CMIstark Documentation

Release 1.3

CFEL Controlled Molecule Imaging

30. May 2020

Contents

1 User documentation **3**
- 1.1 Installing CMIstark . . . 3
  - 1.1.1 Prerequisites and obtaining CMIstark . . . 3
  - 1.1.2 Installing CMIstark . . . 3
  - 1.1.3 Installing CMIstark: in user-specified path . . . 4
- 1.2 CMIstark user guide . . . 4
  - 1.2.1 General usage . . . 5
  - 1.2.2 cmistark_calculate_energy . . . 6
  - 1.2.3 cmistark_print_energy . . . 6
  - 1.2.4 cmistark_plot_energy . . . 6
- 1.3 Add a new molecule to CMIstark . . . 7
- 1.4 Extending and embedding CMIstark . . . 7
  - 1.4.1 Manually accessing the CMIstark results . . . 7

2 Developer documentation **9**
- 2.1 cmistark package . . . 9
  - 2.1.1 Submodules . . . 9
    - 2.1.1.1 cmistark.molecule module . . . 9
    - 2.1.1.2 cmistark.moleculeparameters module . . . 9
    - 2.1.1.3 cmistark.starkeffect module . . . 9
  - 2.1.2 Module contents . . . 9
- 2.2 CMIstark developer guide . . . 9
  - 2.2.1 Source code formatting . . . 9
  - 2.2.2 Version control (git) details . . . 10
  - 2.2.3 State labels of stored Stark curves . . . 10
  - 2.2.4 Structure of <moleculename>.molecule . . . 11
  - 2.2.5 Descriptions of source code files . . . 11
- 2.3 ToDo list . . . 12
- 2.4 References . . . 13

3 Indices and tables **15**

Bibliography **17**

`CMIstark` is a software package and tool for DC Stark effect calculations of molecules in electric fields.

See [Chang2014] for a scientific description of version 1.0 and as a formal, citeable reference.

CHAPTER 1

User documentation

1.1 Installing CMIstark

1.1.1 Prerequisites and obtaining CMIstark

Since CMIstark is written in Python, you need to install Python; CMIstark requires Python version 3.5 or higher.

In addition, you need various Python extension packages, these include

- cmiext
- NumPy
- SciPy
- Tables
- Matplotlib

CMIstark is avaiable on GitHub, please contact Jochen Küpper <jochen.kuepper@cfel.de> for further details.

1.1.2 Installing CMIstark

A normal installation is performed by simply running the command:

```
python setup.py install
```

However, often you do not have the administrative rights to install in global directories, or simply do not want to overrride a global installtion. In this case, you might want to perform a local installation in your user directory using:

```
python setup.py install --user
```

A similar setup can be achieved using:

```
python setup.py develop --user
```

which, however, sets up the installation in such a way that changes to your source directory are automatically and immediately visible through the installed version. This avoids repeated re-installs while you are developing code.

Once you are satisfied with your changes you might consider reinstalling using one of the above two options.

Fur further details of `develop` install, see http://naoko.github.io/your-project-install-pip-setup

1.1.3 Installing CMIstark: in user-specified path

Use PYTHONUSERBASE to specify the installation path:

```
setenv PYTHONUSERBASE $HOME/.local
python setup.py install --user
```

In the above example of installation (in tcsh shell), the module will be installed in the following path:

```
$HOME/.local/lib/python/site-packages
```

and the scripts will be installed in the following path:

```
$HOME/.local/bin
```

To import modules and call scripts of such user-specific installation, the following environment declarifications are required:

```
setenv PATH /opt/local/bin:$HOME/.local/bin:$PATH
setenv PYTHONUSERBASE $HOME/.local
```

The above example is provided for the tcsh shell. You can also then use `site` module of python in python command prompt to make sure the environment is properly set up. For example:

```
>>> import site
>>> site.USER_BASE
'$HOME/.local'
```

Also type "which `name of script file`" to find the real path of the script called. It should be in "$HOME/.local/bin".

For further details, see https://docs.python.org/3/install/index.html#inst-alt-install-user and https://docs.python.org/3/using/cmdline.html#envvar-PYTHONUSERBASE

1.2 CMIstark user guide

This program can calculate, view, and analyze the energy levels of adiabatic Stark energy curves of linear, symmetric top and asymmetric top molecules.

The program package is developed and maintained by the Controlled Molecule Imaging group (CMI) at the Center for Free-Electron Laser Science (CFEL), Hamburg, Germany.

It is documented in

- Yuan-Pin Chang, Frank Filsinger, Boris G. Sartakov, Jochen Küpper, *Comp. Phys. Comm.*, **185**, 339 (2014).
- arXiv:1308.4076 [physics] (eprint of the above CPC paper).

See [Chang2014] for the full reference and links.

1.2.1 General usage

The following provides examples of the general usage.

Todo: Yuan-Pin, you misunderstood my earlier todos. You should first write about the general usagem adn then provide a specific example command at the end of the section of each tool. Put together, these shall give a working example. At every stage, you should provide info on the actual output. It will surely be very useful to even provide the output graphics of `plot`.

Calculate the Stark energies of water:

```
cmistark_calculate_energy --isomer=0 --Jmax_calc=10 --Jmax_save=2 --water -
↪-dc-fields=0:150:16
```

After the calculation finishes, it yields a Stark energy file `water.molecule`.

Plot the Stark energy file `water.molecule`:

```
cmistark_plot_energy --Jmax=0 water.molecule
```

A plot of the Stark energy of J=0 state for water will be created.

Print the Stark energy from the file `water.molecule`:

```
cmistark_print_energy --Jmax=0 water.molecule
```

And the print result is:

```
# state: 0 0 0 0 0
0.0 0.0 0.0
10.0 -26.1645127287 5.23264284222
20.0 -104.652856844 10.4642471557
30.0 -235.449455843 15.6937755522
40.0 -418.528367889 20.9201929214
50.0 -653.853314271 26.1424675625
60.0 -941.377719139 31.3595723071
70.0 -1281.04476041 36.57048563
80.0 -1672.78743174 41.7741927455
90.0 -2116.52861532 46.9696866871
100.0 -2612.18116548 52.1559693657
110.0 -3159.64800264 57.332052607
120.0 -3758.82221762 62.496959162
130.0 -4409.58718588 67.6497236918
140.0 -5111.81669146 72.7893937211
150.0 -5865.3750603 72.7893937211
Closing remaining open files:water.molecule...done
```

1.2.2 cmistark_calculate_energy

A (command line) script file called `cmistark_calculate_energy` is provided as a driver for the calculation of Stark curves. Its command-line options are the following:

```
--help: help
--<moleculename>: specify which molecule is used in the calculation.
--dc-fields=: specify the range of the DC electric field (in kV/cm) by the␣
↪following way: start:end:step, example: --dc-fields=0:150:151.
--Jmax_calc=: specify the maximum value of J included in the calculation.
--Jmax_save=: specify the maximum value of J of the Stark curves saved in␣
↪the output file.
--Jmin=:      specify the minimum value of J included in the calculation.
--Mmax=:      specify the maximum value of M included in the calculation.␣
↪[TM: What is the default?  Jmax_calc?]
--isomer=:    specify which isomer is used, when <moleculename> has more␣
↪than one isomer defined in moleculeparameter.py
```

Example of using `cmistark_calculate_energy` with options:

```
cmistark_calculate_energy --isomer=0 --Jmax_calc=40 --Jmax_save=20 --3-
↪aminophenol --dc-fields=0:150:151
```

After executing this command line, the script will use cmistark packages to calculate stark energies of isomer 0 of 3-aminophenol up to J=40, and save results up to J=20 in an output file called `3-aminophenol.molecule`. The Stark curve of each quantum state starts from 0 to 150kV/cm with a step of 1kV/cm.

The output file `<moleculename>.molecule` stores Stark curves of individual quantum states a data format called HDF5. Such a file format can be read directly by using PyTables packages in Python. Two scripts in this program, `cmistark_plot_energy` and `cmistark_print_energy` are provided to easily access these `<moleculename>.molecule` files. Their options and descriptions are provided below.

1.2.3 cmistark_print_energy

The script file called `cmistark_print_energy` can access existing Stark files (`<moleculename>.molecule`) and print the stored curves on the screen. Options:

```
--help: help
--Jmin=, --Jmax=: specify min. or max. value of J
--Kamax=: specify max. value of Ka
--Mmin=, --Mmax=: specify min. or max. value of M
--isomer=: specify which isomer to print
```

Example of using `cmistark_print_energy` with options:

```
cmistark_print_energy --Jmin=0 --Jmax=2 --Mmin=1 --Mmax=1 <moleculename>.
↪molecule
```

1.2.4 cmistark_plot_energy

The script file called `cmistark_plot_energy` can access existing Stark files (`<moleculename>.molecule`) and plot the stored curves. Options:

```
--help: help
--energy-unit=: specify the unit of energy, options: MHz, invcm, J
--Jmin=, --Jmax=: specify min. or max. value of J
--Mmin=, --Mmax=: specify min. or max. value of M
--Kamax=: specify max. value of Ka
--states=: specify states to plot, format: J K_a K_c or J K_a K_c M, when␣
↪M is not specified,
           all M levels of the J state are plotted. Example: "000,1010"
--dipole: plot the effective dipole moments
--isomer=: specify which isomer to plot
```

Example of using `cmistark_plot_energy` with options:

```
cmistark_plot_energy --Jmin=0 --Jmax=2 --Mmin=1 --Mmax=1 <moleculename>.
↪molecule
```

1.3 Add a new molecule to CMIstark

Firstly, in the file called `moleculeparameter.py` (in the `cmistark` folder), add all relevant molecular constants/parameters as a function. See the examples provided in this file.

Next, in the script file `cmistark_calculate_energy` (in the `script` folder), add an option for calling the above added function for the mew molecule. See the examples provided in this file.

1.4 Extending and embedding CMIstark

1.4.1 Manually accessing the CMIstark results

Todo: here we need a short summary and then the references specified below.

The following example source code of Python shows how to read the curve for the 00000 state from `<moleculename>.molecule by` using PyTables:

```
import tables
import numpy
stark_file = "<moleculename>.molecule"
f=tables.openFile(stark_file)
array = f.getNode("/_0/_0/_0/_0/_0/dcstarkenergy")
print numpy.array(array.read())
```

CHAPTER 2

Developer documentation

The full documentation of the source code is available in the following subpages. This includes, for instance, available classes and methods, and their respective parameters and return values.

2.1 cmistark package

2.1.1 Submodules

2.1.1.1 cmistark.molecule module

2.1.1.2 cmistark.moleculeparameters module

2.1.1.3 cmistark.starkeffect module

2.1.2 Module contents

2.2 CMIstark developer guide

Here are a few simple guidelines to please be obeyed when working on CMIdiffract

- Document your code!
  - Use spinx-compatible docstrings to document all classs, methods, functions, etc.
- Write code that is compatible with the latest stable Python 3.x version.
- Make use of NumPy as much as possible.

2.2.1 Source code formatting

- CMIstark uses the 4-spaces standard for indentation of blocks.

- Do not use tabs, always expand to spaces.
- Try to not extend lines beyond 100 characters
- Keep the utf-8 coding directive in the first line
- Keep the Emacs local variables section at the end of all files, and try to stick to the directives (manually) when not using Emacs.

2.2.2 Version control (git) details

- CMIstark uses git (Stash) as a version control system with a central repositories in the DESY-CFEL-CMI stash server.
  - CMIstark uses the git-flow branching model
    - the principal development branch is `develop`
    - all new developments should be done on a `feature/` branch and, once ready, be branched into `develop`
  - never touch the branch `master` – this is to be done by the maintainers.
    - the `master` branch is only for releases, which for CMIstark are rare and surely only after discussion with me. There should never be any development done on `master`, nor any release preparations. The latter is done on `release/`, then the release is put onto `master`, and possibly necessary fixes are done on `hotfix/`!
  - Do not repeatedly branch feature branches into `develop` instead merge `develop` into your `feature/` branch.
  - General documentation work should always be made on `develop` (only)!
    - commit such doc-only updates as separate commits!
    - one can then merge these doc-only commits into `feature/` branches
  - **never implement a change twice** manually! Implement it on the most appropriate branch, then merge it into whatever branch you want to have it!
- make sure the git repo stays clean and one can follow it with our turning crazy!

2.2.3 State labels of stored Stark curves

In the output file `<moleculename>.molecule`, each Stark curve has a state label (J, K_a, K_c, M, isomer), which represents the adiabatic quantum number label of the rotational state in the field, as well as the type of isomer. J, K_a, K_c, M are integers, assuming no orbital angular momentum and spin of electrons and nuclear spins involved. For all types of rotors, the value of J is not less than zero.

For asymmetric tops and linear rotors, only states with positive M are stored, as all curves of nonzero M states are doubly degenerate. The values of both K_a and K_c are not less than zero for asymmetric tops, or set to zero for linear rotors. The state lable for linear rotors is thus (J, 0, 0, M, isomer).

For symmetric tops, states having products of K and M equal to $+|KM|$ and $-|KM|$ split in the DC electric field. In the output file states, corresponding to negative $|KM|$ are stored with negative K (and positive M); this is really an implementation detail and the sign stored with K in this case is always the sign of the product KM. We note that states with $K > 0$ and $M < 0$ also yield $-|KM|$. Thus,

all curves of nonzero M states in the output file are also doubly degenerate. Finally, the state label for prolate tops is (J, K, 0, M, isomer), and (J, 0, K, M, isomer) for oblate tops.

2.2.4 Structure of <moleculename>.molecule

For each state (J, K_a, K_c, M,isomer), the Stark energy as function of DC field strength is stored in the following structures:

```
/_J/_Ka/_Kc/_M/_isomer/dcfield
/_J/_Ka/_Kc/_M/_isomer/dcstarkenergy
```

The state label (J, K_a, K_c, M, isomer) is manifested as a dictionary strcuture here for storing state-specific information.

The following example source code of Python shows how to read the curve for the 00000 state from `<moleculename>.molecule` by using PyTables:

```
import tables
import numpy
stark_file = "<moleculename>.molecule"
f=tables.openFile(stark_file)
array = f.getNode("/_0/_0/_0/_0/_0/dcstarkenergy")
print numpy.array(array.read())
```

A script `cmistark_print_energies`, that provides ASCII output for specified conditions and states, is provided in the package for convenience.

2.2.5 Descriptions of source code files

Three source code files in `cmistark` folder provide all functions used to calculate and then write/read Stark curves. The above script files perform the calculations by calling these functions. The basic descriptions of each file in lib folder are as follows:

- `molecule.py`: perform the Stark effect calculation by calling functions from `starkeffect.py` and store results in an output file
- `moleculeparameter.py`: contain all molecular parameters of individual molecules
- `starkeffect.py`: contain all functions, equations and algorithms required for calculating the Stark effect.

(Only for stand-alone version) The descriptions of the rest of the files in the lib folder are as follows:

- `codata.py`: store most scientific constants
- `const.py`: call required math. and phys. constants from codata.py
- `convert.py`: perform unit conversions
- `hdf5.py`: read/write output files in the format of hdf5 via PyTables
- `moleculeproperty.py`: create a molecule (as an object) from a list of atoms
- `state.py`: create state labels and corresponding id numbers
- `util.py`: provide array operations

2.3 ToDo list

Todo: here we need a short summary and then the references specified below.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/extend line 8.)

Todo: References only give a name and a year (and possibly a town) are not useful... If it's what it is, say private communication. But try to avoid it.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/referen line 140.)

Todo: For NIST SPec DB I would only provide references for general http://www.nist.gov/pml/data/msd-di/index.cfm, http://www.nist.gov/pml/data/msd-tri/index.cfm, etc. or maybe even only one entry pointing at http://www.nist.gov/pml/data/molspec.cfm

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/referen line 143.)

Todo: Please write correct names, e.g., using the appropriate umlauts.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/referen line 148.)

Todo: journals should be printed in italics, volumes in bold

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/referen line 150.)

Todo: the output formatting should be much nicer... not sure how to do this, but please check. Maybe we can have a table or something like that?

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/referen line 152.)

Todo: Yuan-Pin, you misunderstood my earlier todos. You should first write about the general usagem adn then provide a specific example command at the end of the section of each tool. Put together, these shall give a working example. At every stage, you should provide info on the actual output. It will surely be very useful to even provide the output graphics of `plot`.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/cmistark/checkouts/stable/doc/user-guide.rst, line 25.)

2.4 References

Todo: References only give a name and a year (and possibly a town) are not useful... If it's what it is, say private communication. But try to avoid it.

Todo: For NIST SPec DB I would only provide references for general http://www.nist.gov/pml/data/msd-di/index.cfm, http://www.nist.gov/pml/data/msd-tri/index.cfm, etc. or maybe even only one entry pointing at http://www.nist.gov/pml/data/molspec.cfm

Todo: Please write correct names, e.g., using the appropriate umlauts.

Todo: journals should be printed in italics, volumes in bold

Todo: the output formatting should be much nicer... not sure how to do this, but please check. Maybe we can have a table or something like that?

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Bibliography

[Andreev1976] B.A. Andreev, A.V. Burenin, E.N. Karyakin, A.F. Krupnov, and S.M. Shchapin, *J. Mol. Spectrosc.* **62**, 125 (1976). DOI: 10.1016/0022-2852(76)90344-1

[Blanco2003] Blanco S et al, *J. Chem. Phys.* **119**, 880 (2003), DOI: 10.1063/1.1578996

[Bonham2009] R. A. Bonham, G. Cooper and A. P. Hitchcock, *J. Chem. Phys.* **130**, 144303 (2009) DOI: 10.1063/1.3108490

[Borst2001] D. R. Borst, T. M. Korter and D. W. Pratt, *Chem. Phys. Lett.* **350**, 485 (2001). DOI: 10.1016/S0009-2614(01)01344-6

[Brand2012] C. Brand, O. Oelterman, M. Wilke, J. Tatchen and M. Schmitt, *Chem. Phys. Chem.* **13**, 3134 (2012). *DOI: 10.1002/cphc.201200345 <http://dx.doi.org/10.1016/0009-2614(89)87081-2>*

[Brown1988] R. D. Brown, P. D. Godfrey, D. McNaughton and A. P. Pierlot, *J. Am. Chem. Soc.* **110**, 2329 (1988). DOI: 10.1016/0022-2860(88)80283-7

[Brown1989] R. D. Brown, P. D. Godfrey, D. McNaughton and A. P. Pierlot, *Chem. Phys. Lett.* **156**, 61 (1989). DOI: 10.1016/0009-2614(89)87081-2

[Brunken2006] Brünken, S.; McCarthy, M. C.; Thaddeus, P.; et al., *Astron. Astrophys.* **459**, 317-320 (2006)

[Cabana1976] A. Cabana, M.L.C. Pepin, and W.J. Lafferty, *J. Mol. Spectrosc.* **59**, 13 (1976). DOI: 10.1016/0022-2852(76)90037-0

[Caminati1990] W. Caminati and S. Dibernardo, J. Mol. Struct. 240, 253 (1990). DOI: 10.1016/0022-2860(90)80514-K

[Chang2014] Y.-P. Chang, F. Filsinger, B.G. Sartakov, J. Küpper, *Comp. Phys. Comm.*, **185**, 339 (2014). DOI: 10.1016/j.cpc.2013.09.001, arXiv:1308.4076 [physics]

[Clough1973] Clough, Beers, Klein, Rothman, *J. Chem. Phys.* **59**, 2254-2259 (1973). DOI: 10.1063/1.1680328

[DeLucia1971] F.C. De Lucia, R.L. Cook, P. Helminger, and W. Gordy, *J. Chem. Phys.* **55**, 5334 (1971). DOI: 10.1063/1.1675675

[DeLucia1974] F.C. DeLucia, P. Helminger, and W.H. Kirchhoff, *J. Phys. Chem. Ref. Data* **3**, 211 (1974). DOI: 10.1063/1.3253138

[Dorosh2007] Dorosh, Bialkowskajaworska, Kisiel, Pszczolkowski, J. Mol. Spec. 246, 228-232 (2007). DOI: 10.1016/j.jms.2007.09.010

[Dutta1985] Dutta, A and Jaman, A I, Pramana, *J. Phys.* **24** (1985). DOI: 10.1007/BF02894850

[Ecija2014] P. Ecija et al., *J. Phys. Chem. B* **118**, 5357 (2014). DOI: DOI: 10.1021/jp5014785

[Filsinger2008] F. Filsinger et al., Phem. Chem. Chem. Phys. 10, 666 (2008), DOI: 10.1039/b711888k

[Foster1965] P.D. Foster, V.M. Rao, R.F. Curl Jr., J. Chem. Phys. 43, 1064 (1965), DOI: 10.1063/1.1696821

[Franz2014] J. Franz and F. A. Gianturco, *Eur. Phys. J. D* **68**, 279 (2014). DOI: DOI: 10.1140/epjd/e2014-50072-0

[Gadhi1989] Gadhi, Wlodarczak, Legrand, & Demaison, Chem. Phys. Lett., 156, 401-404 (1989), DOI: 10.1016/0009-2614(89)87116-7

[Gordy:MMS:1984] Gordy and Cook, *Microwave Molecular Spectra*, John Wiley & Sons (1984)

[Hamaguchi1981] Hamaguchi, Buckingham, Jones, *Mol. Phys.* **43**, 6 (1981). DOI: 10.1080/00268978100102081

[Hansen2013] Hansen, Jonas L and Omiste Romero, Juan J and Nielsen, Jens Hedegaard and Pentlehner, Dominik and Küpper, Jochen and Gonz{‘a}lez-F{‘e}rez, Rosario and Stapelfeldt, Henrik, J. Chem. Phys. 139, 234313 (2013), DOI: 10.1063/1.4848735

[Helminger1985] Paul A. Helminger and Frank C. De Lucia JOURNAL OF MOLECULAR SPECTROSCOPY 111, 66-72 (1985), DOI: 10.1016/0022-2852(85)90069-4

[Hodgeson1963] J.A. Hodgeson, E.E. Sibert, and R.F. Curl, Jr., J. Phys. Chem. 67, 2833 (1963), DOI: 10.1021/j100806a079

[Horke] Daniel Horke

[Huber1979] Huber, K.P.; Herzberg, G., *Molecular Spectra and Molecular Structure. IV. Constants of Diatomic Molecules,* Van Nostrand Reinhold Co., (1979)

[Jaman1981] Jaman, A I and Nandi, R N and Ghosh, D K, J.of Mol. Spec., 86, 269.274 (1981), DOI: 10.1016/0022-2852(81)90279-4

[Kang2005] Kang, Korter, Pratt, J. Chem. Phys. 122, 174301 (2005), DOI: 10.1063/1.1883635

[Kim1976] Kim, Hameka, Svendsen, *Chem. Phys. Lett.* **41**, 2 (1976). DOI: 10.1016/0009-2614(76)80794-4

[Kong2005] QY Kong and J Kim and M Lorenc and TK Kim and H Ihee and M Wulff, J. Phys. Chem. A, 109, 10451-10458 (2005), DOI: 10.1021/jp053390q

[Korter1998] Korter, Pratt, Küpper, *J. Phys. Chem.* A **102**, 7211 (1998), DOI: 10.1021/jp982456x

[Kuepper2010] J. Küpper, 2010

[Mckellar1976] Mckellar and Goetz and Ramsay, *The Astrophysical Journal* **207**, (1976), DOI: 10.1086/154533

[Miller2009] Diane M. Miller, Philip J. Morgan and David W. Pratt, *J. Phys. Chem.* A, **113**, 6964-6970 (2009), DOI: 10.1021/jp9017585

[MinaCamilde1996] Mina-Camilde et al. JCE 73 p.804 (1996) http://web.ist.utl.pt/farinha/LQF/pdf_files/CO_ref4_JCE1986.pdf

[Neil:JMolSpec269:21] J. L. Neill, S. T. Shipman, L. Alvarez-Valtierra, A. Lesarri, Z. Kisiel, and B. H. Pate, "Rotational spectroscopy of iodobenzene and iodobenzene-neon with a direct digital 2-8 GHz chirped-pulse Fourier transform microwave spectrometer" *J. Mol. Spec.* **269**, 21 (2011).

[NISTCCCBD_CO] NIST Computational Chemistry Comparison and Benchmark DataBase http://cccbdb.nist.gov

[NISTCCCBD_SO] NIST Computational Chemistry Comparison and Benchmark DataBase - SO

[NISTspecDB_H2O] NIST Spectral Database - H2O

[NISTspecDB_OCS] NIST Spectral Database - OCS

[Oelterman2012] Oeltermann, Olivia and Brand, Christian and Engels, Bernd and Tatchen, Jorg and Schmitt, Michael, Phys. Chem. Chem. Phys. 14, 10266 (2012), DOI: 10.1039/C2CP41094J

[Orcutt1963] *J. Chem. Phys.* **39(3)**, 605 (1963), DOI: 10.1063/1.1734300

[Patel1979] Patel, D. and Margolese, D. and Dyke, T. R., J. Chem. Phys. 70, 2740-2747 (1979), DOI: 10.1063/1.437860

[Pietila1996] Pietilä, Koivusaari, Alanko, & Anttila, Mol. Phys. 87, 523 (1996), DOI: 10.1080/00268979600100351

[Reinartz1974] Reinartz, J. & Dymanus, A., Chemical Physics Letters **24**, 346-351 (1974) DOI: 0.1016/0009-2614(74)85275-9

[Poynter1975] R. L. Poynter, R. K. Kakar, The Astrophysical Journal Supplement Series **277**, 87-96 (1975)

[Puzzarini2011] Puzzarini, Cristina; Barone, Vincenzo, Phys. Chem. Chem. Phys. 13, 7158-7166 (2011), DOI: 10.1039/C0CP02636K

[Rychlewski1980] J. Rychlewski1980, *Mol. Phys.* **41**, 4 (1980) DOI: 10.1080/00268978000103191

[Roesch2011] Daniel Rösch, Basel, 2011

[Scharpen1970] L.H. Scharpen, J.S. Muenter, and V.W. Laurie, J. Chem. Phys. 53, 2513 (1970), DOI: 10.1063/1.1674355http://dx.doi.org/10.1063/1.1674355

[Shostak1991] Shostak, Ebenstein, and J. Muenter, J. Chem. Phys., 94, 5875 (1991), DOI: 10.1063/1.460471

[Songhee2011] Songhee Han, 2011

[Steenbeckeliers1973] G. Steenbeckeliers, and J. Bellet, J. Mol. Spectrosc. 45, 10 (1973), DOI: 10.1016/0022-2852(73)90171-9

[Trefler1968] M. Trefler and H. P. Gush, *Phys. Rev. Lett.* **20**, 703 (1968) DOI: 10.1103/PhysRevLett.20.703http://dx.doi.org/10.1103/PhysRevLett.20.703

[Trippel2012] Trippel, Chang, Stern, Mullins, Holmegaard, Küpper, Phys. Rev. A 86, 033202 (2012) DOI: 10.1103/PhysRevA.86.033202

[Veseth1974] Veseth, Lofthus, Mol. Phys. 27, 2 511-519 (1974), DOI: 10.1080/00268977400100461

[Wilcox2011] D. Wilcox, A. Shirar, O. Williams, B. Dian, Chem. Phys. Lett. 508, 10 (2011), DOI: 10.1016/j.cplett.2011.04.001

[Wohlfart2008] Wohlfart, Schnell, Grabow, Küpper, J. Mol. Spec. 247, 119-121 (2008), DOI: 10.1016/j.jms.2007.10.006

[Wlodarczak1987] Wlodarczak, Boucher, Bocquet, & Demaison, J. Mol. Spectros., 124, 53-65 (1987), DOI: 10.1016/0022-2852(87)90120-2