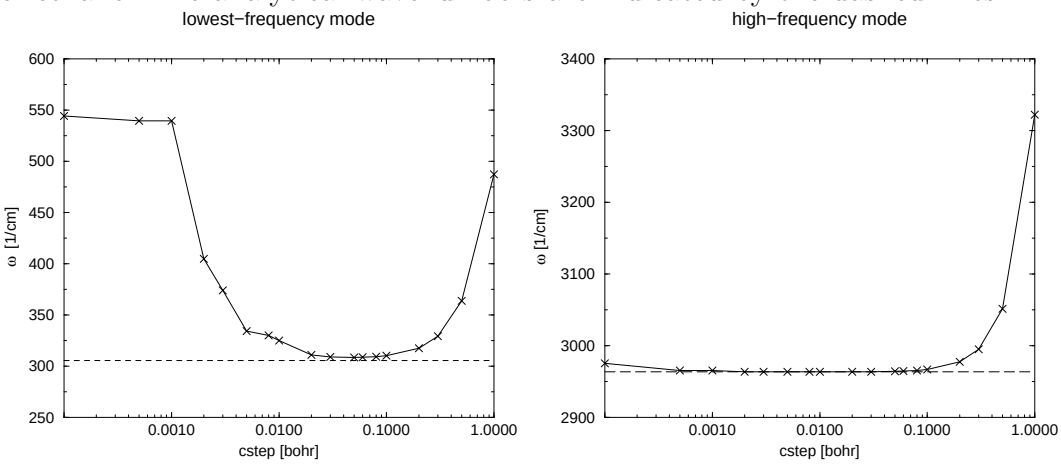
```
/usr/bin/pvm << eor
quit
eor

echo `uname -n` > adfhosts
echo `uname -n` 4 > nodeinfo

export SCMSPAWNSCRIPT=$ADFBIN/adfs
export NSCM=4
```

Furthermore, it should be checked that the main Adf run is not started as `$ADFBIN/adf -n1 << eor`, but as `$ADFBIN/adf << eor` by `adf.in`.

24. Parameter analysis: step size for numerical differentiation

To determine an optimum step size for the numerical differentiation, we carried out displacements of the atoms in the ethane molecule along the lowest-frequency normal mode, and along one high-frequency mode, and calculated the force constants as numerical derivatives of the analytical gradients (BP86/TZVP). The resulting frequencies as a function of the step size are displayed in Fig. 24.1. Note that these issues have been discussed in detail in Ref. [22].

Figure 24.1: Vibrational wavenumbers in cm^{-1} as a function of the step size for the numerical differentiation for the lowest-frequency (left) and a high-frequency mode (right) of ethane. The analytical wavenumbers are indicated by the dashed lines.



From this figure it can be seen that it might be advantageous — though the general recommendation is 0.01 — to increase the step size `cstep` to values of up to 0.1 for *low-frequency* modes.

25. Program history of Akira

Modifications after version 4.0.0 are listed below according to the version numbers.

4.0.1 Serial version of Akira now available. This version is integrated into MoViPac 1.0.1.

4.0.0 Completely revised version for integration into MoViPac 1.0.0.

Bibliography

[1] Thomas Weymuth, Moritz P. Haag, Karin Kiewisch, Sandra Luber, Stephan Schenk, Christoph R. Jacob, Carmen Herrmann, Johannes Neugebauer, Markus Reiher. MoViPac: Vibrational Spectroscopy with a Massively Parallelized, Robust and Inverse Meta-Program. *J. Comput. Chem.*, **33** (2012) 2186–2198.

[2] S. Luber, M. Reiher. Theoretical Raman Optical Activity Study of the β Domain of Rat Metallothionein. *J. Phys. Chem. B*, **114** (2010) 1057–1063.

[3] J. Neugebauer, M. Reiher, C. Kind, B. A. Hess. Quantum chemical calculation of vibrational spectra of large molecules—Raman and IR spectra for Buckminsterfullerene. *J. Comput. Chem.*, **23**(9) (2002) 895–910.

[4] G. Herzberg. *Molecular Spectra and Molecular Structure II: Infrared and Raman Spectra of Polyatomic Molecules.* Van Nostrand Reinhold, New York, 1945.

[5] E. Bright Wilson, Jr., J. C. Decius, P. C. Cross. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra.* McGraw-Hill, New York, 1955.

[6] D. A. Long. *The Raman Effect: A Unified Treatment of the Theory of Raman Scattering by Molecules.* John Wiley & Sons, New York, 2002.

[7] L. D. Barron. *Molecular Light Scattering and Raman Optical Activity.* Cambridge University Press, Cambridge, 2nd ed., 2004.

[8] P. R. Bunker, P. Jensen. *Molecular Symmetry and Spectroscopy.* NRC Research Press, Ottawa, Canada, 2nd ed., 1998.

[9] W. Miller, Jr. *Symmetry Groups and their Applications.* Academic Press, New York, 1972.

[10] D. S. Schonland. *Molecular Symmetry: An Introduction to Group Theory and its Uses in Chemistry.* Van Nostrand Reinhold Company, London, 1971.

[11] R. Ahlrichs, M. Bär, M. Häser, H. Horn, C. Kölmel. Electronic structure calculations on workstation computers: The program system Turbomole. *Chem. Phys. Lett.*, **162**(3) (1989) 165–169.

[12] C. Angeli, K. L. Bak, V. Bakken, O. Christiansen, R. Cimiraglia, S. Coriani, P. Dahle, E. K. Dalskov, T. Enevoldsen, B. Fernandez, C. Hättig, K. Hald, A. Halkier, H. Heiberg, T. Helgaker, H. Hettema, H. J. Aa. Jensen, D. Jonsson, P. Jørgensen, S. Kirpekar, W. Klopper, R. Kobayashi, H. Koch, A. Ligabue, O. B. Lutnæs, K. V. Mikkelsen, P. Norman, J. Olsen, M. J. Packer, T. B. Pedersen, Z. Rinkevicius, E. Rudberg, T. A. Ruden, K. Ruud, P. Sałek, A. Sanchez de Meras, T. Saue, S. P. A. Sauer, B. Schimmelpfennig, K. O. Sylvester-Hvid, P. R. Taylor, O. Vahtras, D. J. Wilson, H. Ågren. DALTON Release 2.0, 2005. http://www.kjemi.uio.no/software/dalton/dalton.html.

[13] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, J. A. Pople. GAUSSIAN03 revision D.01, 2004. Gaussian, Inc., Wallingford, CT. http://www.gaussian.com.

[14] G. te Velde, F. M. Bickelhaupt, E. J. Baerends, C. Fonseca Guerra, S. J. A. van Gisbergen, J. G. Snijders, T. Ziegler. Chemistry with ADF. *J. Comput. Chem.*, **22**(9) (2001) 931–967.

[15] H.-J. Werner, P. J. Knowles, R. Lindh, M. Schütz, P. Celani, T. Korona, F. R. Manby, G. Rauhut, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, C. Hampel, G. Hetzer, A. W. Lloyd, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, P. Palmieri, R. Pitzer, U. Schumann, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson. MOLPRO version 2002.6, 2003. Birmingham, U.K. http://www.molpro.net.

[16] T. Williams, C. Kelley. Gnuplot. http://www.gnuplot.info.

[17] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas,

J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, , P. Salvador, J. J. Dannenberg, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, A. G. Baboul, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, J. A. Pople. Gaussian98 revision A.11.4, 2002. Gaussian, Inc., Pittsburgh, PA. http://www.gaussian.com.

[18] J. J. P. Stewart, I. Rossi, W.-P. Hu, G. C. Lynch, Y.-P. Liu, Y.-Y. Chuang, J. Li, C. J. Cramer, P. L. Fast, D. G. Truhlar. Mopac 5.09mn, 1999. University of Minnesota, Minneapolis.

[19] G. Schaftenaar, J. H. Noordik. Molden: a pre- and post-processing program for molecular and electronic structures. *J. Comput. Aided Mol. Des.*, **14**(2) (2000) 123–134.

[20] Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org.

[21] M. Reiher, J. Neugebauer. A mode-selective quantum chemical method for tracking molecular vibrations applied to functionalized carbon nanotubes. *J. Chem. Phys.*, **118**(4) (2003) 1634–1641.

[22] M. Reiher, J. Neugebauer. Convergence characteristics and efficiency of mode-tracking calculations on pre-selected molecular vibrations. *Phys. Chem. Chem. Phys.*, **6**(19) (2004) 4621–4629.

[23] Ernest R. Davidson. The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices. *J. Comp. Phys.*, **17** (1975) 87–94.

[24] H. A. Van der Vorst G. L. G. Sleijpen. A Jacobi–Davidson Iteration Method for Linear Eigenvalue Problems. *SIAM Review*, **42** (2000) 267–293.

[25] C. Lanczos. *J. Res. Nat. Bur. Stand.*, **45** (1950) 255.

[26] C. Herrmann, J. Neugebauer, M. Reiher. QM/MM vibrational mode tracking. *J. Comput. Chem.*, **29**(14) (2008) 2460–2470.

[27] Christoph R. Jacob, Markus Reiher. Localizing normal modes in large molecules. *J. Chem. Phys.*, **130** (2009) 084106.

[28] N. S. Bieler, M. P. Haag, C. R. Jacob, M. Reiher. Analysis of the Cartesian Tensor Transfer Method for Calculating Vibrational Spectra of Polypeptides. *J. Chem. Theory Comput.*, **7** (2011) 1867–1881.

[29] Sandra Luber, Johannes Neugebauer, Markus Reiher. Intensity tracking for theoretical infrared spectroscopy of large molecules. *J. Chem. Phys.*, **130**(6) (2009) 064105.

[30] Sandra Luber, Markus Reiher. Intensity-carrying modes in raman and raman optical activity spectroscopy. *ChemPhysChem*, **10** (2009) 2049–2057.

[31] K. Kiewisch, J. Neugebauer, M. Reiher. Selective calculation of high-intensity vibrations in molecular resonance Raman spectra. *J. Chem. Phys.*, **129**(20) (2008) 204103.

[32] C. Herrmann, M. Reiher. First-principles approach to vibrational spectroscopy of biomolecules. *Top. Curr. Chem.*, **268** (2007) 85–132.

[33] Karin Kiewisch, Sandra Luber, Johannes Neugebauer, Markus Reiher. Intensity tracking for vibrational spectra of large molecules. *CHIMIA*, **63** (2009) 270–274.

[34] C. Herrmann, J. Neugebauer, M. Reiher. Finding a needle in a haystack: Direct determination of vibrational signatures in complex systems. *New J. Chem.*, **31**(6) (2007) 818–831.

[35] G. Brehm, M. Reiher, S. Schneider. Estimation of the vibrational contribution to the entropy change associated with the low- to high-spin transition in $Fe(phen)_2(NCS)_2$ complexes: Results obtained by IR and Raman spectroscopy and DFT calculations. *J. Phys. Chem. A*, **106**(50) (2002) 12024–12034.

[36] S. Schneider, M. O. Schmitt, G. Brehm, M. Reiher, P. Matousek, M. Towrie. Fluorescence kinetics of aqueous solutions of tetracycline and its complexes with Mg^{2+} and Ca^{2+}. *Photochem. Photobiol. Sci.*, **2**(11) (2003) 1107–1117.

[37] C. F. Leypold, M. Reiher, G. Brehm, M. O. Schmitt, S. Schneider, P. Matousek, M. Towrie. Tetracycline and derivatives—assignment of IR and Raman spectra via DFT calculations. *Phys. Chem. Chem. Phys.*, **5**(6) (2003) 1149–1157.

[38] T. B. Adler, N. Borho, M. Reiher, M. A. Suhm. Chirality-induced switch in hydrogen-bond topology: Tetrameric methyl lactate clusters in the gas phase. *Angew. Chem., Int. Ed.*, **45**(21) (2006) 3440–3445.

[39] G. Brehm, M. Reiher, B. Le Guennic, M. Leibold, S. Schindler, F. W. Heinemann, S. Schneider. Investigation of the low-spin to high-spin transition in a novel $[Fe(pmea)(NCS)_2]$ complex by IR and Raman spectroscopy and DFT calculations. *J. Raman Spectrosc.*, **37**(1–3) (2006) 108–122.

[40] C. Herrmann, K. Ruud, M. Reiher. Importance of backbone angles versus amino acid configurations in peptide vibrational Raman optical activity spectra. *Chem. Phys.*, **343**(2–3) (2008) 200–209.

[41] C. R. Jacob, S. Luber, M. Reiher. Calculated Raman optical activity signatures of tryptophan side chains. *ChemPhysChem*, **9**(15) (2008) 2177–2180.

[42] S. Luber, M. Reiher. Raman optical activity spectra of chiral transition metal complexes. *Chem. Phys.*, **346**(1–3) (2008) 212–223.

[43] Sandra Luber, Markus Reiher. Prediction of raman optical activity spectra of chiral 3-acetylcamphorato-cobalt complexes. *ChemPhysChem*, **11** (2010) 1876–1887.

[44] Christoph R. Jacob, Sandra Luber, Markus Reiher. Understanding the Signatures of Secondary-Structure Elements in Proteins with Raman Optical Activity Spectroscopy. *Chem. Eur. J.*, **15** (2009) 13491–13508.

[45] Thomas Weymuth, Christoph R. Jacob, Markus Reiher. Identifying Protein β-Turns with Vibrational Raman Optical Activity. *ChemPhysChem*, **12** (2011) 1165–1175.

[46] Sandra Luber, Markus Reiher. Calculated raman optical activity spectra of 1,6-anhydro--d-glucopyranose. *J. Phys. Chem. A*, **113** (2009) 8268–8277.

[47] J. Neugebauer, M. Reiher, B. A. Hess. Coupled-cluster Raman intensities: Assessment and comparison with multiconfiguration and density functional methods. *J. Chem. Phys.*, **117**(19) (2002) 8623–8633.

[48] M. Reiher, V. Liégeois, K. Ruud. Basis set and density functional dependence of vibrational Raman optical activity calculations. *J. Phys. Chem. A*, **109**(33) (2005) 7567–7574.

[49] J. Neugebauer, B. A. Hess. Fundamental vibrational frequencies of small polyatomic molecules from density-functional calculations and vibrational perturbation theory. *J. Chem. Phys.*, **118**(16) (2003) 7215–7225.

[50] M. Reiher, G. Brehm, S. Schneider. Assignment of vibrational spectra of 1,10-phenanthroline by comparison with frequencies and Raman intensities from density functional calculations. *J. Phys. Chem. A*, **108**(5) (2004) 734–742.

[51] M. Reiher, J. Neugebauer, B. A. Hess. Quantum chemical calculation of Raman intensities for large molecules: The photoisomerization of $[\{Fe`S_4`(PR_3)\}_2(N_2H_2)]$ (`S4'$^{2-}$ = 1,2-bis(2-mercaptophenylthio)-ethane(2−)). *Z. Phys. Chem.*, **217**(2) (2003) 91–103.

[52] J. Neugebauer, B. A. Hess. Resonance Raman spectra of uracil based on Kramers–Kronig relations using time-dependent density functional calculations and multireference perturbation theory. *J. Chem. Phys.*, **120**(24) (2004) 11564–11577.

[53] J. Neugebauer, M. Reiher. Mode tracking of preselected vibrations of one-dimensional molecular wires. *J. Phys. Chem. A*, **108**(11) (2004) 2053–2061.

[54] J. Neugebauer, M. Reiher. Vibrational center–ligand couplings in transition metal complexes. *J. Comput. Chem.*, **25**(4) (2004) 587–597.

[55] M. Reiher, J. Neugebauer. Comment on "Gradient-based direct normal-mode analysis" [J. Chem. Phys. 122, 184106 (2005)]. *J. Chem. Phys.*, **123**(11) (2005) 117101.

[56] C. Herrmann, K. Ruud, M. Reiher. Can Raman optical activity separate axial from local chirality? A theoretical study of helical deca-alanine. *ChemPhysChem*, **7**(10) (2006) 2189–2196.

[57] C. Herrmann, M. Reiher. Direct targeting of adsorbate vibrations with mode-tracking. *Surf. Sci.*, **600**(9) (2006) 1891–1900.

[58] C. Herrmann, J. Neugebauer, M. Presselt, U. Uhlemann, M. Schmitt, S. Rau, J. Popp, M. Reiher. The First photoexcitation step of ruthenium-based models for artificial photosynthesis highlighted by resonance Raman spectroscopy. *J. Phys. Chem. B*, **111**(21) (2007) 6078–6087.

[59] S. Luber, C. Herrmann, M. Reiher. Relevance of the electric-dipole–electric-quadrupole contribution to Raman optical activity spectra. *J. Phys. Chem. B*, **112**(7) (2008) 2218–2232.

[60] Christoph R. Jacob, Sandra Luber, Markus Reiher. Analysis of Secondary Structure Effects on the IR and Raman Spectra of Polypeptides in Terms of Localized Vibrations. *J. Phys. Chem. B*, **113** (2009) 6558–6573.

[61] Thomas Weymuth, Christoph R. Jacob, Markus Reiher. A Local-Mode Model for Understanding the Dependence of the Extended Amide III Vibrations on Protein Secondary Structure. *J. Phys. Chem. B*, **114** (2010) 10649–10660.

[62] Sandra Luber, Johannes Neugebauer, Markus Reiher. Enhancement and de-enhancement effects in vibrational resonance raman optical activity. *J. Chem. Phys.*, **132**(4) (2010) 044113.

[63] XMol version 1.3.1, 1993. Research Equipment Inc., Minnesota Supercomputer Center, Inc.

[64] J. Cioslowski. A new population analysis based on atomic polar tensors. *J. Am. Chem. Soc.*, **111**(22) (1989) 8333–8336.

[65] Harald Solheim, Kenneth Ruud, Per-Olof Åstrand. *J. Chem. Phys.*, **120** (2004) 10368–10378.

[66] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philos. Trans. R. Soc. A*, **210**(459–470) (1911) 307–357.

[67] G. Rauhut, P. Pulay. Transferable scaling factors for density functional derived vibrational force fields. *J. Phys. Chem.*, **99**(10) (1995) 3093–3100.

[68] L. D. Barron, L. Hecht, I. H. McColl, E. W. Blanch. Raman optical activity comes of age. *Mol. Phys.*, **102**(8) (2004) 731–744.

[69] P. W. Atkins, L. D. Barron. Rayleigh scattering of polarized photons by molecules. *Mol. Phys.*, **16**(5) (1969) 453–466.

[70] A. D. Buckingham, M. B. Dunn. Optical activity of oriented molecules. *J. Chem. Soc. A*, (1971) 1988–1991.

[71] L. D. Barron, A. D. Buckingham. Rayleigh and Raman scattering from optically active molecules. *Mol. Phys.*, **20**(6) (1971) 1111–1119.

[72] L. D. Barron, M. P. Bogaard, A. D. Buckingham. Raman scattering of circularly polarized light by optically active molecules. *J. Am. Chem. Soc.*, **95**(2) (1973) 603–605.

[73] W. Hug, S. Kint, G. F. Bailey, J. R. Scherer. Raman circular intensity differential spectroscopy. Spectra of $(-)$-α-pinene and $(+)$-α-phenylethylamine. *J. Am. Chem. Soc.*, **97**(19) (1975) 5589–5590.

[74] Ronald F. Boisvert, Roldan Pozo, Karin A. Remington. The matrix market exchange formats: Initial design. *NISTIR*, **5935** (1996.

[75] Donald A. McQuarrie, John D. Simon. *Molecular Thermodynamics.* University Science Books, Sausalito, 1999.

[76] A. Fernández-Ramos, B. A. Ellingson, R. Meana-Pa neda, J. M. C. Marques, D. G. Truhlar. Symmetry numbers and chemical reaction rates. *Theor. Chem. Acc.*, **118**(4) (2007) 813–826.

[77] P. Bouř, J. Sopková, L. Bednárová, P. Maloň, T. A. Keiderling. Transfer of Molecular Property Tensors in Cartesian Coordinates: A New Algorithm for Simulation of Vibrational Spectra. *J. Comput. Chem.*, **18** (1997) 646–659.

[78] K. Eichkorn, O. Treutler, H. Öhm, M. Häser, R. Ahlrichs. Auxiliary basis sets to approximate Coulomb potentials. *Chem. Phys. Lett.*, **240**(4) (1995) 283–290.

[79] K-E. J. Hallin, J. W. C. Johns, A. Trombetti. The infrared spectrum of di-imide near 7.6 μm. *Can. J. Phys.*, **59**(5) (1981) 663–672.

[80] F. Hegelund, H. Bürger, O. Polanz. The High-Resolution Infrared Spectrum of the ν_4, ν_5, and ν_6 Bands of trans-Di-imide Revisited. *J. Mol. Spectrosc.*, **167**(1) (1994) 1–10.

[81] V. E. Bondybey, J. W. Nibler. Infrared and Raman spectra of solid and matrix isolated diimide, HNNH. *J. Chem. Phys.*, **58**(5) (1973) 2125–2134.

[82] B. A. Hess. NUMFREQ. University of Erlangen-Nürnberg, 2001. [Based on work by S. Grimme, C. Marian and M. Gastreich, University of Bonn, 1998].

[83] Bernd A. Hess. Memmgr (Version 1.19), 1999.

[84] Bernd A. Hess. Delrem (Revision 1.66), 2001.

[85] T. Helgaker, H. J. Aa. Jensen, P. Jørgensen, J. Olsen, K. Ruud, H. Ågren, A. A. Auer, K. L. Bak, V. Bakken, O. Christiansen, S. Coriani, P. Dahle, E. K. Dalskov, T. Enevoldsen, B. Fernandez, C. Hättig, K. Hald, A. Halkier, H. Heiberg, H. Hettema, D. Jonsson, S. Kirpekar, R. Kobayashi, H. Koch, K. V. Mikkelsen, P. Norman, M. J. Packer, T. B. Pedersen, T. A. Ruden, A. Sanchez, T. Saue, S. P. A. Sauer, B. Schimmelpfennig, K. O. Sylvester-Hvid, P. R. Taylor, O. Vahtras. DALTON, Release 1.2, 2001.

[86] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, J. A. Pople. GAUSSIAN03, revision D.02. Gaussian, Inc., Wallingford, CT. http://www.gaussian.com.

[87] Ira N. Levine. *Molecular Spectroscopy.* Wiley, New York, 1975.

[88] W. G. Bickley. Formulae for numerical differentiation. *Math. Gaz.*, **25** (1941) 19–27.

[89] B. Liu. *Numerical Algorithms in Chemistry: Algebraic Methods* (edited by C. Moler and I. Shavitt). Lawrence Berkeley Laboratory LBL-8158. Livermore, CA, 1978.

[90] Christopher W. Murray, Stephen C. Racine, Ernest R. Davidson. Improved Algorithms for the Lowest Few Eigenvalues and Associated Eigenvectors of Large Matrices. *J. Comp. Phys.*, **103** (1992) 382–389.

[91] W. Butscher, W. J. E. Kammer. *J. Comput. Phys.*, **20** (1976) 13.