# HELIOS-K

***Release 2.0***

**Simon Grimm**

**Jan 29, 2024**

# SETUP:


1 Setup 3

2 Databases 7

3 Running HELIOS-K 21

4 OPTIONS 25

5 Output Files 43

6 Atomic Opacities plots 47

7 Citations 71

HELIOS-K calculates opacity functions for planetary atmopheres by using opacity line lists from different databases. Before the opacity functions can be calculated, the line lists need to be downloaded and preprocessed into binary files that can be read from HELIOS-K.

HELIOS-K provides tools to automatically download and preprocess files from the ExoMol, HITRAN, HITEMP, NIST, Kurucz and VALD3 databases.

HELIOS-K is running on GPUs and require a Nvidia GPU with compute capability of 3.0 or higher.

Developed by Simon Grimm & Kevin Heng.
University of Bern.

CHAPTER

ONE

# SETUP

## 1.1 Requirements

The opacity calculation from HELIOS-K is running on GPUs and require a Nvidia GPUi with compute capability of 3.0 or higher. It can run on Nvidia Tesla, GeForce and Quadro GPUs.

The code needs the CUDA toolkit to be installed. This can be downloaded from https://developer.nvidia.com/cuda-downloads.

Helper code for downloading the files and preprocessing are written in C++ and Python3. They require the following libraries:

- exomol.py and exomol2.py
  - bs4
  - requests
  - sys
  - os
  - subprocess
  - numpy
  - argparse
  - math
- Kurucz2.py
  - numpy
  - math
  - struct
  - os
  - argparse
- nist_ELevels2.py and nist_Lines3.py
  - sys
  - argparse
  - csv
  - requests

- nist_partition.py
  - numpy
  - sys
  - argparse
- nist_Lines2.py
  - numpy
  - struct
  - math
  - csv
  - pandas
  - argparse
  - re
- vald_request.py
  - selenium
  - argparse
- vald_download.py
  - numpy
  - math
  - struct
  - os
  - argparse
- vald.py
  - numpy
  - math
  - struct
  - os
  - argparse

Note, when using a computing cluster, all these libraries can be installed locally in the home directory with:

```
pip3 install --user <package name>
```

## 1.2 Compilation

HELIOS-K can be compiled with the provided Makefile by typing

```
make SM=xx
```

into the terminal, where `xx` corresponds to the compute capability of the installed GPU. For example use `make SM=20` for compute capability 2.0, or `make SM=35` for 3.5. A table with all compute capabilities can be found here. On a computing cluster, eventually the CUDA module must be loaded before compiling the code.

### 1.2.1 On Windows machines

If using Cygwin on Windows, then HELIOS-K can be compiled the same way with `make SM=xx`. If using the Windows Command Prompt, type `nmake -f MakefileW SM=xx`. Note, that the Windows C++ compiler `cl` must be installed, and the compiler path must be loaded in the shell. If this is not the case, it can be loaded similar to this command: `call "C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools\vsvars32.bat"` , where the exact path and file name must eventually be changed.

CHAPTER

TWO

# DATABASES

## 2.1 Download and pre-process the line lists

If you have access to the hulk cluster at the University of Bern, then most of the pre-processed files are already available at `scratch/sigrimm/EXOMOL/`. This files can then directly be used for the opacity calculation.

### 2.1.1 Supported line list databases

HELIOS-K supports line lists from the HITRAN, HITEMP, ExoMol, Kurucz, NIST or VALD databases. Before the line lists can be used, they have to be pre-processed into binary files. This saves in generally memory space and allows HELIOS-K to read the line lists in a more efficient way.

The next section explains the structure of these files and shows how to generate them.

### 2.1.2 The `<species_name>.param` file

Each molecular or atomic species, which will be processed by HELIOS-K, needs a `<species_name>.param` file, which contains all necessary information about the line list database and other parameters. This files can be generated automatically by the tools in HELIOS-K. However, when using a different database, they must be created manually.

### 2.1.3 Example

An example for HITRAN 2016 H2O is given in the `data/01_hit16.param` file, which is shown here and available in the repository. An example for HITEMP and ExoMol are also given in the repository files `data/01_HITEMP2010.param` and `data/1H2-16O__BT2.param`.

```
01_hit16.param
------------------------------------------------------------------
Database = 0
Molecule number = 1
Name = hit16
Number of Isotopologues = 7
#ID Abundance      Q(296K)   g     Molar Mass(g)  partition file :
11 0.997317        174.58    1     18.010565      q1.txt
12 0.002000        176.05    1     20.014811      q2.txt
13 3.718840E-04    1052.14   6     19.01478       q3.txt
14 3.106930E-04    864.74    6     19.01674       q4.txt
15 6.230030E-07    875.57    6     21.020985      q5.txt
```

(continues on next page)

(continued from previous page)

```
16 1.158530E-07    5226.79   36    20.020956      q6.txt
17 2.419700E-07    1027.80   1     20.022915      q129.txt
Number of columns in partition File = 2
Number of line/transition files = 1
Number of lines per file :
304225
Line file limits :
0
30000
#ExoMol :
Number of states = 0
Number of columns in transition files = 0
Default value of Lorentzian half-width for all lines = 0
Default value of temperature exponent for all lines = 0
Version = 0
-----------------------------------------------------------------------
```

### 2.1.4 Arguments

- Database: Indicate which database format to use.
- 0 = Hitran or HITEMP
- 2 = ExoMol
- 30 = Kurucz atomic database
- 31 = NIST atomic database
- Molecule number: This is optional and follows an old HELIOS-K version.
- Name: name of the line list files.
- Number of isotopologues in the line list files.
- For each isotopologue:
  - ID: for Hitran and HITEMP the same as the first three digits in the `*.par` files e.g. 11 for 1H2-18O, or 2A for 18O-13C-17O. For ExoMol no meaning.
  - Abundance, only relevant for more than one isotopologue.
  - Reference partition function value. For ExoMol no meaning.
  - Molar mass in g.
  - Name of the partition function file.
- Number of columns in the partition function file.
- Number of line/transition files.
- Line by line, the number of molecular/atomic lines in each line/transition file.
- Line by line, the wavenumber range of each line/transition file, must have one entry more than the number of line/transition files to cover all lower and upper range limits.
- Number of states in ExoMol `*.states`files. For other databases no meaning.
- Number of columns in ExoMol `*.trans`files. For other databases no meaning.

- ExoMol default value of Lorentzian half-width. For other databases no meaning.
- ExoMol default value of temperature exponent. For other databases no meaning.
- Version of the line list.

## 2.2 ExoMol

### 2.2.1 Step 1, Species Properties

The HELIOS-K repository provides a file called `Exomol_species.dat`. This file contains all available species from the ExoMol database. The file format is:

`Molecule name , Isotopologue name, Full name, path on exomol.com, range of *.trans files, number of *.trans files, number of digits in *.trans file ranges.`

The full name of the species contains the isotopologue and the line list name. This full name should be used when as species name for the opacity calculation, e.g. `1H2-16O__BT2`.

The `Exomol_species.dat` file can be recreated or updated by running:

```
python3 exomol2.py
```

### 2.2.2 Step 2, Download the files and create the <species_name>.param file

The ExoMol files can be downloaded with the a python script as:

```
python3 exomol.py -M <id>
```

where <id> is the full species name e.g. `1H2-16O__BT2`. The script needs the file `Exomol_species.dat` to be availalbe. If this file needs to be updated, it can be done by running `python3 exomol2.py`.

The `exomol.py` script automatically writes the <species_name>.param files for each molecule.

### 2.2.3 Step 3, create the binary files

The downloaded line list files must be pre-processed into binary files with the following code:

```
./prepareExomol -M < id >
```

where < id > is the full species name, e.g. `1H2-16O__BT2`. After this step, the `*.trans` and `*.states` files from ExoMol can be deleted.

### 2.2.4 Step 4, data path

Include the path of the directory, which contains the obtained binary files, the `*.pf`file and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

## 2.3 ExoMol super-lines

### 2.3.1 Step 1, Species Properties

The HELIOS-K repository provides a file called `Exomol_species.dat`. This file contains all available species from the ExoMol database. The file format is:

`Molecule name , Isotopologue name, Full name, path on exomol.com, range of *.trans files, number of *.trans files, number of digits in *.trans file ranges.`

The full name of the species contains the isotopologue and the line list name. This full name should be used when as species name for the opacity calculation, e.g. `1H2-16O__BT2`.

The `Exomol_species.dat` file can be recreated or updated by running:

```
python3 exomol2.py
```

### 2.3.2 Step 2, Download the files and create the `<species_name>.param` file

The ExoMol super line files can be downloaded with the a python script as:

```
python3 exomol.py -M <id> -D 3 -Temp <T>
```

where <id> is the full species name e.g. `1H2-16O__POKAZATEL`. The `-D 3` specifies to download the super-lines instead of the full transition and states files. The <T> specifies the temperature of the super lines in K, e.g. `-Temp 1000`. When multiple temperatures are needed, the each temperature file must be downloaded and processed individually.

The script needs the file `Exomol_species.dat` to be availalbe. If this file needs to be updated, it can be done by running `python3 exomol2.py`.

The `exomol.py` script automatically writes the `<species_name>.param` files for each molecule.

### 2.3.3 Step 3, create the binary files

The downloaded line list files must be pre-processed into binary files with the following code:

```
./prepareExomolSuper -M < id >
```

where < id > is the full super-line file name, e.g. `1H2-16O__POKAZATEL__00000-41200__1000K_super.`. After this step, the `*.super` files from ExoMol can be deleted.

### 2.3.4 Step 4, data path

Include the path of the directory, which contains the obtained binary files, the `*.pf`file and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

### 2.3.5 Running HELIOS-K

When running HELIOS-K, the temperature of the calculation must corespond to the temparture of the super-lines. Otherwise the result is not not correct.

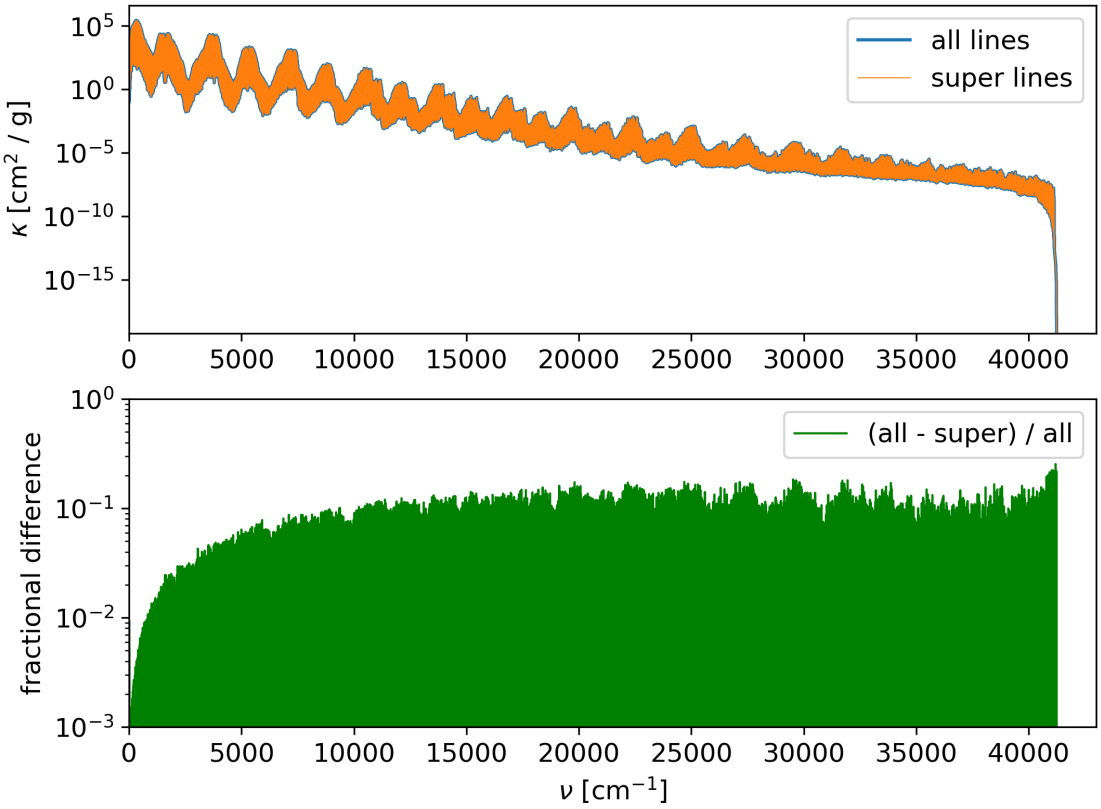



Fig. 2.1: Comparison of the full POKAZATEL 1H2-16O opacity and super-lines for 1000 K. The voigt profiles are truncated at 100 cm^-1.

## 2.4 HITRAN

### 2.4.1 Step 1, Species Properties

The HELIOS-K repository provides a file called `Hitran_species.dat`. This file contains all available species of the Hitran database. The file format is:

`Species ID, Abundance, Molecular Mass in g/mol, Q(296K), partition function file, gi, Isotopologue Formula` The same information can be found at `https://hitran.org/docs/iso-meta/`.

The species ID consitst of a two digits molecule ID and a one digit local isotopologue ID. Note that the local isotogologue ID can sometimes consist of non numerical values. e.g A or B,

For identifying a species, the molecule number and the isotopologue number should be combinded, e.g. `01_1` for 1H2-16O or `01` for all isotopogolues from H20.

The `Hitran_species.dat` file can be recreated or updated with the python code `hitran2.py`.

### 2.4.2 Step 2, Download the files

The line list files must be downloaded manually from *www.hitran.org* `, and note that it is necessary to register on the hitran homepage. To download the files, select DataAcess and then Line-by-line. Select the molecule id, select all isotopologues (single isotopologues can be filtered later), leave wavenumber range blank, select `.par`file and store the file on your computer under the name `Molecule-ID_hit16.par`, e.g. 01_hit16.par for H2O.

Download all the necessary partition function files from Documentation/Isotogologues.

### 2.4.3 Step 3, create <species_name>.param file and the binary files

All necessary files can be created with:

```
./hitran -M < molecule ID > -ISO < isotopologue ID > -in < line list name >
```

The <molecule ID> is the two digit molecule number, e.g. 01 for H2O. The <isotopologue ID> is the hitran internal isotopologue identifier, e.g. 1 for 1H2-16O. The <line list name> is the name that was given in the download section, e.g. `hit16`.

### 2.4.4 Step 4, data path

Include the path of the directory, which contains the obtained binary files, the `*.txt` partition function files and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

## 2.5 HITEMP

### 2.5.1 Step 1, Species Properties

For HITEMP, the same `Hitran_species.dat` is needed as for HITRAN. See *Step 1, Species Properties*.

### 2.5.2 Step 2, Download the files

The line list files must be downloaded manually from `https://hitran.org/hitemp/`. The same partition function files are needed as for HITRAN, see *Step 2, Download the files*.

### 2.5.3 Step 3, create < species >.param file and the binary files

All necessary files can be created with:

```
./hitran -M < molecule ID > -ISO < isotopologue ID > -in < line list name >
```

The <molecule ID> is the two digit molecule number, e.g. 01 for H2O. The <isotopologue ID> is the hitran internal isotopologue identifier, e.g. 1 for 1H2-16O. The <line list name> is the name that was given in the download section, e.g. `HITEMP2010`.

### 2.5.4 Step 4, data path

Include the path of the directory, which contains the obtained binary files, the `*.txt` partition function files and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

### 2.5.5 References

HITEMP gives an example how to cite their work:

```
How to cite HITEMP

When using the HITEMP database, please cite the appropriate HITEMP article (e.g., [1])␣
↪as well as the original sources of data. To assist the user, each line transition has␣
↪reference codes that are provided for each parameter and are consistent with those in␣
↪HITRAN for the same molecule. We hope that you will find HITEMP helpful in your␣
↪research.

References

[1] L. S. Rothman, I. E. Gordon, R. J. Barber, H. Dothe, R. R. Gamache, A. Goldman, V.␣
↪Perevalov, S. A. Tashkun, J. Tennyson, "HITEMP, the high-temperature molecular␣
↪spectroscopic database", J. Quant. Spectrosc. Radiat. Transfer 111, 2139-2150 (2010).␣
↪[link to article] [ADS]

[2] L. S. Rothman, R. B. Wattson, R. R. Gamache, J. Schroeder, A. McCann, "HITRAN, HAWKS␣
↪and HITEMP: High-Temperature Molecular Database", Proc. SPIE, Atmospheric Propogation␣
↪and Remote Sensing IV 2471, 105-111 (1995). [link to article] [ADS]

[3] R. J. Hargreaves, I. E. Gordon, L. S. Rothman, S. A. Tashkun, V. I. Perevalov, S. N.␣
↪Yurchenko, J. Tennyson, H. S. P. Müller, "Spectroscopic line parameters of NO, NO2,␣
↪and N2O for the HITEMP database", J. Quant. Spectrosc. Radiat. Transfer 232, 35-53␣
↪(2019). [link to article] [ADS]

[4] R. J. Hargreaves, I. E. Gordon, M. Rey, A. V. Nikitin, V. G. Tyuterev, R. V.␣
↪Kochanov, L. S. Rothman, "An accurate, extensive, and practical line list of methane␣
↪for the HITEMP database", Astrophys. J. Supp. Ser. 247, 55 (2020). [link to article]␣
↪[ADS]

[5] G. Li, I. E. Gordon, L. S. Rothman, Y. Tan, S.-M. Hu, S. Kassi, A. Campargue, E. S.␣
↪Medvedev, "Rovibrational line lists for nine isotopologues of the CO molecule in the␣
↪X1+ ground electronic state", Astrophys. J. Supp. Ser. 216, 15 (2015). [link to␣
↪article] [ADS]
```

## 2.6 Kurucz Database

### 2.6.1 Step 1, Download the files

The needed files can be downloaded with the following command:

```
python3 Kurucz3.py -D 1 -Z <z> -I <i>
```

where <z>is the atomic number and <i>the ionization state

- D 1, means to download the file `gfallwn08oct17.dat` and all available partition function files.
- D 2, means to download only the partition function files.
- D 0, means no download from the Kurucz website.
- Z i, means to use species with atomic number Z = i.
- Z -1, means to use all species.
- I j, means to use ionisation state j, where 0 are neutral atoms.
- I -1, means to use ionisation states 0, 1 and 2.

**If the source file `gfallwn08oct17.dat` should be changed, then the script `Kurucz3.py` and `Kurucz4.py`** must be modified.

This step writes a file `gfnew<Z><I>.dat` containing the line list data.

### 2.6.2 Step 2, Correct isotope fractions

The file `gfallwn08oct17.dat` contains some incorrect isopte fractions for hyperfine splitted lines. This can be corrected with:

```
python3 KuruczHyper.py -Z <z> -I <i>
```

This wrtie a new file `gfnewCorr<Z><I>.dat` with the corrected data.

### 2.6.3 Step 3, calculate natural broadening and create the binary files and `< species >.param` files

```
python3 Kurucz4.py -Z <z> -I <i>
```

This step creates the `< species >.param` and the **.bin* files. It also calculates missing natural broadening coefficients. This can take a while to complete.

### 2.6.4 Step 4, data path

Include the path of the directory, which contains the obtained binary files, the `*.pf` partition function files and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

## 2.7 NIST Database

### 2.7.1 Step 1, Download the Energy levels

This step can be done either manually or by using a script.

- For manuall download:
  - visit `https://physics.nist.gov/PhysRefData/ASD/levels_form.html`
  - enter the species and ionization state, e.g. `Z= 3 0`
  - select the Format output to Tab-delimited
  - select the `g` option
  - store the data in a file called `NIST_ELevels<Z><I>.dat`
  - remove all " in the file
- Automatical download:
  - type `python3 nist_ELevels2.py -Z <z> -I <i>`, where <z>is the atomic number and <i>the ionization state , e.g. -Z 3 -I 0
  - The script will download the data and store it in a file called `NIST_ELevels<Z><I>.dat`.

### 2.7.2 Step 2, Generate the partition functions

Run `python3 nist_partition.py -Z <z> -I <i>`. This script will read the previously produces file `NIST_ELevels<Z><I>.dat`, and writes a `*.pf` file.

### 2.7.3 Step 3, Download atomic masses

- visit `https://www.nist.gov/pml/atomic-weights-and-isotopic-compositions-relative-atomic-masses`
- select `All Elements`
- select `Linearized ASCII Output`
- select `All isotopes`
- click `Get Data`and store the data in a file called `masses.txt`

### 2.7.4 Step 4, Download the line list

This step can be done either manually or by using a script.

- For manuall download:
  - visit https://physics.nist.gov/PhysRefData/ASD/lines_form.html
  - enter the species and ionization state, e.g. `Z= 3 0`
  - select `Show Advanced Settings`
  - select the Format output to csv
  - Select `Wavenumbers (in cm-1)` (keep `Observed`, `Ritz` and `Uncertainties` selected)
  - Select `Vacuum (all wavelengths)`
  - Select `g`
  - click `Retrieve Data`and store the data in a file called `NIST_Lines<Z><I>.dat`.
  - remove all `?, =, [,],( and )`from the file
- Automatical download:
  - type `python3 nist_Lines3.py -Z <z> -I <i>`.
  - The script will download the data and store it in a file called `NIST_Lines<Z><I>.dat`.

### 2.7.5 Step 5, Create `<species_name>.param` file and the binary files

All necessary files can be created with:

```
python3 nist_Lines2.py -Z <z> -I <i>
```

This step includes the calculation of the natural broadening coefficients. This can take o moment to complete.

### 2.7.6 Step 6, data path

Include the path of the directory, which contains the obtained binary files, the `*.pf` partition function files and the `*.param` file to the HELIOS-K `param.dat` file under `pathToData`.

### 2.7.7 References

NIST gives an example how to cite their work:

```
Example of how to reference these results:
Kramida, A., Ralchenko, Yu., Reader, J., and NIST ASD Team (2019).
NIST Atomic Spectra Database (ver. 5.7.1), [Online]. Available: https://physics.nist.gov/
↪asd [2020, July 22].
National Institute of Standards and Technology, Gaithersburg, MD. DOI: https://doi.org/
↪10.18434/T4W30F


@Misc{NIST_ASD,
author = {A.~Kramida and {Yu.~Ralchenko} and
```

(continues on next page)

(continued from previous page)

```
J.~Reader and {and NIST ASD Team}},
HOWPUBLISHED = {{NIST Atomic Spectra Database
(ver. 5.7.1), [Online]. Available:
{\tt{https://physics.nist.gov/asd}} [2020, July 22].
National Institute of Standards and Technology,
Gaithersburg, MD.}},
year = {2019},
}
```

## 2.8 VALD3 Database

### 2.8.1 Step 1, Request the files

The download of the line lists can either be done manually through the website, or in a more automated way, by using navigation scripts.

Before the VALD data can be acessed, one has to register on the VALD website.

Attention, there is a limit of 100000 transition lines per query. Some species have more lines than that. Is this case, the wavenumber ranges have to be split into two parts. This is especially the case for Fe and Cr.

#### Method A, manual request

- Vist `http://vald.astro.uu.se/`
- Register with email address
- Choose Mirror Server
- Click `Extract Element`
- Click `Unit selections` and select:
  - Energy level units: cm-1
  - Give wavelenths in medium: vacuum
  - Wavelength units: cm-1
  - Van der Waals syntax: Default
  - Isotopic scaling: Off
  - click `Save settings`
- click again `Extract Element`
- check the units, they must be `Energy unit: 1/cm - Medium: vacuum - Wavelength unit: 1/cm - VdW syntax: default`
- Enter Starting wavenumber, e.g. 0.001
- Enter Ending wavenumber : e.g. 1000000
- Enter Element: e.g Fe 1, (ionizaion number = 1 means neutral)
- Extraction format : Long format
- Retrieve data via : FTP

- Select `Include HFS splitting`
- Enter a comment in `Optional comment for request`, for example the element and ionization number.
- Click `Submit request`

#### Method B, using scripts

Also this mehod requires to first visit the web interface manually and to choose the correct units, the same way as Method A. After that, the vald_request.py script can navigate the webiste and fill in the necessary webforms. For this, a geckodriver needs to be installed.

Then type:

```
python3 vald_request.py -Z <z> -I <i> -u <email address>
```

where <z>is the atomic number and <i>the ionization state (i = 0 is neutral) The email address must be the same as the user account from the VALD website.

### 2.8.2 Step 2, download the files

After a few minutes you will get an email from VALD with a link to download the data file.

#### Method A, manual download

Download the file and unpack it. Rename the file to `VALD<Z><I>.dat`. E.g VALD2600.dat for Z = 26, I = 0 (I = 0 means neutral atoms).

#### Method B, using scripts

The requested files are stored temporarely on the VALD webiste with a name similar to `SimonGrimm.4199418`, where the first part is the user name, and the second part is the job number. The vald_download.py script can download all requested files, scan which element they contain and rename them according to the element number.

```
python3 vald_download.py -jobstart <start> -jobend <end> -jobname <name>
```

The name is the user name from the VALD webiste, e.g SimonGrimm. The jobstart is the first jobnumber (check email from VALD) .e.g 4199418. The jobend is the last jobnumber (check email from VALD) .e.g 4199420.

The files are renamed as `VALD<Z><I>.dat`.

### 2.8.3 Step 3, partition function files

Partition functions are not available in the VALD database. Therefore we use the partition functions calculated from the NIST database.

### 2.8.4 Step 4, calculate natural broadening coefficients and create the binary files and < species >.param files

For many species, the vald database provides broadening coefficients, but not for all. The missing coefficients are calculated in this step. That can take a while to complete.

```
python3 vald.py -Z <z> -I <i>
```

This step creates the < species >.param and the *.*bin* files.

### 2.8.5 Step 5, data path

Include the path of the directory, which contains the obtained binary files, the *.pf partition function files and the *.param file to the HELIOS-K param.dat file under pathToData.

### 2.8.6 References

Vald provides reference files for each species. We rename these *.bib files the same way as the data files. When using data from VALD, all these references must be cited in publications.

We list here also the citation conditions from VALD:

```
A short note on using VALD data :

The VALD data and services are provided free of charge on the following conditions:

    1. If VALD has been used in your research work, you agree to include the acknowledge
       below as well as citations to the following papers where appropriate:

    "This work has made use of the VALD database, operated at Uppsala University, the
     Institute of Astronomy RAS in Moscow, and the University of Vienna."

        Ryabchikova T., Piskunov, N., Kurucz, R.L., et al., Physics Scripta, vol 90,␣
↪issue 5, article id. 054005 (2015), (VALD-3)
        Kupka F., Ryabchikova T.A., Piskunov N.E., Stempels H.C., Weiss W.W., 2000,␣
↪Baltic Astronomy, vol. 9, 590-594 (2000), (VALD-2)
        Kupka F., Piskunov N.E., Ryabchikova T.A., Stempels H.C., Weiss W.W., A&AS 138,␣
↪119-133 (1999), (VALD-2)
        Ryabchikova T.A. Piskunov N.E., Stempels H.C., Kupka F., Weiss W.W. Proc. of the␣
↪6th International Colloquium on Atomic Spectra and Oscillator Strengths, Victoria BC,␣
↪Canada, 1998, Physica Scripta T83, 162-173 (1999), (VALD-2)
        Piskunov N.E., Kupka F., Ryabchikova T.A., Weiss W.W., Jeffery C.S., A&AS 112,␣
↪525 (1995) (VALD-1)

    2. You also agree to reference to original source(s) of the line data that are␣
↪important for your research.
       A complete list of references is available from the bibliography section of the␣
↪VALD manual. A short list
       of references is compiled and sent to you with every reply from VALD.
```

CHAPTER

# THREE

# RUNNING HELIOS-K

## 3.1 Using HELIOS-K

Before HELIOS-K can be used, the molecular or atomic line-lists must be downloaded and pre-processed. HELIOS-K provides some scripts for that, which are described in the previous section.

All necessary parameters for HELIOS-K are set in the file `param.dat`, which is described next. Some parameters can also be set by console arguments.

HELIOS-K can be started with

```
./heliosk
```

followed by optional console arguments, which are listed later.

### 3.1.1 HELIOS-K Input parameters

The input parameters can be specified in the `param.dat` file. The used parameters are listed here, the order can not be changed.

- name: This name will appear in the output filenames.
- T: Temperature in Kelvin
- P: Pressure in standard atmospheres atm (1atm = 1.01325 bar, 1bar = 0.98692367 atm, see https://en.wikipedia.org/wiki/Standard_atmosphere_(unit))
- PFile: A '-' ignores this option, otherwise this option specifies a filename which contains multiple values for P.
- Species Name: The name of the molecular or atomic param file. e.g. 01_hit16 or 1H2-16O__BT2.
- SpeciesFile: A '-' ignores this option, otherwise this option specifies a filename which contains multiple species for gas mixtures.
- ciaSystem: A '-' ignores this option, otherwise a cia system is read here. supported are H2-H2, H2-H2_eq, H2_H2_norm, H2-He, H2-He_eq, H2-He_norm, H2-CH4_eq, H2-CH4_norm and H2-H.
- pathToData: The location where the HITRAN or HITEMP data files are located, e.g. pathToData = ../HITEMP/ , pathToData = /data/HITEMP/ or empty when the files are in the same directory pathToData =
- numin: minimum wavenumber in 1/cm
- numax: maximum wavenumber in 1/cm
- dnu: spatial resolution in wavenumber in 1/cm
- Nnu per bin: The number of points in wavenumbers nu per bin.

  - 0; Nnu is ignored and dnu is used
  - otherwise, dnu is overwritten.
- cutMode: Set the voigt profile cutting mode:
  - 0: cut at absolute wavenumbers
  - 1: cut at multiple values of Lorentz half widths
  - 2: cut at multiple values of Doppler half widths
- cut: value where to cut the line wings, according to the cutMode, if cut = 0, then no cutting is performed.
- doResampling:
  - 0: no resampling is done.
  - 1: all the sorted opacity functions per bin are resampled with a Chebyshev polynomial, and the coefficients from all bins are written to the file `Out_<name>_cbin.dat`.
  - 2: all the sorted opacity functions per bin are resampled with a Chebyshev

  polynomial, and the coefficients from each bin are written to a sepparate file `Out_<name>_cbin<bin_number>.dat`.
- nC: Number of Chebyshev coefficients use for the resampling.
- doTransmission: Compute the transmission function tau for a set of column masses m.
  - 0: no transmission function is calculated.
  - 1: the transmission function per bin is calculated, and all bins are written to the file `Out_<name>_tr.dat`.
  - 2: the transmission function per bin is calculated, and each bin is written to a sepparate file `Out_<name>_tr<bin_number>.dat`.
- nTr: nummber of points used for the transmission function.
- dTr: spacing of the points used for the transmission function in exp space: m_i = exp((i - nTr/2) * dTr)
- doStoreFullK:
  - 0: the opacity functions is not written to a file.
  - 1: the full unsorted opacity function is written to the text file `Out_<name>.dat`.
  - 2: the full unsorted opacity function is written to the binary file `Out_<name>.bin`.
  - -1: the opacity functions is not calculated, but read in from a text file. The name of the file is specified in the next argument.
  - -2: the opacity function is nor calculated, but read in from a binary file. The name of the file is specified in the next argument.
- pathToK: path and filename to read in an opacity function. When left blank, then no file is read in.
- doStoreSK:
  - 0: the resampled opacity function is not written to a file.
  - 1: all bins from the resampled opacity function are written to the same file `Out_<name>_bin.dat`.
  - 2: all bins from the resampled opacity function are written to a sepparate file with names `Out_<name>_bin<bin_number>.dat`.
- nbins: number of bins

- binsfile: A '-' ignores this option, otherwise this option specifies a filename which contains the edges of the bins, which can be irregular. This option overrides the numin, numax and nbins arguments.
- OutputEdgesFile: A '-' ignores this option, otherwise this option specifies a file name which contains the edges of the output locations in y for each bin.
- kmin: minimal value for the opacity function
- qalphaL: q value in the Lorentzian half width q = Pself / P
- gammaF: scaling factor for the Lorentzian half width gamma factor, gamma = gamma * gammaF
- doMean:
  - 0: nothing is done here.
  - 1: Calculate the Planck and Rosseland opacity means.
- Units: The units of the opacities.
  - 0: cm^2 / g,
  - 1: cm^2 / molecule
- ReplaceFile:
  - 1: all existing output files are overwritten.
  - 0: the data is appended to the existing files.
- profile:
  - 1: Voigt
  - 2: Lorentzian
  - 3: Gaussian
  - 4: Integrated Binned Gaussian
- subLorentzianfile: A '-' ignores this option, otherwise this option specifies a file name which contains the chi factor parameters for sub-Lorentzian CO2 profiles.
- removePlinth:
- 0: nothing is done here
- 1: the plinth (base) is removed from cutted line profiles
- doTuning:
  - 0: nothing is done here
  - 1: use self tuning routines to specify the best kernel parameters.

### 3.1.2 Console Arguments

Instead of using the parameter file, some arguments can also be set by console arguments. The console arguments have the highest priority and are overwriting the arguments of the `param.dat` file. The options are:

- -name <c>: name
- -T <double> : T
- -P <double> : P
- -M <int> : Molecule Name

- -path <c> : pathToData
- -pathK <c> : pathToK
- -numin <double> : numin
- -numax <double> : numax
- -dnu <double> : dnu
- -cutM <int> : cutMode
- -cut <double> : cut
- -dR <int> : doResampling
- -nC <int> : nC
- -dT <int> : doTRansmission
- -nTr <int> : nTr
- -dTr <double> : dTr
- -dSF <int> : doStoreFullK
- -dSS <int> : doStoreSK
- -nbins <int> : nbins
- -kmin <double> : kmin
- -dev <int> : Device number (For multiple GPU systems)
- -q <double> : qalphaL
- -gammaF <double> : gammaF
- -Mean <int> : doMean
- -tuning <int> : doTuning

where <c>is a string, <double> a floating point number, and <int>an integer.

### 3.1.3 Code parameters

The file define.h contains the physical parameters and some code parameters. After changing some entry here, the code needs to be recompiled. The code parameters are:

- def_TOL: Tolerance parameter in the Voigt function. See Algorithm 916
- def_nthmax: Maximum number of threads per kernel launch. In 2.0 GPUs it can not be larger than 32768.
- def_nlmax: Maximum number of molecular lines per kernel launch. Setting a lower number prevents from a time-out on Desktop machines.
- def_maxlines: Maximum number of lines stored on the GPU.
- def_maxfiles: Maximum number of files per molecule.
- def_NmaxSample: Maximum Number of resample coefficients for K(y)

When using a Desktop GPU running an x session, the runtime of a single kernel launch can be limited to a few seconds. Choosing smaller values for nlmax and nthmax splits the kernel into smaller parts. But it makes the code a bit slower.

CHAPTER

# FOUR

# OPTIONS

## 4.1 Line Profiles

HELIOS-K supports four different line profiles which can be set by the `profile` parameter.
The supported profiles are:

- 1: Voigt
- 2: Lorentz

$$f_L(\nu) = \frac{1}{\pi} \frac{\gamma_L}{(\nu - \nu_0)^2 + \gamma_L^2} \tag{4.1}$$

- 3: Doppler

$$f_G(\nu) = \sqrt{\frac{ln(2)}{\pi}} \frac{1}{\alpha_D} \exp\left(-\frac{(\nu - \nu_0)^2 ln(2)}{\alpha_D^2}\right) \tag{4.2}$$

- 4: Binned Gaussian integrated cross section

$$f_{BG} = \frac{1}{\Delta\nu} \int_{\nu-\nu/2}^{\nu+\nu/2} f_G(\nu) d\nu = \frac{1}{2\Delta\nu} \left[erf(\chi^+) - erf(\chi^-)\right] \tag{4.3}$$

[Yurchenko et al. 2018: (ExoCross : a general program for generating spectra from molecular line lists)]

with the Doppler half-width:

$$\alpha_D = \frac{\nu}{c} \sqrt{\frac{2ln(2)k_B T}{m}} \tag{4.4}$$

and the Lorentz half-width for Hitran like data:

$$\gamma_L = \frac{A}{4\pi c} + \left(\frac{T}{T_{ref}}\right)^{-n} \left[\frac{\alpha_{air}(P - P_{self})}{P_{ref}} + \frac{\alpha_{self} P_{self}}{P_{ref}}\right] \tag{4.5}$$

or the Lorentz half-width for ExoMol like data:

$$\gamma_L = \frac{A}{4\pi c} + \left(\frac{T_{ref}}{T}\right)^{n} \cdot \left(\frac{P}{P_{ref}}\right) \tag{4.6}$$

or the Lorentz half-width for Atomic data:

$$\gamma_L = \frac{\Gamma_{nat}}{4\pi c} + \left(\frac{T_{ref}}{T}\right)^{n} \cdot \left(\frac{P}{P_{ref}}\right) \tag{4.7}$$

In Fig. 4.1 is shown an example with four different line profiles.

Relevant parameters for this example:

- doStoreFullK = 1
- profile = 1 or 2 or 3 or 4

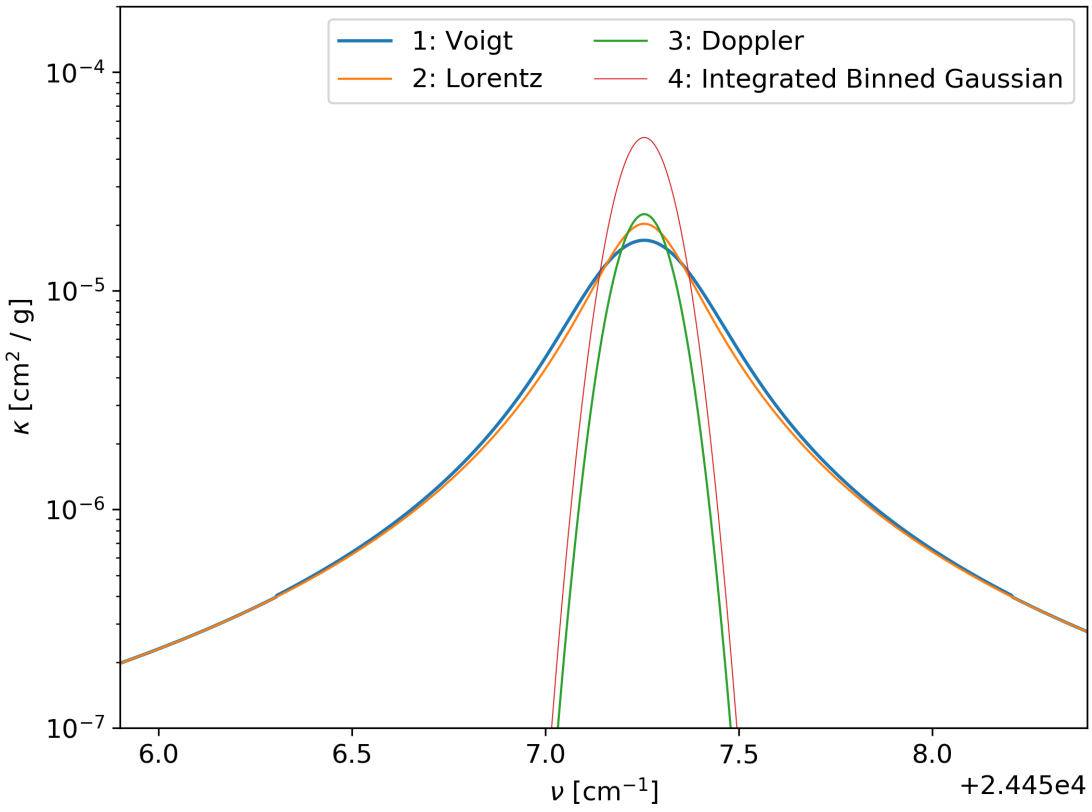



Fig. 4.1: Example with four different line profiles

## 4.2 CO2 sub-Lorentzian line wings

HELIOS-K supports sub-Lorentzian profiles according to the $\chi$ factors from Perrin and Hartmann 1998. The $\chi$ factors are specified in the files `chiPH89__CO2-CO2.dat` and `chiPH89__CO2-N2.dat`. In these files, the affected wavenumber range can be set. The `param.dat` file contains the option `subLorentzianfile`, where the file name of the used $\chi$ factors file can be set. If a filename is set there, the sub-Lorentzian line wings are enabled.

In principle, the $\chi$ factors can be applied also to other molecules than CO2, but this is not recommended to do.

M.Y.Perrin and J.M.Hartmann, 1988 (TEMPERATURE-DEPENDENT MEASUREMENTS AND MODELING OF ABSORPTION BY CO2-N2 MIXTURES IN THE FAR LINE-WINGS OF THE 4.3 # m CO2 BAND).

In Fig. 4.2 is shown an example of the sub-Lorentzian profile for CO2.

Relevant parameters for full Voigt:

- doStoreFullK = 1
- cutMode = 0
- cut = 25.0
- profile = 1
- subLorentzianfile = -

Relevant parameters for sub-Lorentzian Voigt profile in specific wavenumber band:

- doStoreFullK = 1
- cutMode = 0
- cut = 25.0
- profile = 1
- subLorentzianfile = chiPH89__CO2-CO2.dat
- **in chiPH89__CO2-CO2.dat:**
  - range in cm^-1
  - 2100 2500

Relevant parameters for sub-Lorentzian Voigt profile in the full wavenumber range:

- doStoreFullK = 1
- cutMode = 0
- cut = 25.0
- profile = 1
- subLorentzianfile = chiPH89__CO2-CO2.dat
- **in chiPH89__CO2-CO2.dat:**
  - range in cm^-1
  - 0 10000

## 4.3 Cut the line wings

The line wings can be cut by the `cutMode` and `cut` parameters. Reasons for cutting the line wings can either be that the line wings would be overestimated, or just to save computing time. Line wings or cut at the specified distance from their central peak location. Choosing a broader cutting lenght can increase the computational time. In Fig. 4.3 is shown an example with three different cutting lenghts.

Relevant parameters for this example:

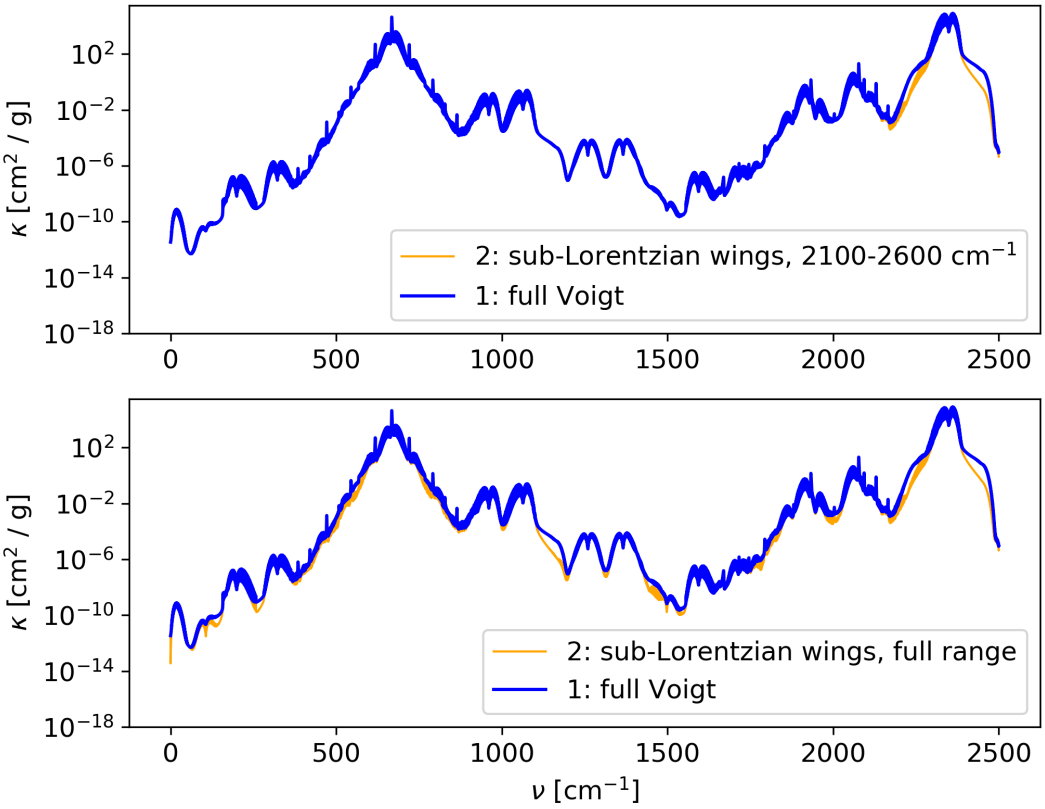



Fig. 4.2: CO2 sub-Lorentzian wings with $\chi$ factors for Perrin and Hartmann 1998. Top panel,: $\chi$ factors applied to the band 2100-2600 cm^-1. Bottom panel: $\chi$ factors applied to the full wavenumber range. CO2, Hitran 2016, T = 250 K, P = 2 atm.

- doStoreFullK = 1
- cutMode = 0
- cut = 25, 100 or 0

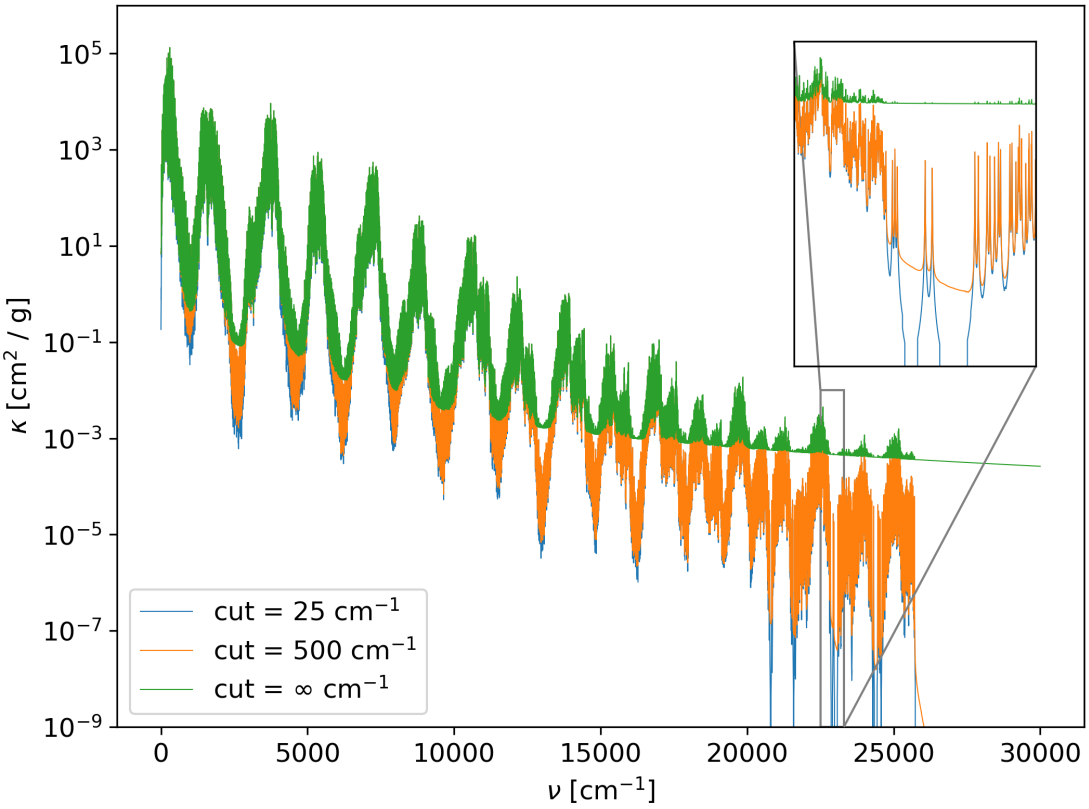



Fig. 4.3: Three cutting lengths of the line wings. An infinity cuttoff lengths can be set by *cut = 0*.

## 4.4 Removing the Plinth

When using cutted line wing profiles in combination with a continuum and far wing background cross section, it can be necessary to remove the plinth of each transition line, since this can be already included in the backround.

See for example:
Ptashnik, Igor & McPheat, Robert & Shine, Keith & Smith, Kevin & Williams, Robert. (2012). Water vapour foreign-continuum absorption in near-infrared windows from laboratory measurements. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences. 370. 2557-77. 10.1098/rsta.2011.0218.
or
S.A. Clough, F.X. Kneizys, R.W. Davies,

Line shape and the water vapor continuum, Atmospheric Research, Volume 23, Issues 3–4, 1989, Pages 229-241, ISSN 0169-8095, https://doi.org/10.1016/0169-8095(89)90020-3.

Removing the plinth from the opacities can be done with the *removePlinth* option in the *param.dat* file. The hight of the plinth is defines as the value of the line profile at the cutting length. And example is shown in Fig. 4.4.

Relevant parameters for this example:

- doStoreFullK = 1
- removePlinth = 1

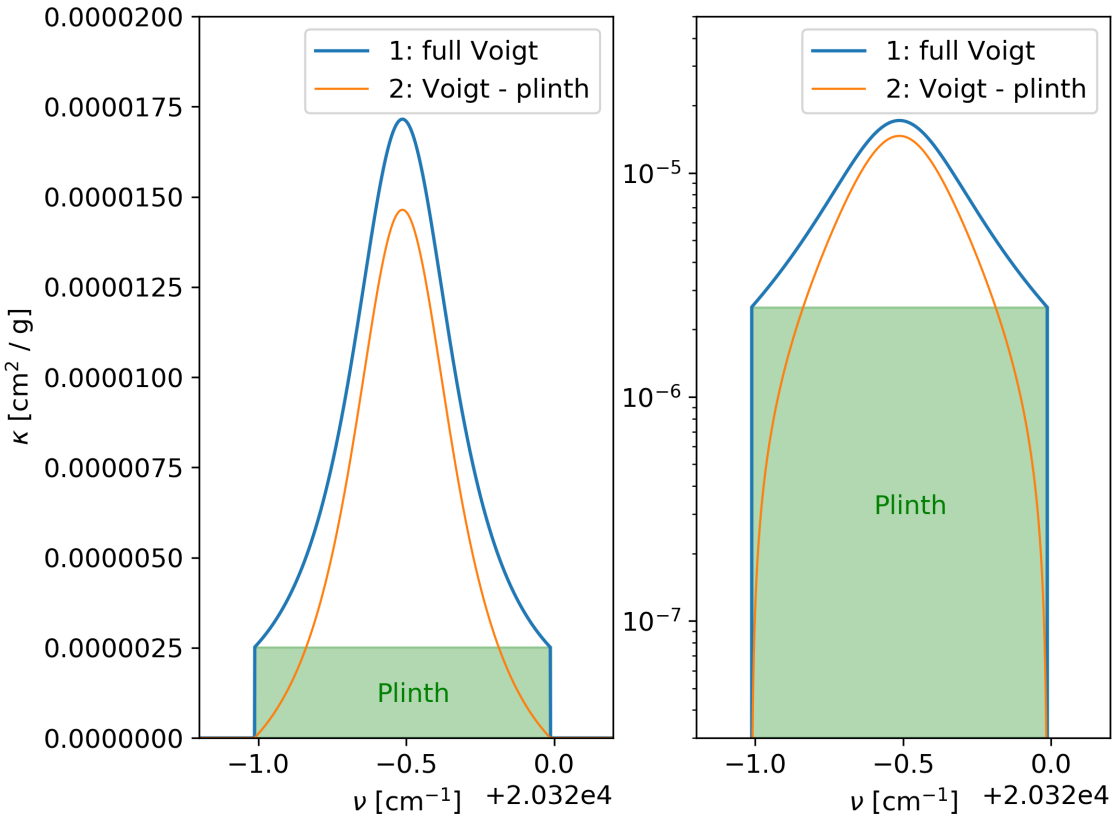



Fig. 4.4: The full line profile (blue) is cutted at the a certain cutting length. The value at the cutting lenght, defines the value of the plinth, which can be removed from the opacity function (orange). Both panels show the same data, the left panel is using linear scales, the right panel is using log scale in the y-axis.

## 4.5 Bins

Bins devide the total range of wavenumbers into equal size or not-equal sized ranges. Inside the bins, the values of the opacity function can be compacted into a more lightweight format, in order to same memory space. An example is shown in Fig. 4.5, where the wavenumber range from 0 30000 cm^-1 is divided into fife bins. Inside of each bin, the opacity function is sorted from small to large, and mapped to a new parameter y, rangin from 0 to 1. When the opcity function has empty parts inside or is not ranging to the full bin width, then the sorted opacity function is zero (or kmin, if used) on the lower part.

Relevant parameters for this example:

- doResampling = 0
- doStoreFullK = 1
- doStoreSK = 2
- nbins = 5

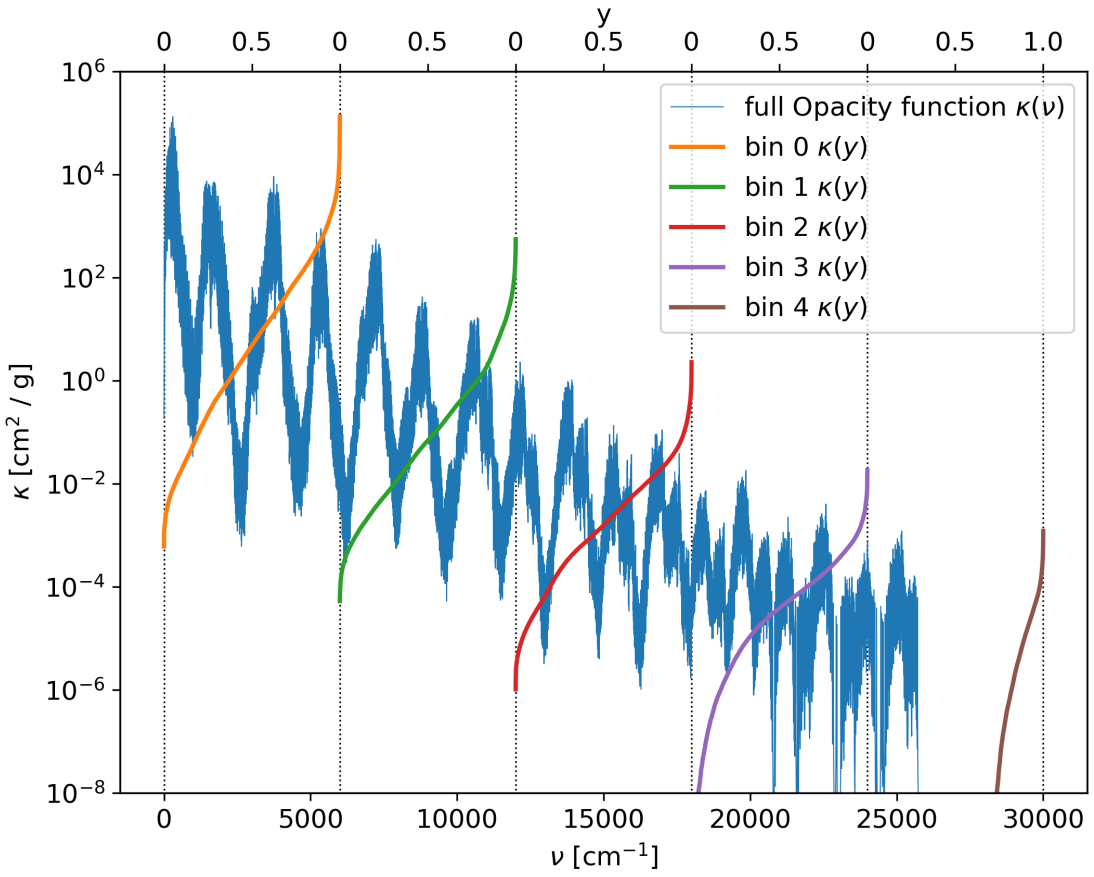



Fig. 4.5: The opacity function is divided into fife bins. Inside each bin, the opacity function is sorted and mapped to a new paramter y, ranging from 0 to 1.

### 4.5.1 Using a binsfile

When a `binsFile` name is given in the `param.dat` file, then this file is used to generate the boundaries of the bins. The bins do not have to be equal sized. | Note that this option does not support the doResampling and doTransmission options. | The binsfile must contain line by line the boundaries of the bins in cm^-1.

An example of a `binsFile` is given below, and the result is shown in Fig. 4.6.

bins.dat:

```
2000
6000
12000
15000
25000
30000
```

Relevant parameters for this example:

- doResampling = 0
- doStoreFullK = 1
- doStoreSK = 2
- binsFile = bins.dat

### 4.5.2 The output edges option

Instead of writing the -per bin sorted- opacity function with the full resolution in wavenumbers, it is possible to print only averaged positions within the bins. For doing that, the `outputedgesFile` option in the `param.dat` file can be used. In that file, the edges of the averaged regions within a bin can be specified. The output possition of the sorted opacity function is then exaclty in between of the edges. The edges must have values between 0 and 1.

An example of a `outputedgesFile` is given below, and the result is shown in Fig. 4.7. Note that the figure is plotted in log scale, therefore the averaged points can appear to have too high values.

edges.dat:

```
0.0
0.1
0.45
0.78
1.0
```

Relevant parameters for this example:

- doResampling = 0
- doStoreFullK = 1
- doStoreSK = 2
- nbins = 5

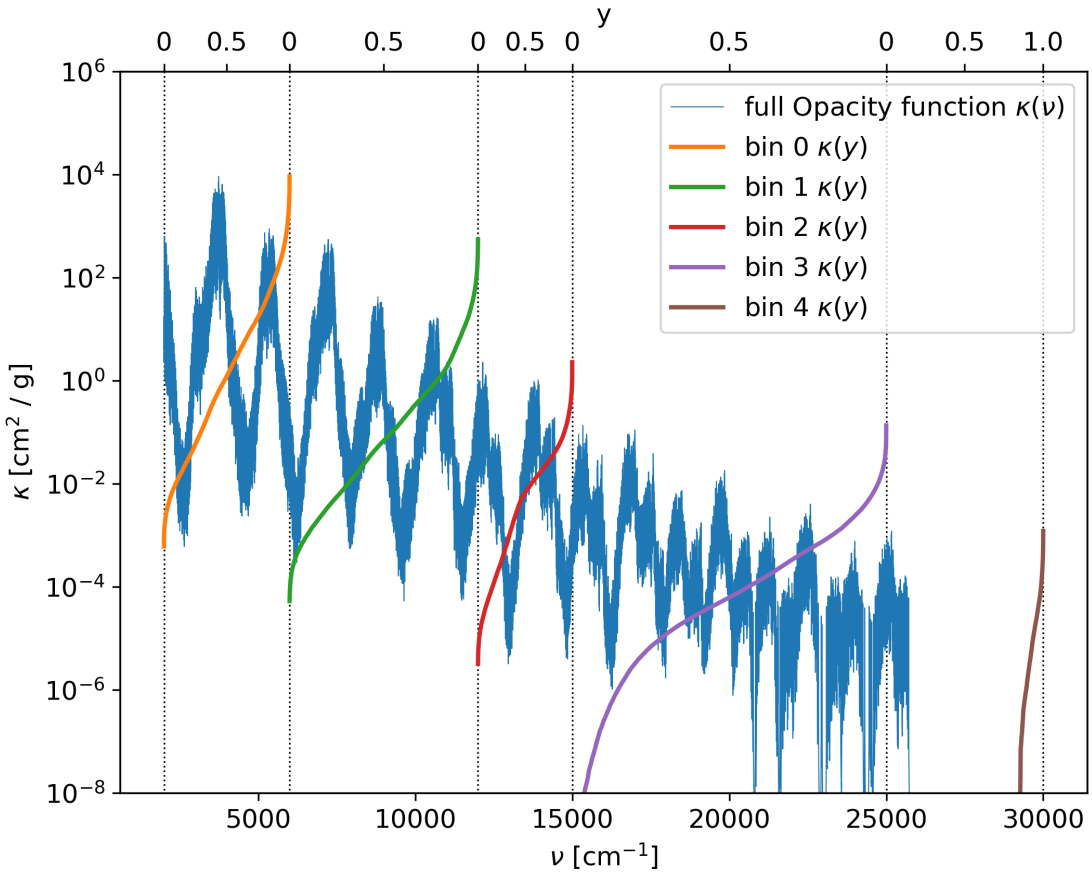



Fig. 4.6: The opacity function is divided into fife bins, according to the entries of the binsfile. Inside each bin, the opacity function is sorted and mapped to a new paramter y, ranging from 0 to 1.

- OutputEdgesFile = edges.dat

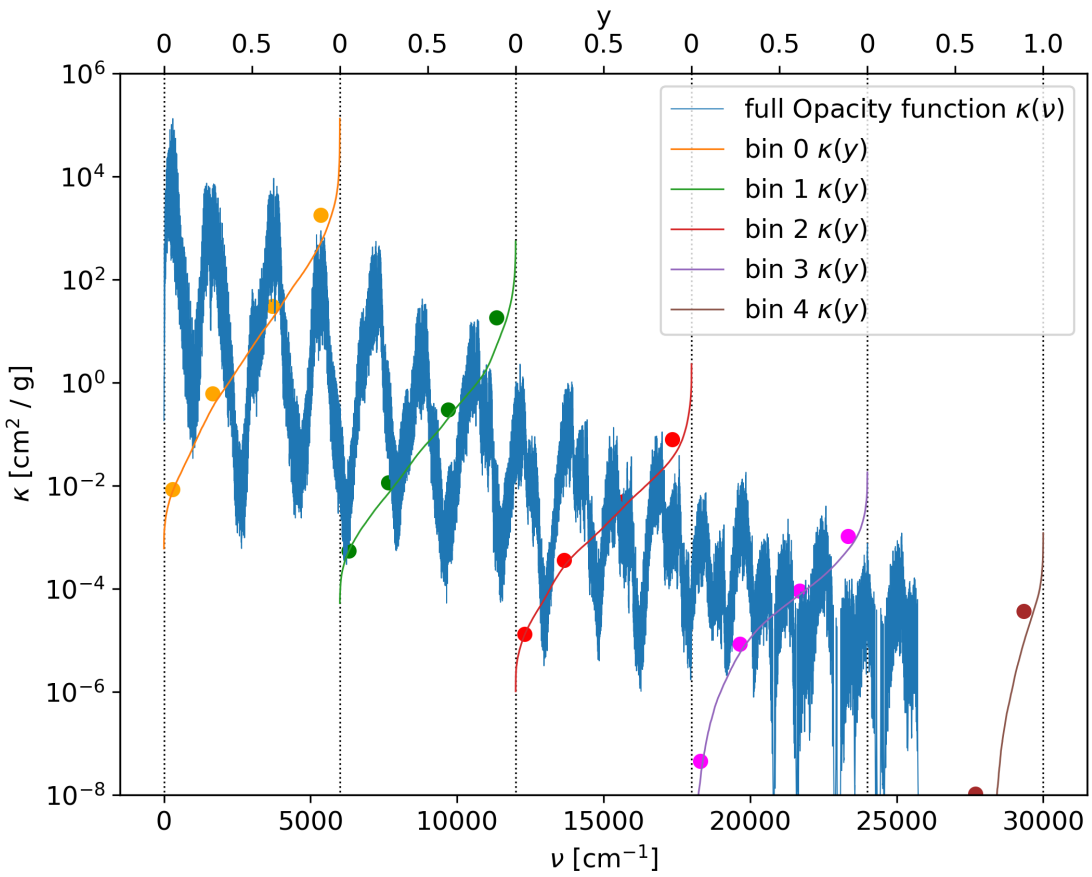



Fig. 4.7: The opacity function is divided into fife bins. Inside each bin, the opacity function is sorted and mapped to a new paramter y, ranging from 0 to 1. The sorted opacity function is averaged within the given edges.

## 4.6 Resampling and k-coefficients

With the `doResampling` option in the `param.dat` file, the per bin sorted opacity function can be resampled with a Chebyshev polynomial. The number of Chebyshev can also be set in the `param.dat` file with the `nC` option. By using the resampling method, a lot of storage space can be saved. However, the exact location of individual transition lines is not contained in the resampled data. The Chebyshev coefficients are reported the `Out<name>_cbin.dat` files.

An example of a resampled bin is shown in Fig. 4.8.

Relevant parameters for this example:

- doResampling = 1
- nC = 20
- doStoreSK = 2

- nbins = 1

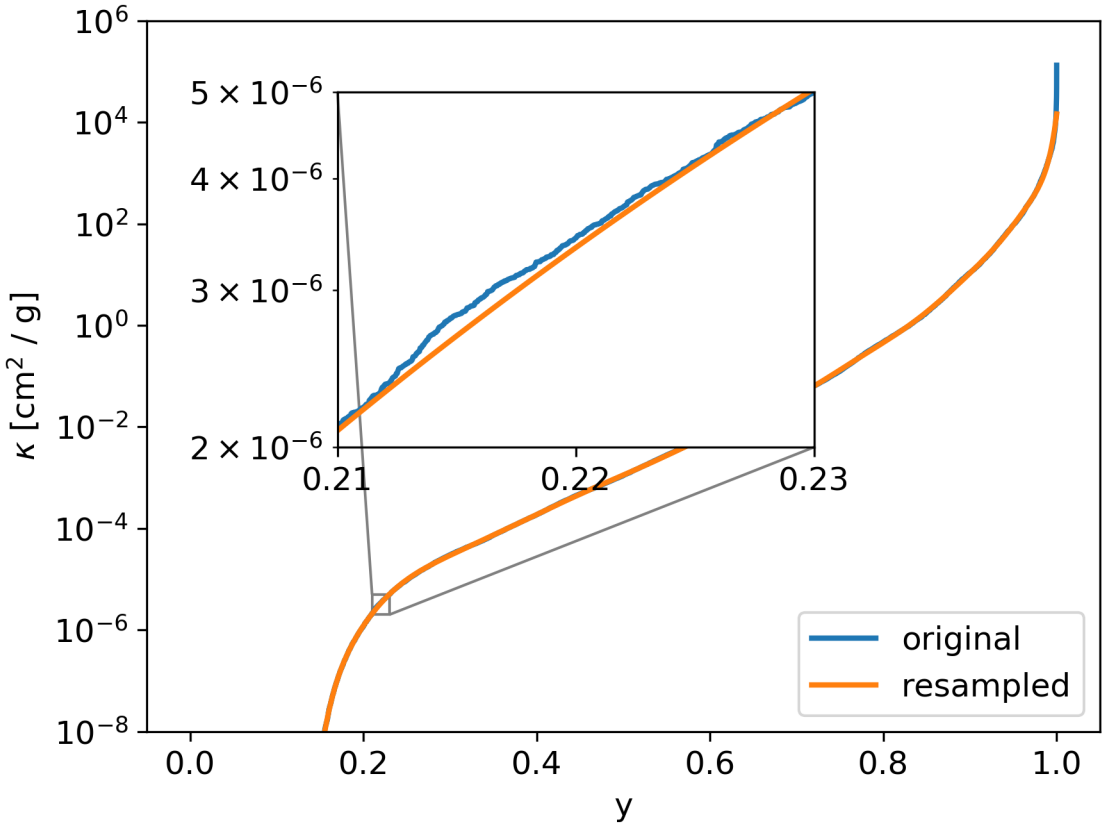



Fig. 4.8: The per bin sorted opacity function is resampled with a Chebyshev polynomial, in order so save storage space.

### 4.6.1 Bins with empty parts

Some bins can contain an empty parts, where the opacity function have no transition lines. This empty parts causes the sorted opacity parts to have a sharp edge between the empty parts and the rest, as shown in Fig. 4.9. Using the Chebyshev polynomial to resample over these sharp edge would introduce oscillations into the resampled opacity functions. In order to avoid these oscillations, we resample only the part of the sorted opacity function, which is not 0 or kmin. That leads to better results, but when reconstructing the polynomial from the Chebyshev coefficients, then the empty parts must be added again. How this can be done is shown next.

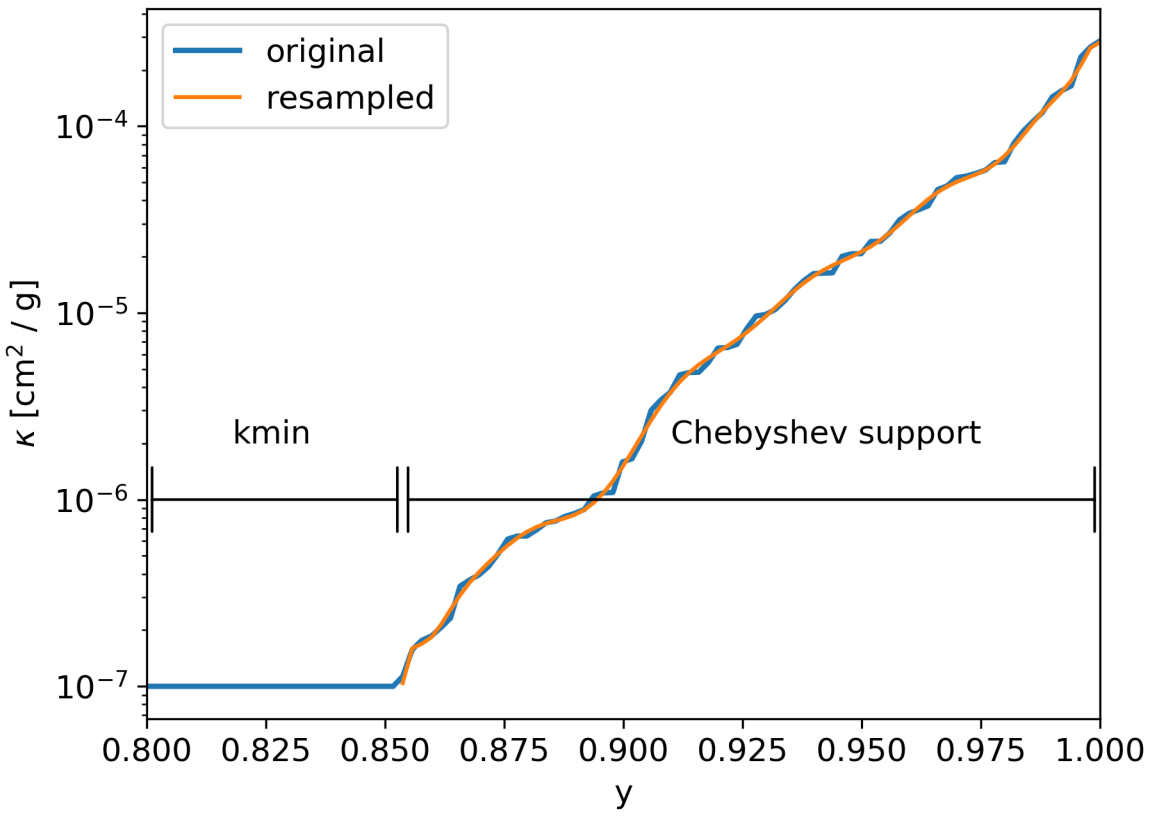



Fig. 4.9: Resampling a bin with an empty part. To avoid oscillations in the resampled Polynoial, only values > kmin are considered.

### 4.6.2 Using the Chebyshev coefficients

The resampled opacity function can be reconstruced with the following code: (See also the code recon_bin.py in the tools directory)

```
import numpy as np
from numpy.polynomial.chebyshev import chebval

#change here the name of the file
data_c = np.loadtxt('Out_name_cbin.dat')

#change here the bin index and the bin size:
binIndex = 0
binSize = 300

x = np.linspace(0, 1.0, num=binSize, endpoint=True)

#extract Chebyshev coefficients
c = data_c[binIndex,2:]
#extract starting point in x of opacity function
xs = data_c[binIndex,1]

#rescale x to the standard Chebychev polynomial range [-1:1]
x1 = x * 2.0 - 1.0
k_res = chebval(x1,c,tensor=False)
x2 = x * (1.0 - xs) + xs

#result is in k_res for x values in x2
k_res = np.exp(k_res)
```

## 4.7 The P file option

When a `PFile` name is given in the `param.dat` file, then this file is used to read multiple values for P. This option is useful to speed up the performance, because multiple reads from the data files can be avoided. Too many entries in the Pfile can lead to a memory shortage.

For example:

```
1.0
10.0
100.0
```

## 4.8 The Species file option

This option must be used to calculate opacities for gas mixtures, containing multiple species. The File contains in the two columns the species name, and the number fraction.

For example:

```
01_hit16    0.9
05_hit16    0.1
```

This example will produce an opacitiy with 90% H2O and 10% CO.

## 4.9 The transmission function

With the `doTransmission` option in the `param.dat` file, the transmissions function $\tau$

$$\tau = \int_0^\infty e^{-\kappa\tilde{m}} d\nu = \int_0^1 e^{-\kappa\tilde{m}} dy$$

can be calculated for a set of column masses $\tilde{m}$, where $\tilde{m}$ is set according to:

$$\tilde{m}_i = e^{((i-nTr/2)\cdot dTr)}.$$

The parameters `nTr` and `dTr` can be set in the `param.dat` file. The transmission function is stored in the file `Out<name>_tr.dat`,

An example of the transmission function for fife bins is shown in Fig. 4.10.

Relevant parameters for this example:

- doStoreFullK = 1
- doTransmission = 2
- nbins = 5
- nTr = 1000
- dTr = 0.05

## 4.10 Planck and Rosseland Means

With the `doMean` option in the `param.dat` file, the Plank and Rosseland means, $\kappa_P$ and $\kappa_R$ respectively:

$$\kappa_P = \frac{\int_0^\infty \kappa B_\nu(T) d\nu}{\int_0^\infty B_\nu(T) d\nu}, \tag{4.8}$$

and

$$\kappa_R = \left( \frac{\int_0^\infty \kappa^{-1} \frac{\partial B_\nu(T)}{\partial T} d\nu}{\int_0^\infty \frac{\partial B_\nu(T)}{\partial T} d\nu} \right)^{-1} \tag{4.9}$$

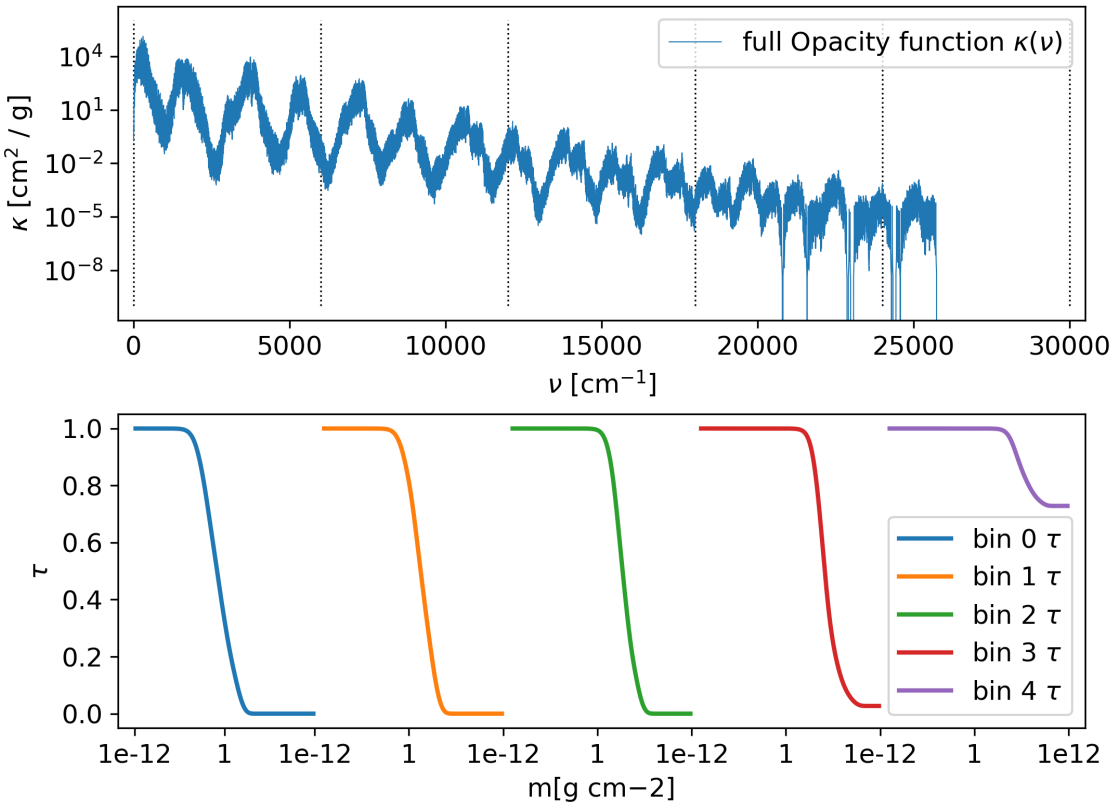



Fig. 4.10: Transmission function

can be calculated, where the infinity integral is truncated to the specified wavenumber limits, and $d\nu$ is equal to `dnu` set in the `param.dat` file.

Note that the denominators in $\kappa_P$ and $\kappa_R$ can be computed analytically as

$$\int_0^\infty B_\nu(T)d\nu = \frac{\sigma T^4}{\pi} \tag{4.10}$$

and

$$\int_0^\infty \frac{\partial(B_\nu(T))}{\partial(T)}d\nu = \frac{4\sigma T^3}{\pi}. \tag{4.11}$$

Therefor, it is usefull to compare those analytic results to the numerical integration

$$\int_0^\infty B_\nu(T)d\nu = \sum_i B_\nu(T)dnu \tag{4.12}$$

and

$$\int_0^\infty \frac{\partial(B_\nu(T))}{\partial(T)}d\nu = \sum_i \frac{\partial(B_\nu(T))}{\partial(T)}dnu \tag{4.13}$$

The results of the Planck and Resseland means are stored in the file `Out<name>_mean.dat`, together with the analytic and numerical expressions (4.10) to (4.13). If the numerical expressions deviate strongly from the analytical expression, then it is a hint that the wavenumber resolution is not set fine enough.

An example of the Planck and Rosseland means is shown in Fig. 4.11.

Relevant parameters for this example:

- doStoreFullK = 1
- doMean = 1

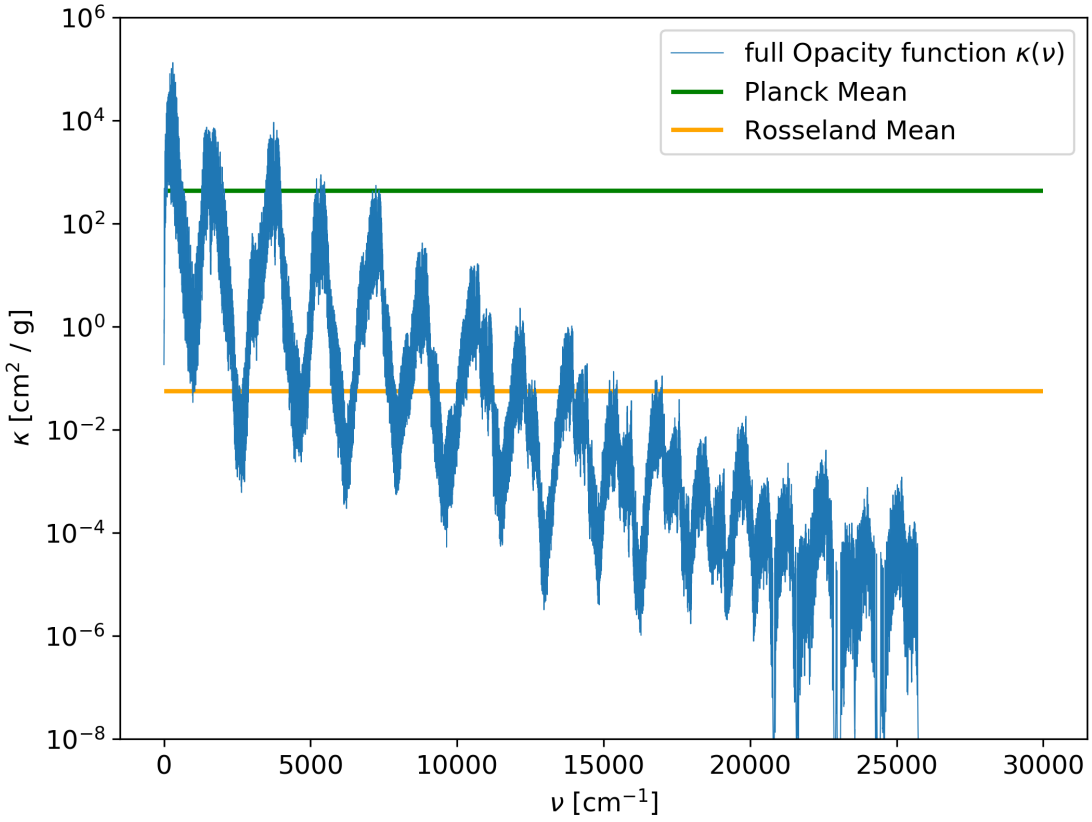



Fig. 4.11: Planck and Rosseland means

CHAPTER

FIVE

# OUTPUT FILES

## 5.1 Output Files

Different Output files are written, depending on the values in the `param.dat` file

### 5.1.1 `Info_<name>.dat`

Contains all used parameters, and timing information

### 5.1.2 `Out_<name>.dat`

It contains nu and K(nu), where nu are the wavenumbers and K(nu) is the full opacity function. When the `PFile` option is used, then the files contain also the values of T and P.

This files are written by using the option:

- doStoreFullK = 1

### 5.1.3 `Out_<name>.bin`

Binary output file, it contains K(nu), where nu are the wavenumbers and K(nu) is the full opacity function. The wavenumbers are not contained in the output files, and have to be calculated manually by using the wavenumber resolution, and file line index. The opacities are stored in a single precision floating point binary format.

This files are written by using the option:

- doStoreFullK = 2

### 5.1.4 Convert Out_<name>.dat to Out_<name>.bin files

Output files in the text format `*.dat` can be converted into binary `*.bin` files with the script `DATtoBIN.py` in the `/tools` directory.

Use for example `python3 DATtoBIN.py -n Out_i` to convert the file `Out_i.dat` to `Out_i.bin`.

### 5.1.5 Out_<name>_bin.dat

It contains the values of y and K(y) per bin. y goes from 0 to 1. K(y) is the per bin sorted opacity function. The bins are separated by two blank lines, starting with the bin with the lowest wavenumber and ending with the bin with the highest wavenumber.

When `doResampling` is set to one, then this file contains the sorted opacity functions, recomputed from the Chebyshev coefficients. When the `PFile` option is used, then the files contain also the values of T, P and point index.

When the `OutputEdgesFile` option is used, then the file contains not all points in y, but the averaged values between the edges, given in the `OutputEdgesFile`.

When `doStoreSK` is set to 2, then the bins are stored in different files with names `Out_<name>_bin< bin index>.dat` .

### 5.1.6 Out_<name>_cbin.dat

It contains the Chebyshev coefficients of the per bins sorted natural logarithm of the opacity functions in the format

`kmin_i ystart_i C0_i C1_i ... C(nC - 1)_i`, where i refers to the bin index, and `C` are the Chebyshev coefficients.

`kmin` is the minimal value of `K(y)`, used e.g. in holes in the opacity funtion.

`ystart` is the position in y when the value of`K(y)` starts to be larger than `kmin`.

`K(y)` can be recomputed as

`K(y) = sum_(0 <= j < nC) (C[j] * T[j](yy))`,

where `T(y)` are the Chebyshev polynomials and `yy = (2.0 * y - 1.0 - ystart) / (1.0 - ystart)`, for y in the range `[ystart, 1]`. The bins are separated with a blank line, starting with the bin with the lowest wavenumber and ending with the bin with the highest wavenumber. When the `PFile` option is used, then the files contains also the values of T and P. When `doResampling` is set to 2, then the bins are stored in different files with names `Out_<name>_cbin< bin index>.dat` .

The following python script can be used to reconstruct the per bin sorted opacity function from the Chebyshev coefficients:

```
import numpy as np
from numpy.polynomial.chebyshev import chebval

#change here the name of the file
data_c = np.loadtxt('Out_name_cbin.dat')

#change here the bin index and the bin size:
binIndex = 0
binSize = 300

#extract Chebyshev coefficients
```

(continues on next page)

(continued from previous page)

```
c = data_c[binIndex,2:]
#extract starting point in x of opacity function
xs = data_c[binIndex,1]

#rescale x to the standard Chebychev polynomial range [-1:1]
x1 = x * 2.0 - 1.0
k_res = chebval(x1,c,tensor=False)
x2 = x * (1.0 - xs) + xs

#result is in k_res for x values in x2
k_res = np.exp(k_res)
```

### 5.1.7 Out_<name>_tr.dat

It contains m and T. m is the column mass, m_i = exp((i - nTr/2) * dTr) T is the Transmission function Int_0^1 exp(-K(y)m) dy When the PFile is used then the files contains also the values of T, P and point index. When doTransmission is set to 2, then the bins are stored in different files with names `Out_<name>_tr< bin index>.dat`

### 5.1.8 Out_<name>_mean.dat

When the argument doMean is set to one, this file contains the Planck and Rosseland means. They are computed over the entire range in wavenumbers from numin to numax with spacing dnu.

The first line is the Planck mean: kappa_P = Int_0^infty (kappa * B_nu * dnu) / Int_0^infty (B_nu * dnu).

The second line is the Rosseland mean: kappa_R = (Int_0^infty (kappa^-1 * del(B_nu)/del(T) * dnu) / Int_0^infty ( del(B)/del(T)_nu * dnu))^-1

The third line is the numerical integral Int_0^infty (B_nu * dnu)

The fourth line is the analytic integral Int_0^infty (B_nu * dnu) = sigma * T^4 / pi

The fifth line is the numerical integral Int_0^infty ( del(B)/del(T)_nu * dnu)

The sixth line is the analytic integral Int_0^infty ( del(B)/del(T)_nu * dnu) = 4 * sigma * T^3 / pi

The value of the numerical integrals should converge to the analytic expressions for high resolutions dnu, numin -> 0 and numax -> infinity.

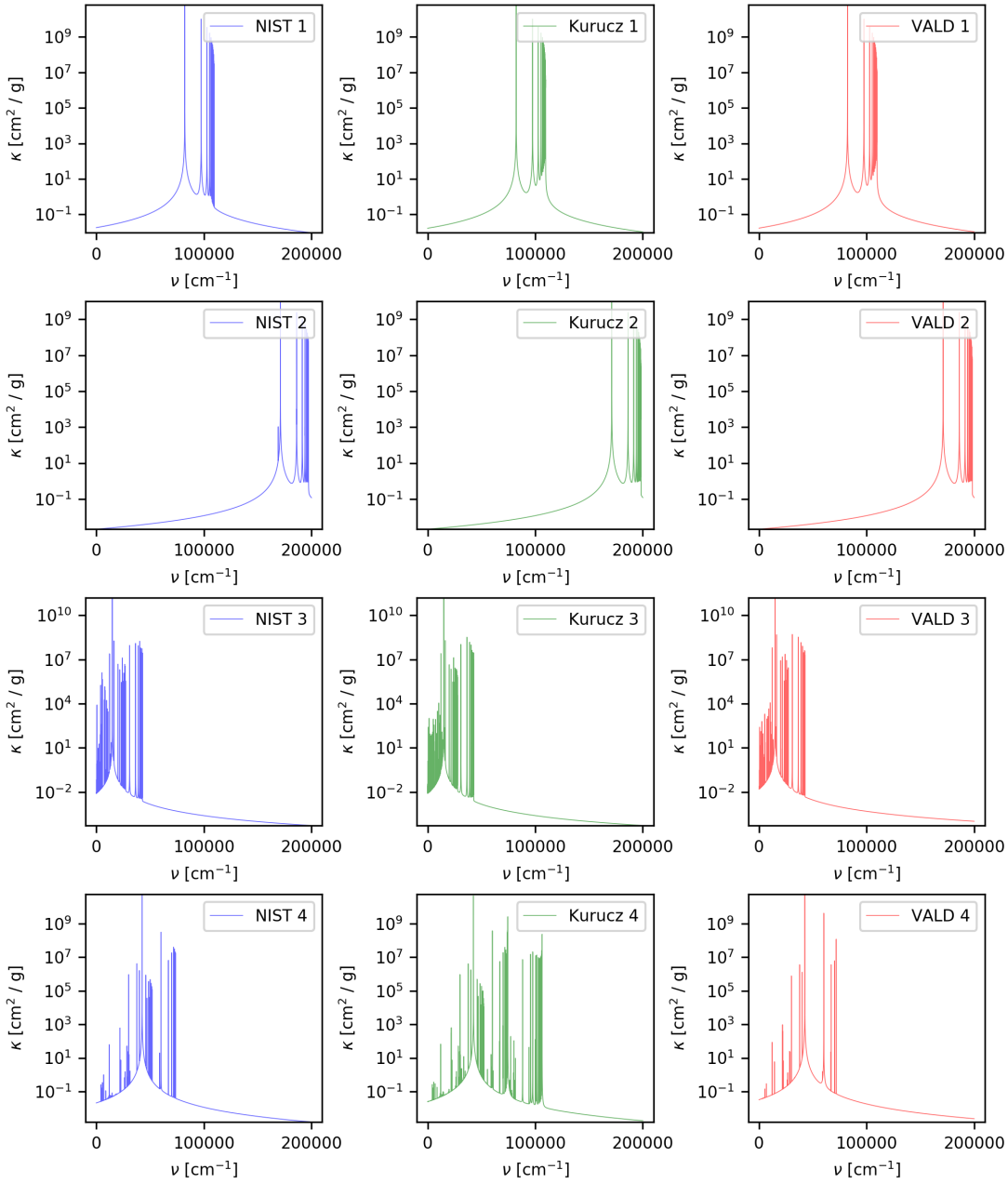
CHAPTER

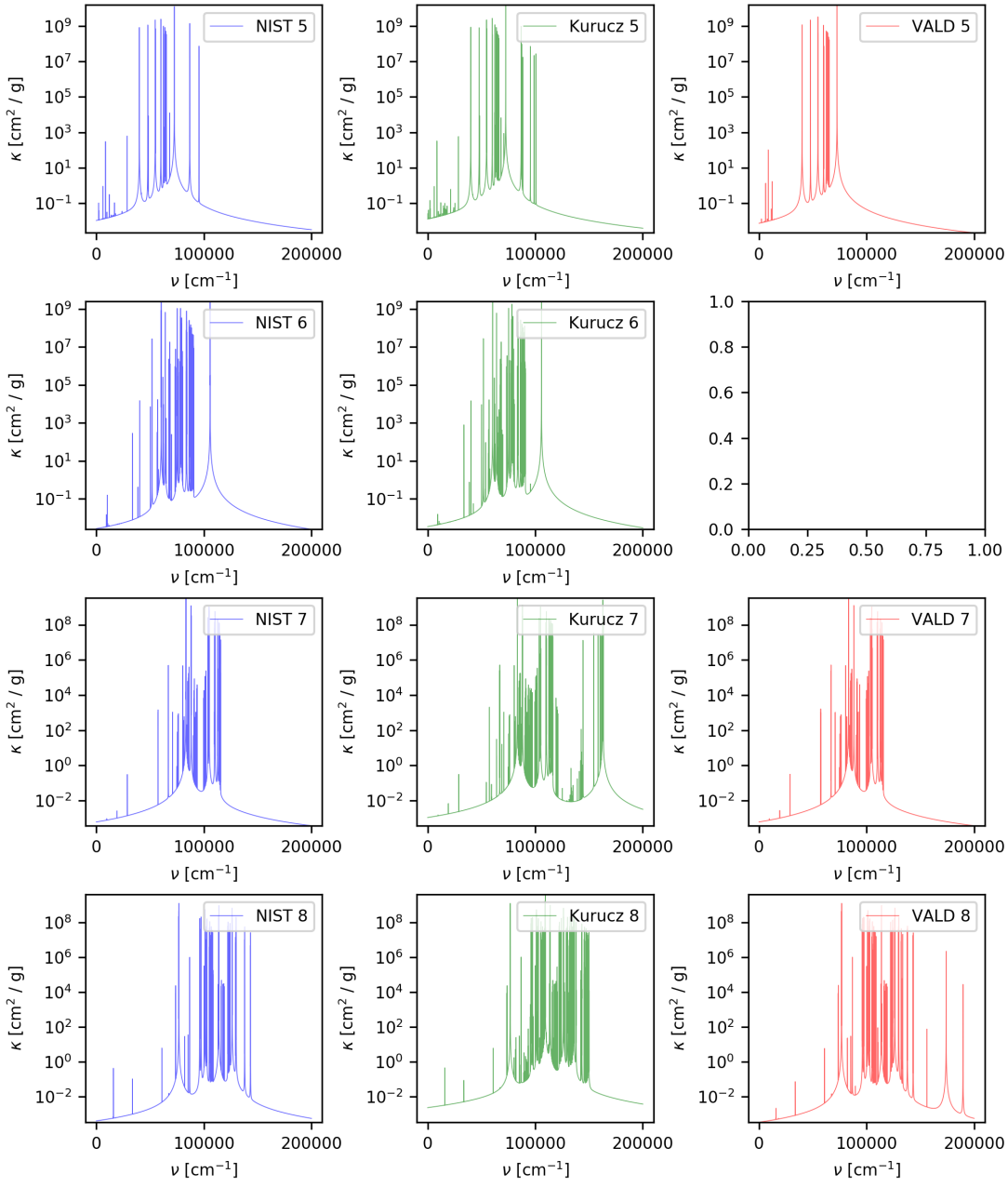
SIX

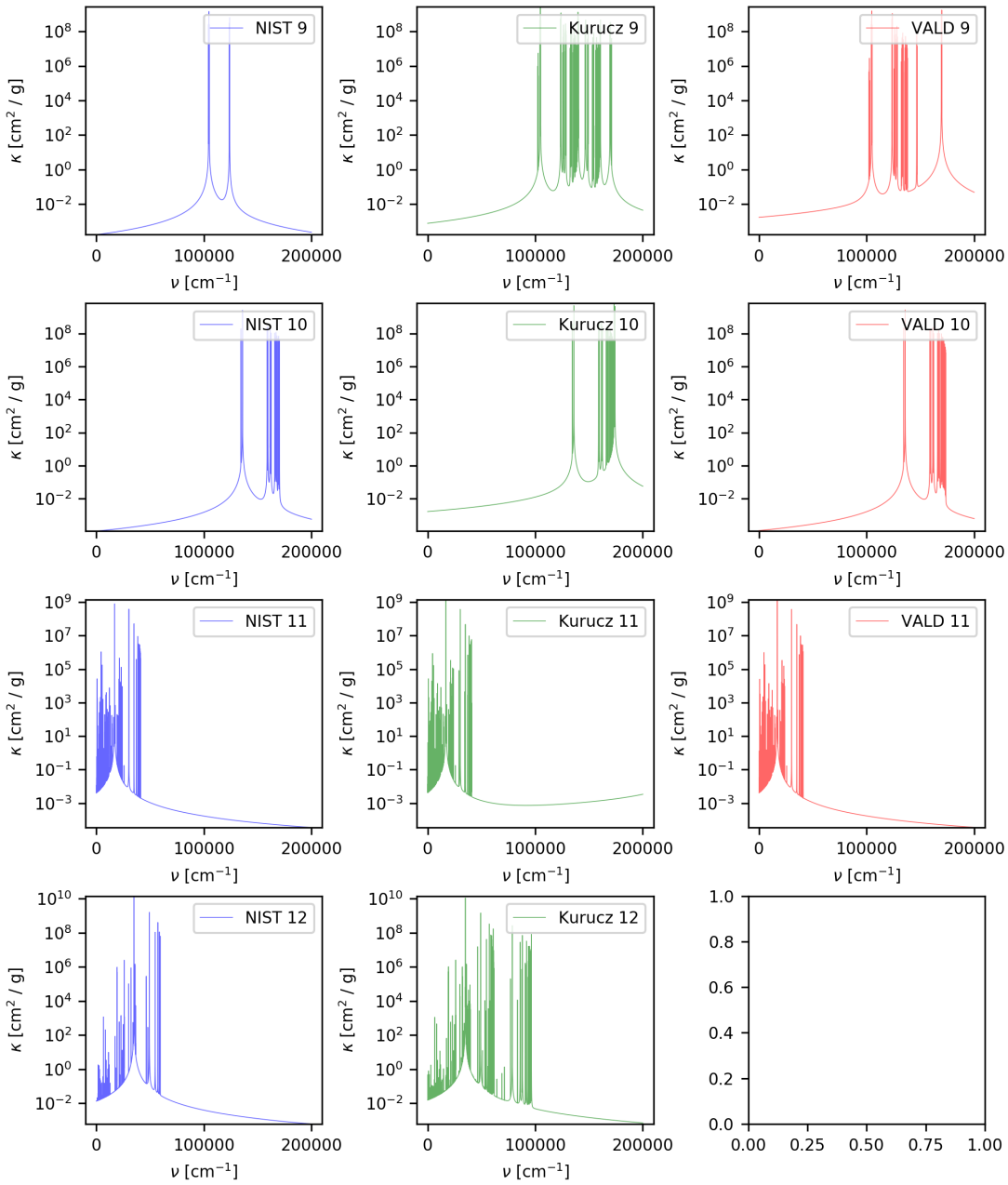
# ATOMIC OPACITIES PLOTS

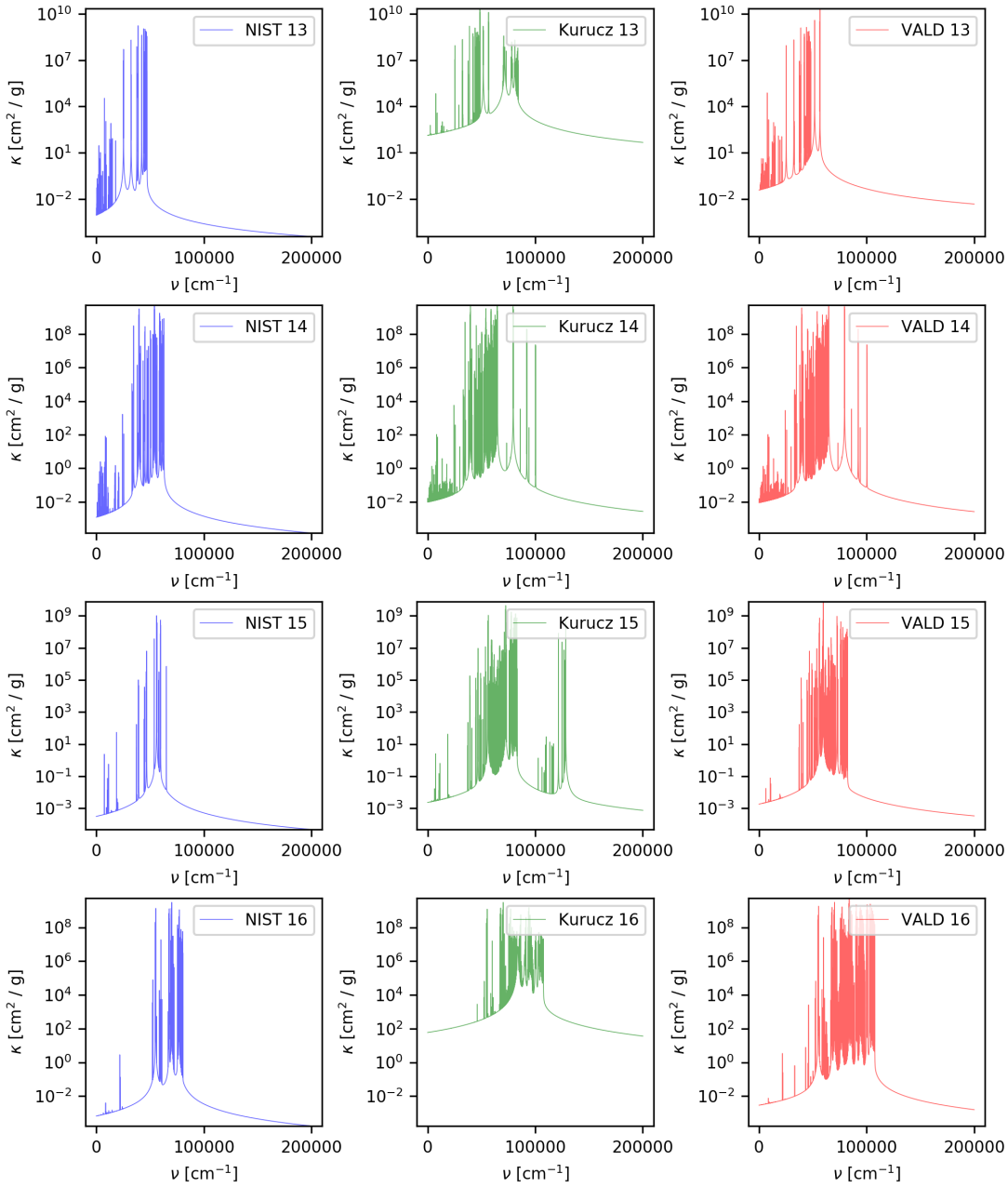
## 6.1 Atomic opacities

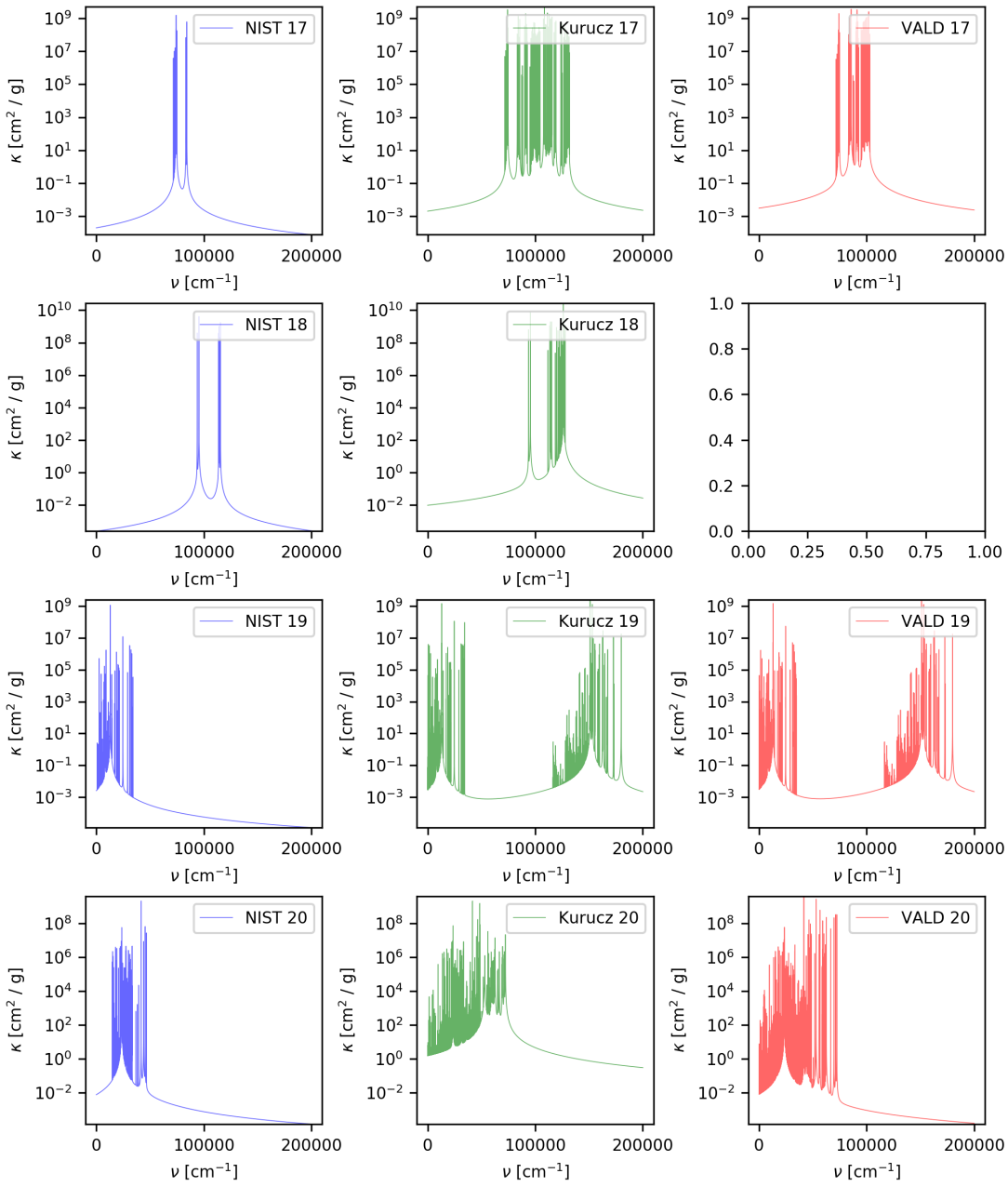
In this section we show all available atomic opacities from the NIST and Kurucz database. We use T = 3000 K and P = 1 bar.

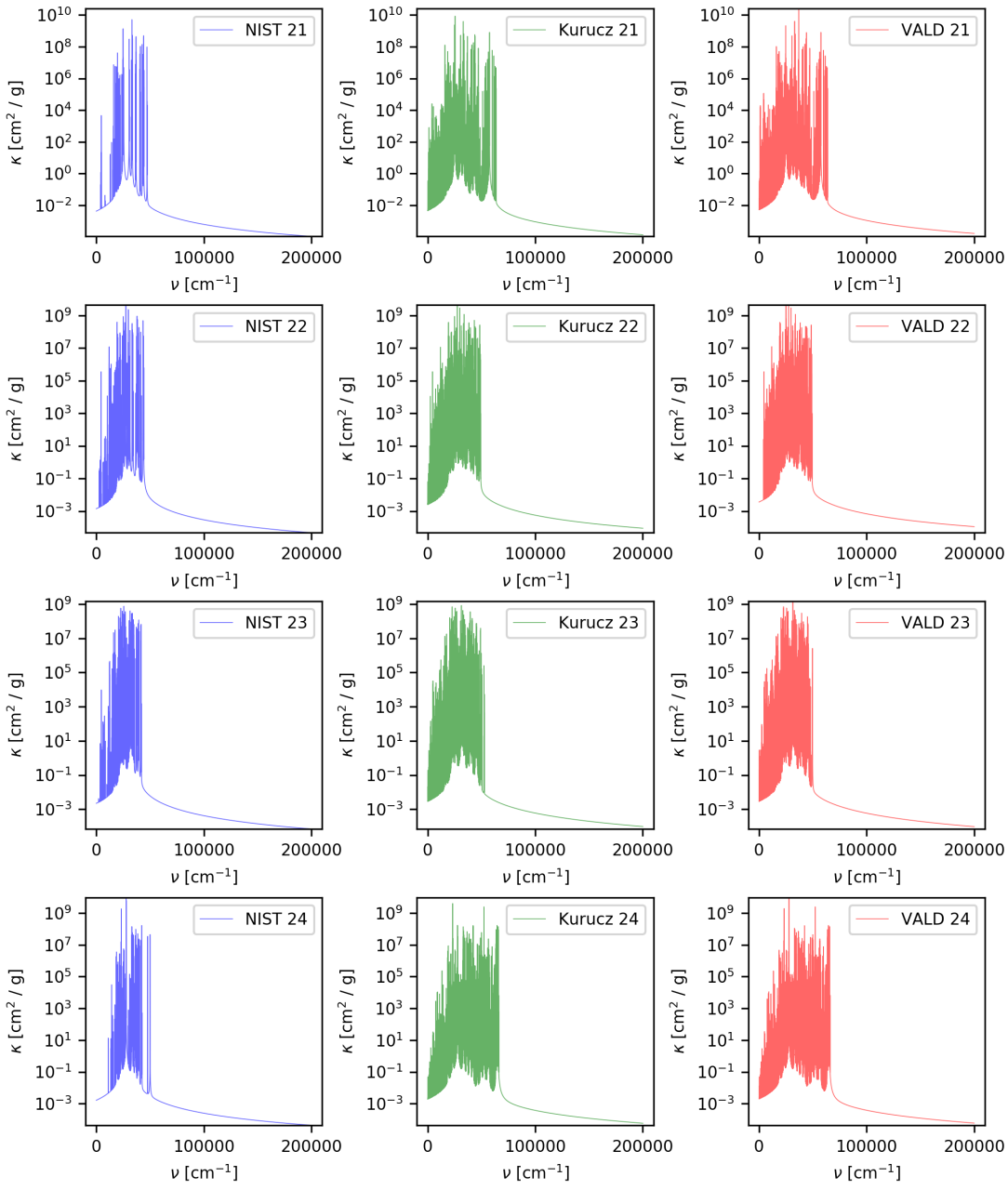

NIST 1
Kurucz 1
VALD 1
NIST 2
Kurucz 2
VALD 2
NIST 3
Kurucz 3
VALD 3
NIST 4
Kurucz 4
VALD 4
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

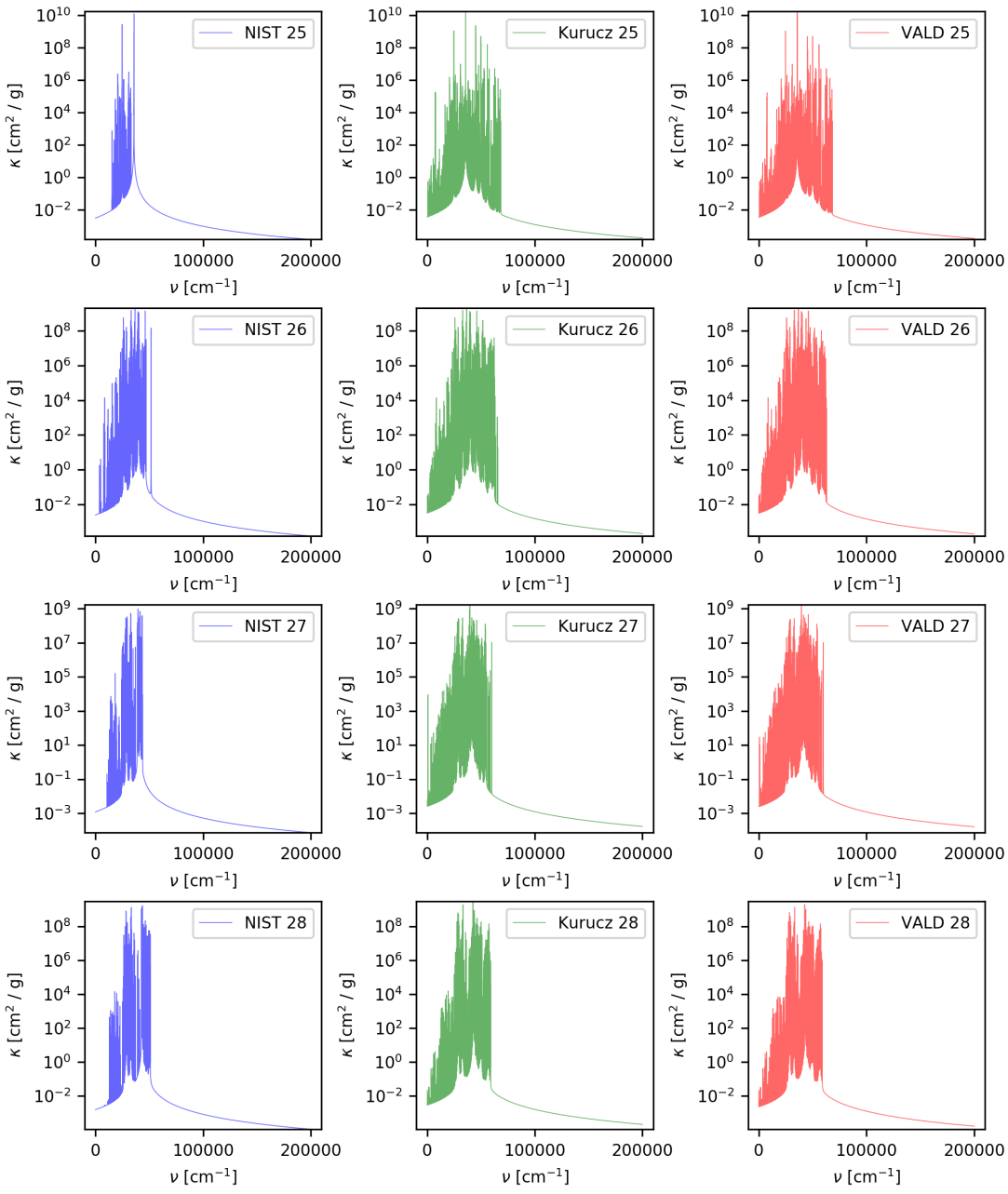

NIST 5
Kurucz 5
VALD 5
NIST 6
Kurucz 6
NIST 7
Kurucz 7
VALD 7
NIST 8
Kurucz 8
VALD 8
κ [cm² / g]
ν [cm⁻¹]

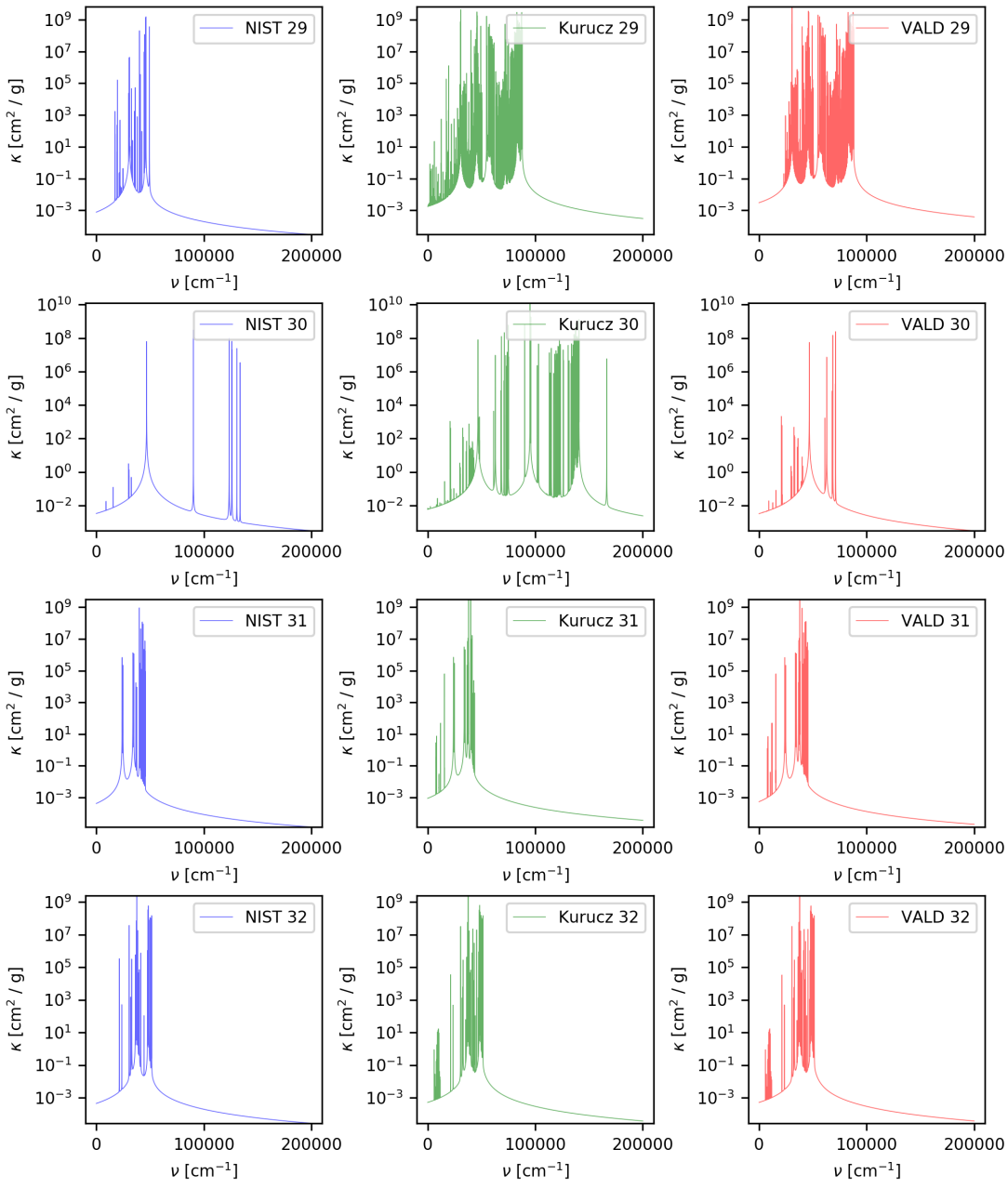

NIST 9
Kurucz 9
VALD 9
NIST 10
Kurucz 10
VALD 10
NIST 11
Kurucz 11
VALD 11
NIST 12
Kurucz 12
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000
0.00
0.25
0.50
0.75
1.00
0.0
0.2
0.4
0.6
0.8
1.0

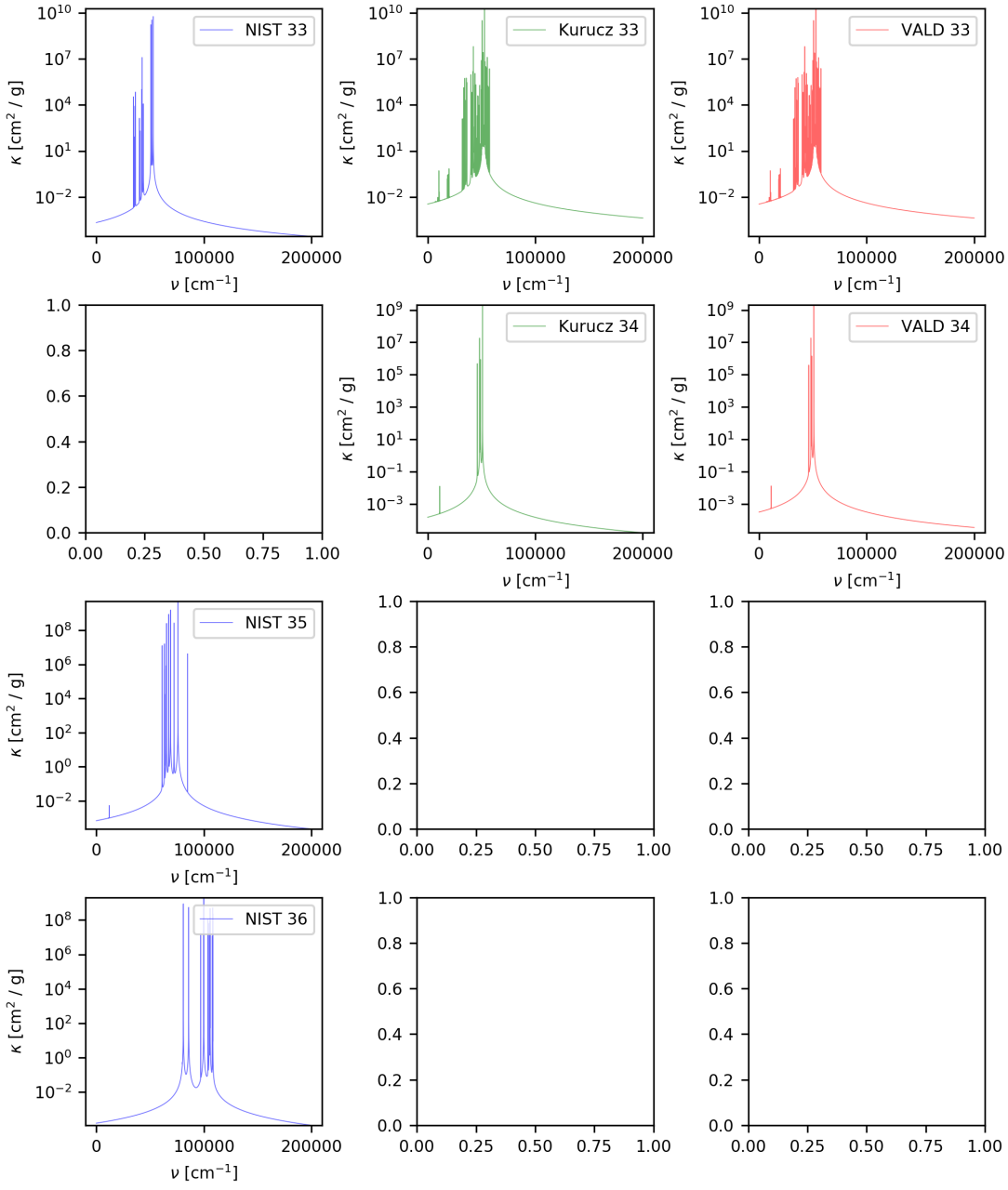

NIST 13
Kurucz 13
VALD 13
NIST 14
Kurucz 14
VALD 14
NIST 15
Kurucz 15
VALD 15
NIST 16
Kurucz 16
VALD 16
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

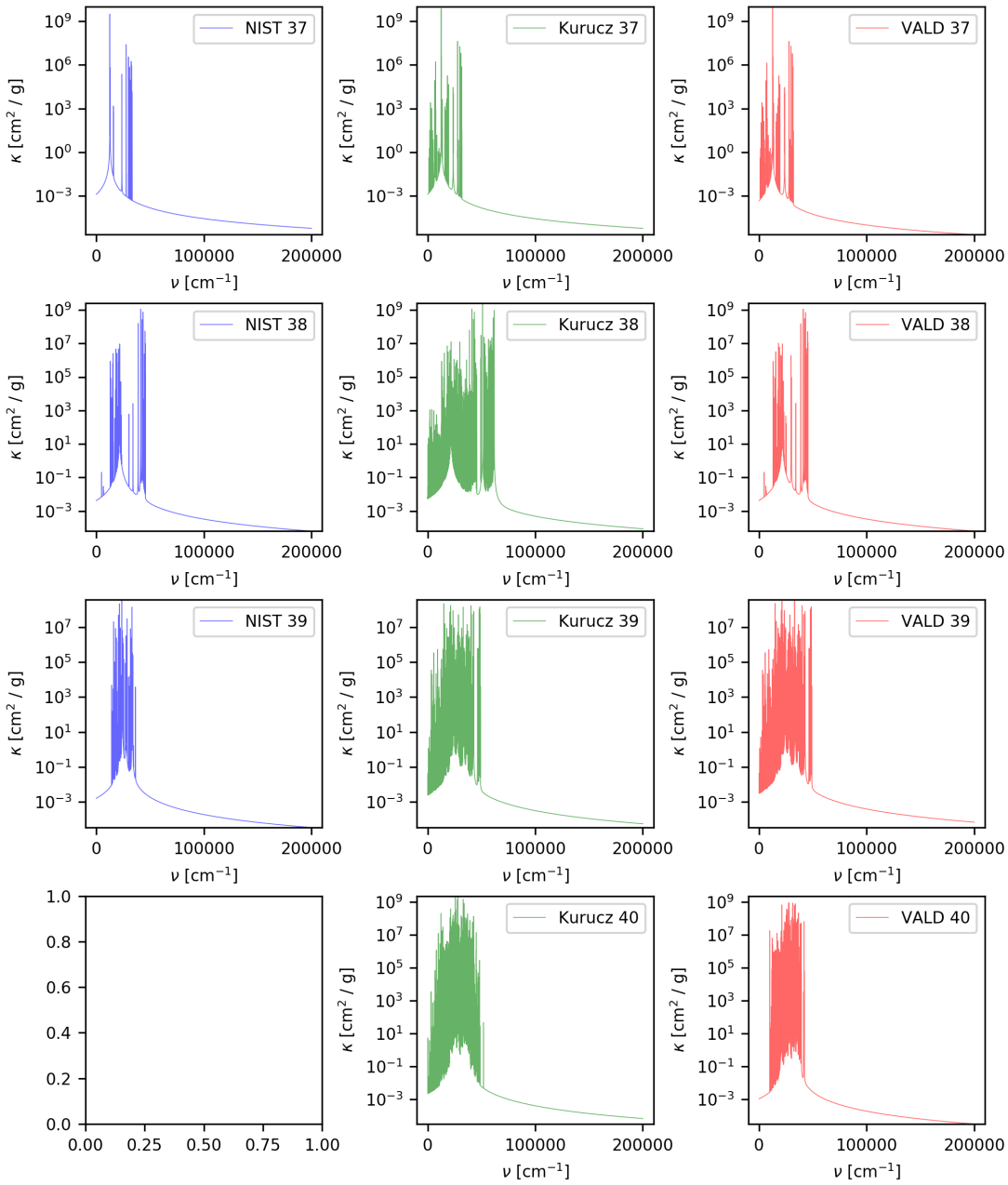

NIST 17
Kurucz 17
VALD 17
NIST 18
Kurucz 18
NIST 19
Kurucz 19
VALD 19
NIST 20
Kurucz 20
VALD 20
κ [cm² / g]
ν [cm⁻¹]

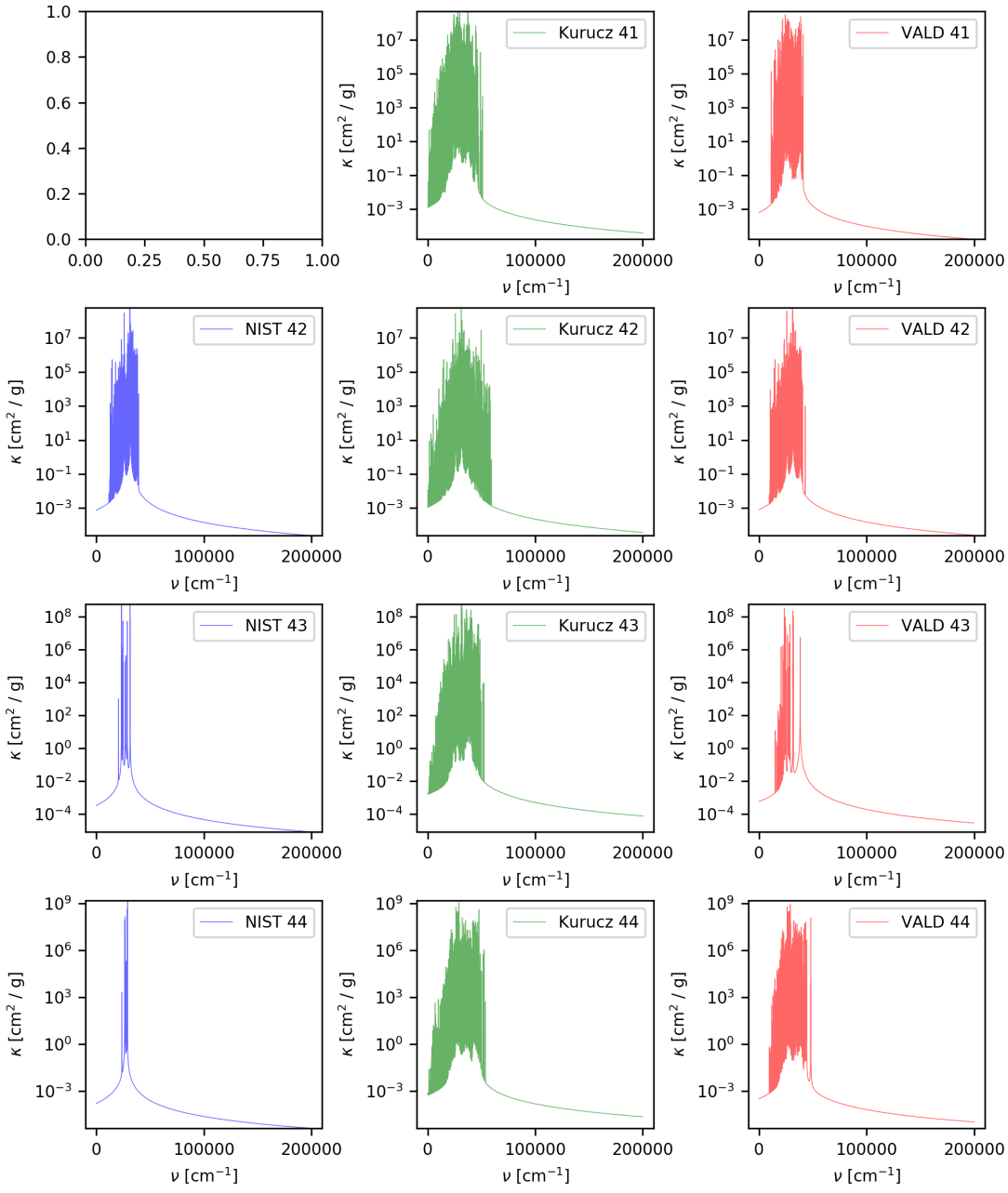

NIST 21
Kurucz 21
VALD 21
NIST 22
Kurucz 22
VALD 22
NIST 23
Kurucz 23
VALD 23
NIST 24
Kurucz 24
VALD 24
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

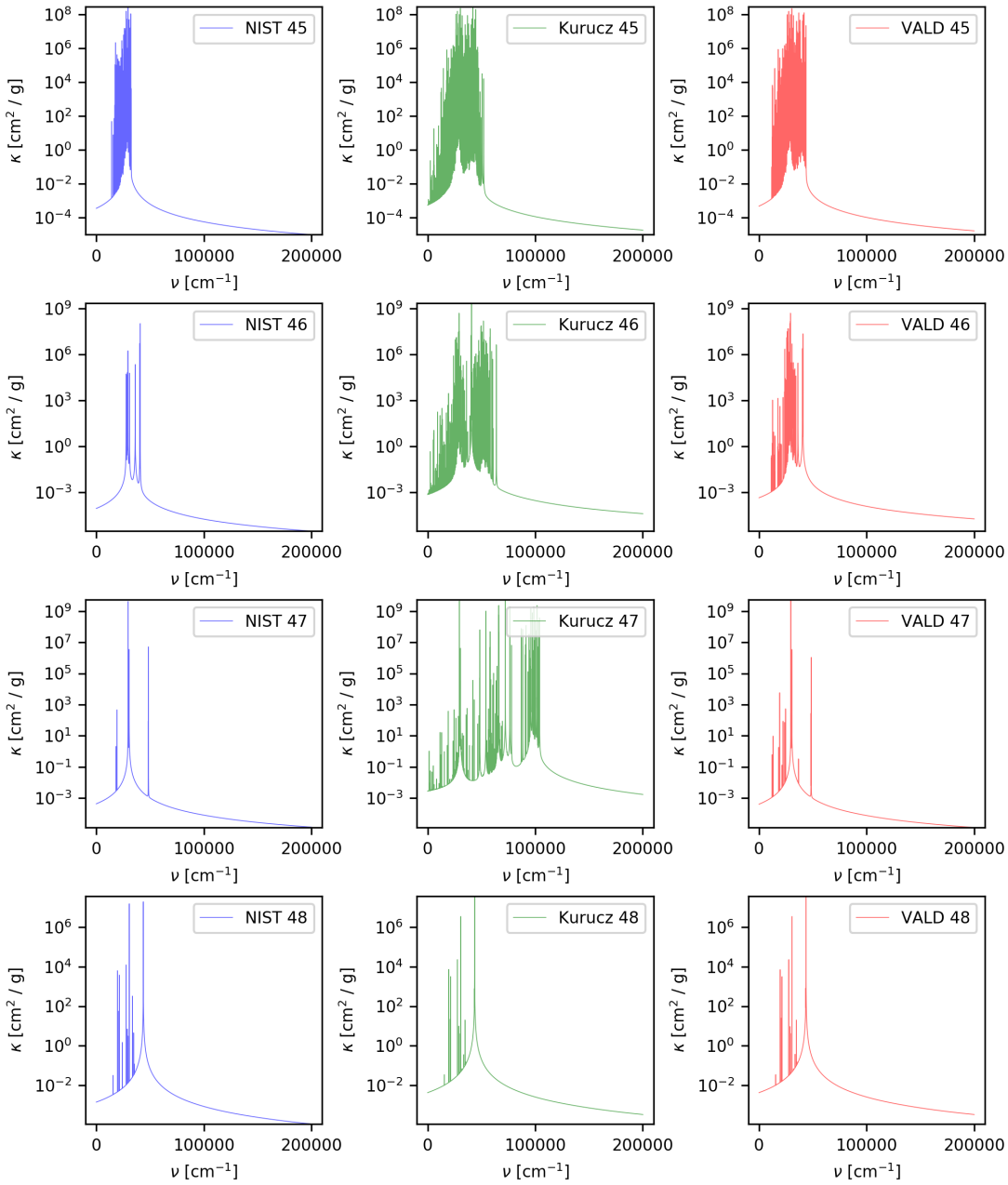

NIST 25
Kurucz 25
VALD 25
NIST 26
Kurucz 26
VALD 26
NIST 27
Kurucz 27
VALD 27
NIST 28
Kurucz 28
VALD 28
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

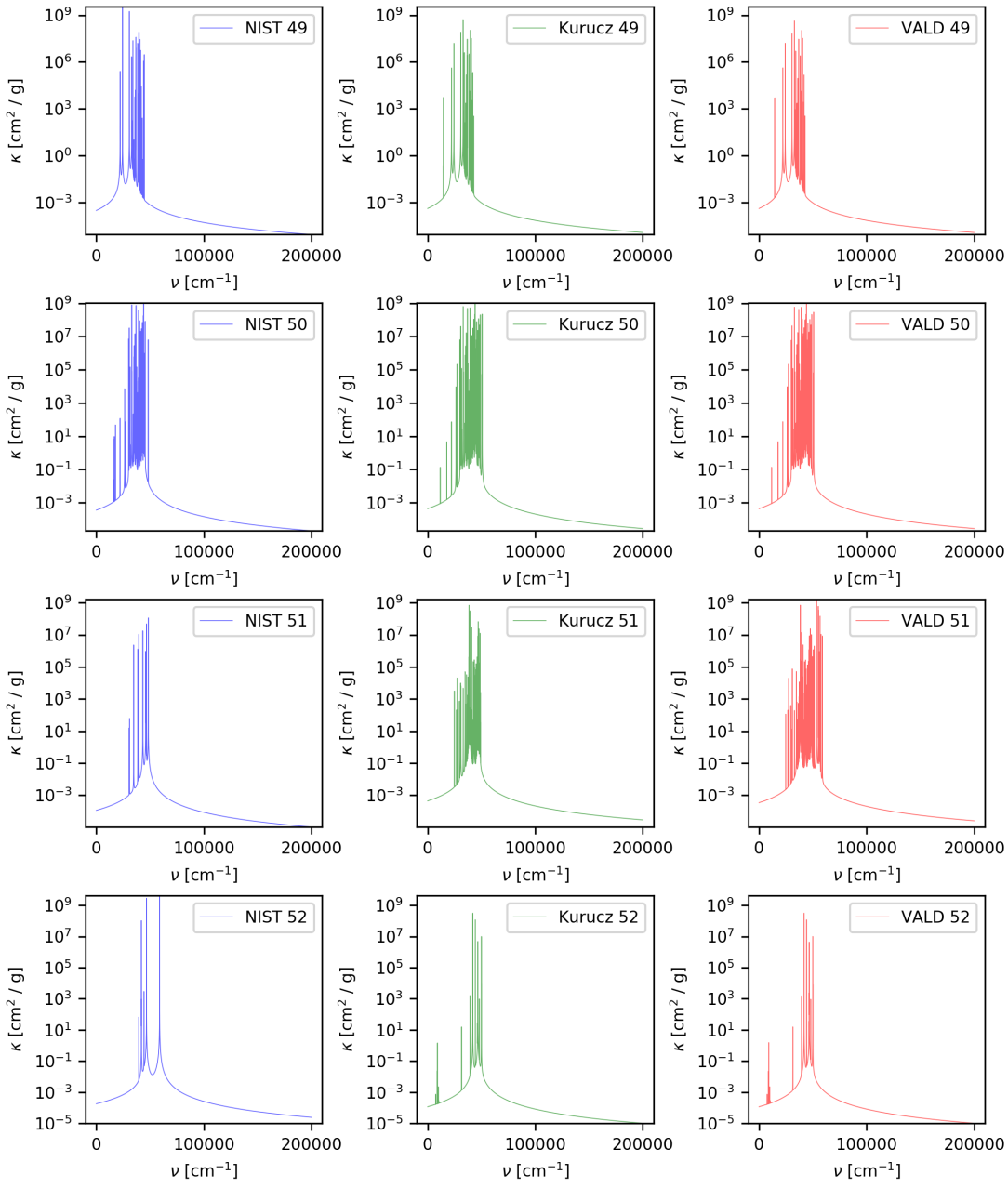

NIST 29
Kurucz 29
VALD 29
NIST 30
Kurucz 30
VALD 30
NIST 31
Kurucz 31
VALD 31
NIST 32
Kurucz 32
VALD 32
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

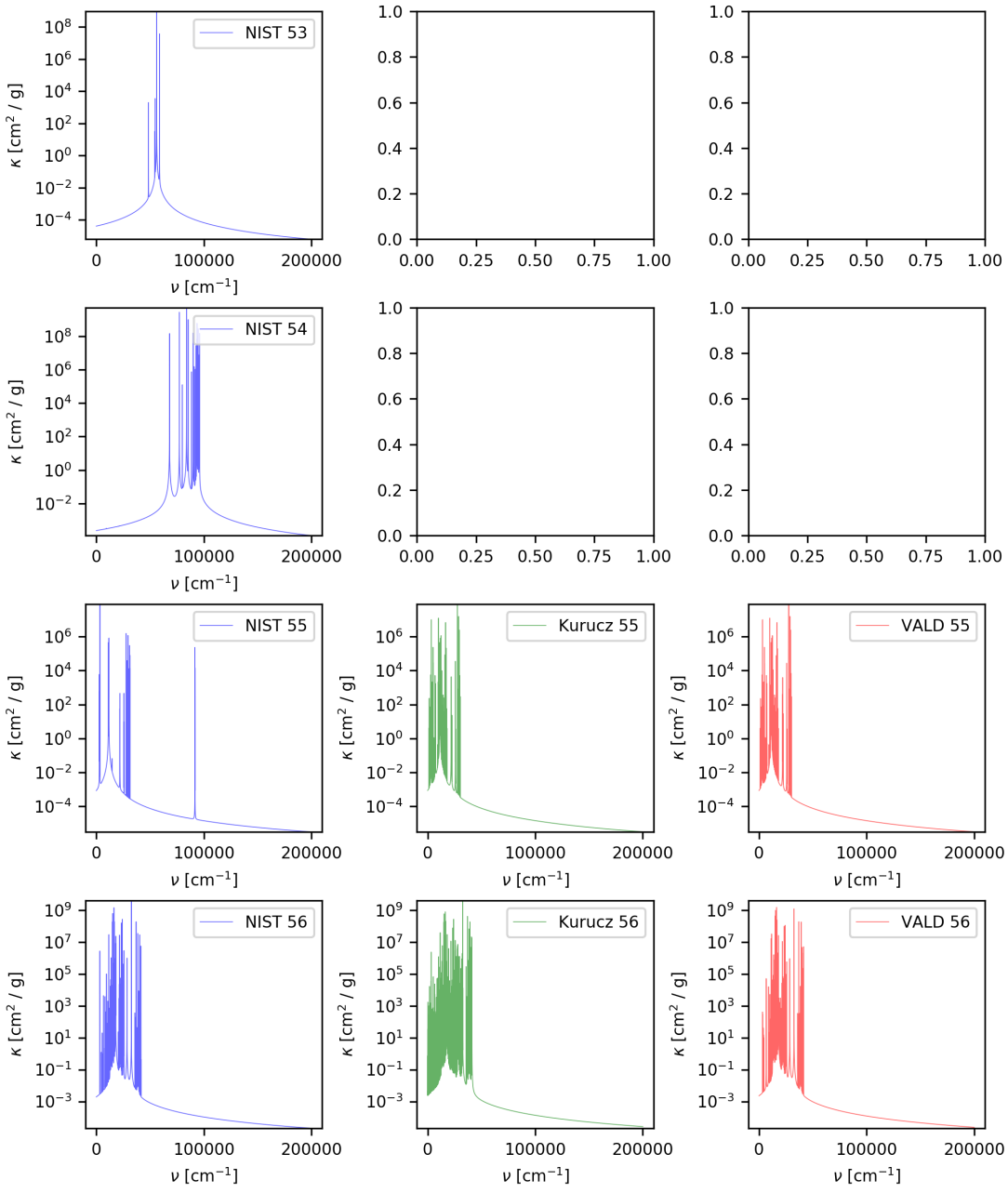

NIST 33
Kurucz 33
VALD 33
Kurucz 34
VALD 34
NIST 35
NIST 36
κ [cm² / g]
ν [cm⁻¹]

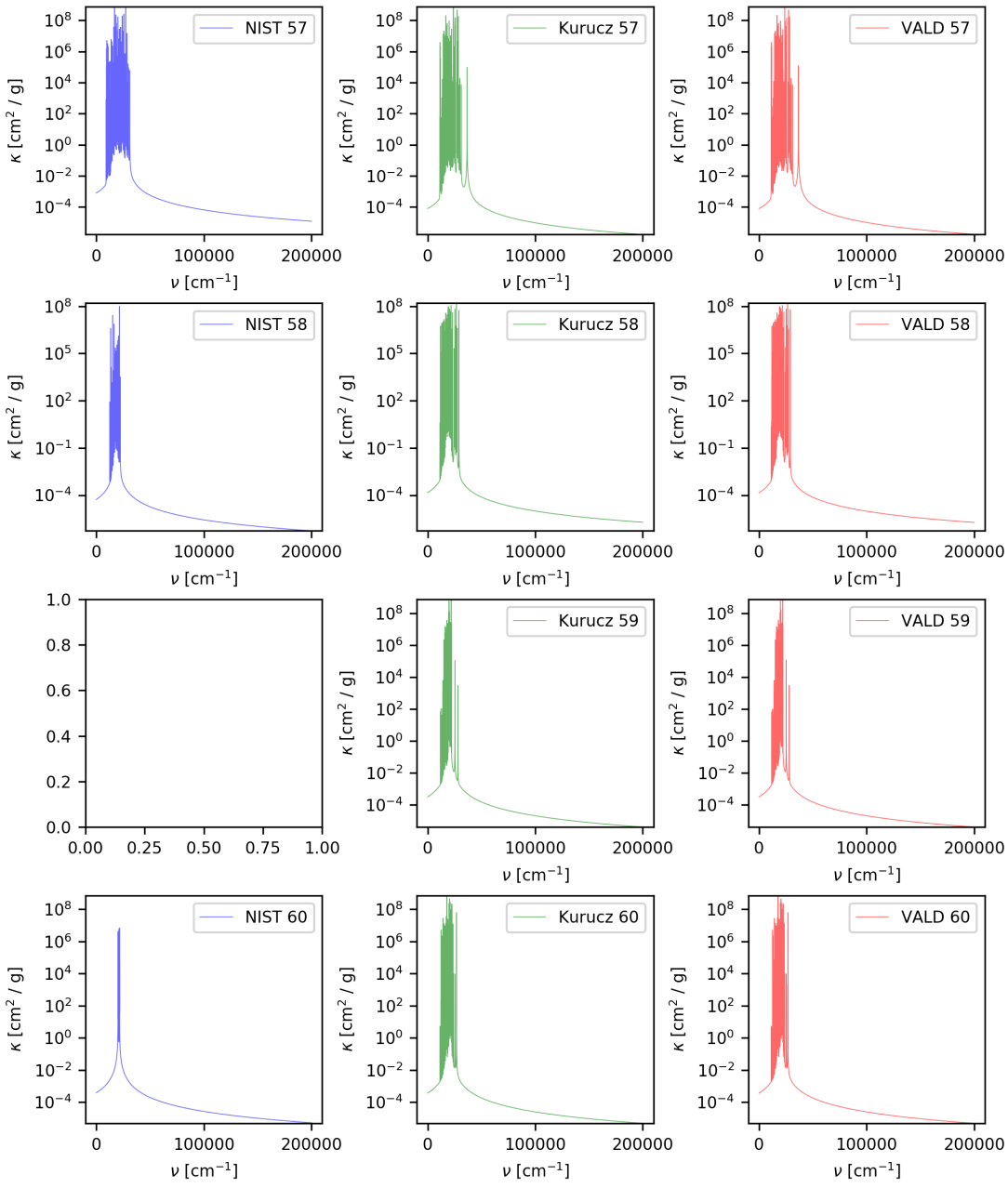

NIST 37
Kurucz 37
VALD 37
NIST 38
Kurucz 38
VALD 38
NIST 39
Kurucz 39
VALD 39
Kurucz 40
VALD 40
κ [cm² / g]
ν [cm⁻¹]

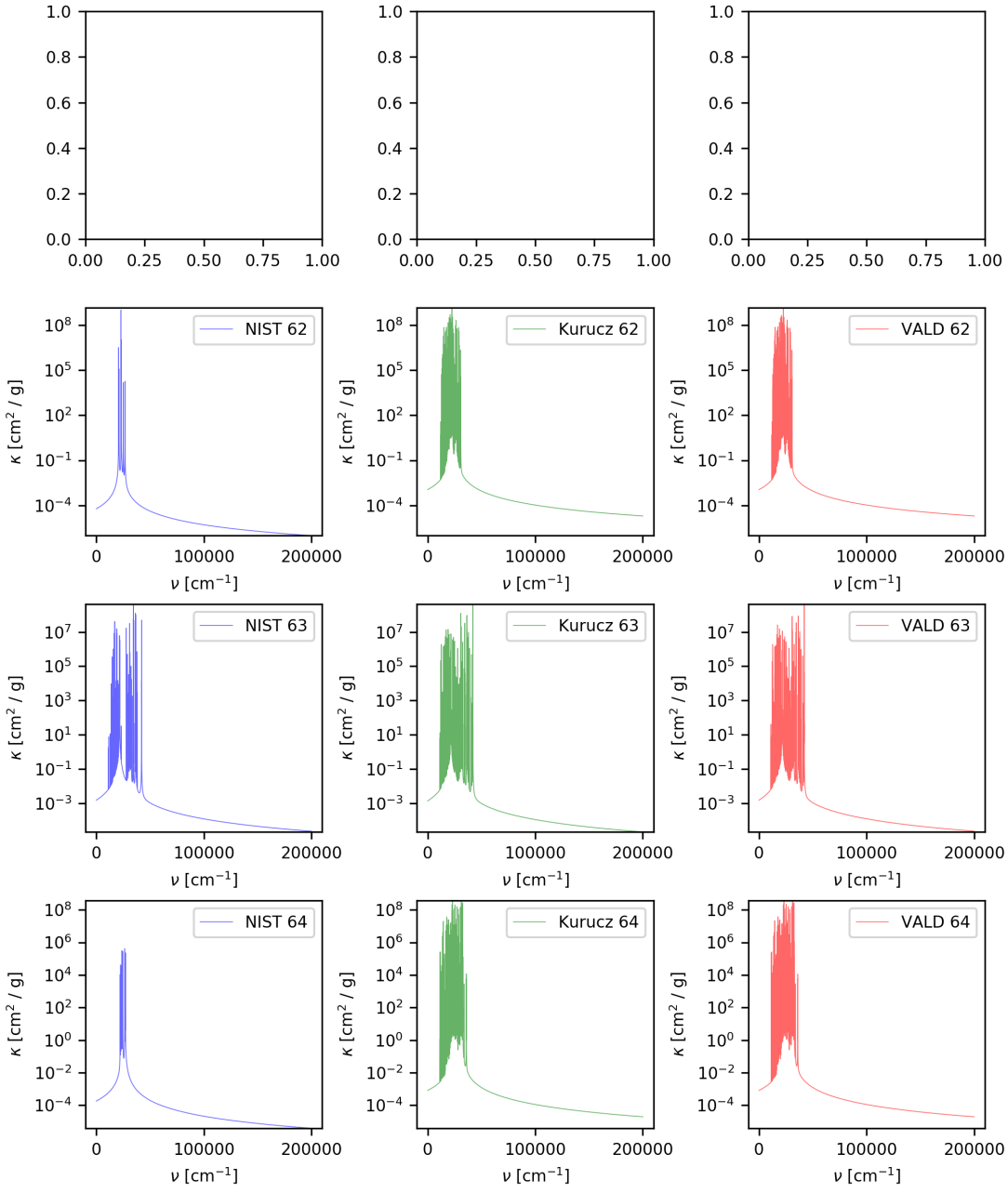

Kurucz 41
VALD 41
NIST 42
Kurucz 42
VALD 42
NIST 43
Kurucz 43
VALD 43
NIST 44
Kurucz 44
VALD 44
κ [cm² / g]
ν [cm⁻¹]

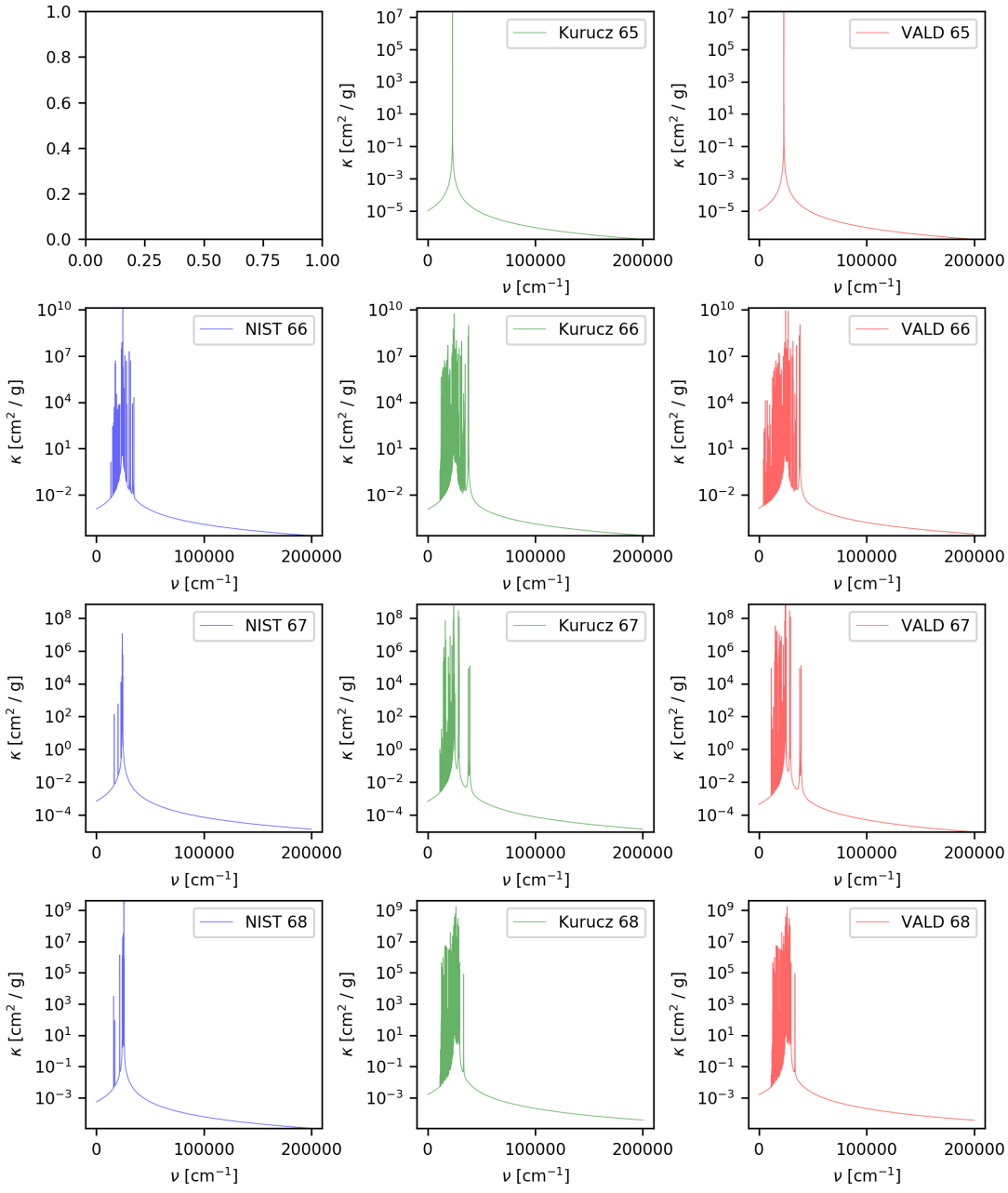

NIST 45
Kurucz 45
VALD 45
NIST 46
Kurucz 46
VALD 46
NIST 47
Kurucz 47
VALD 47
NIST 48
Kurucz 48
VALD 48
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

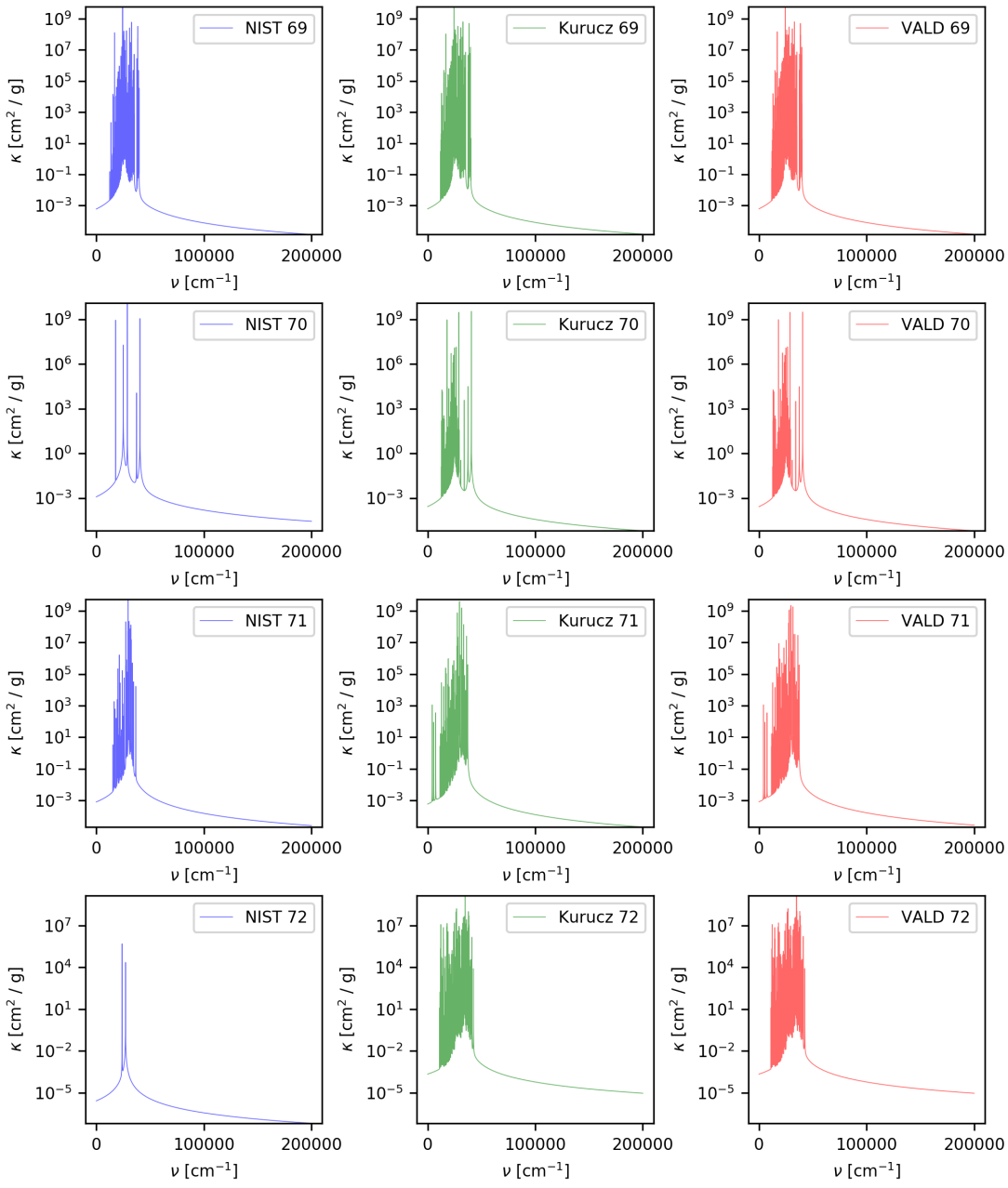

NIST 49
Kurucz 49
VALD 49
NIST 50
Kurucz 50
VALD 50
NIST 51
Kurucz 51
VALD 51
NIST 52
Kurucz 52
VALD 52
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000

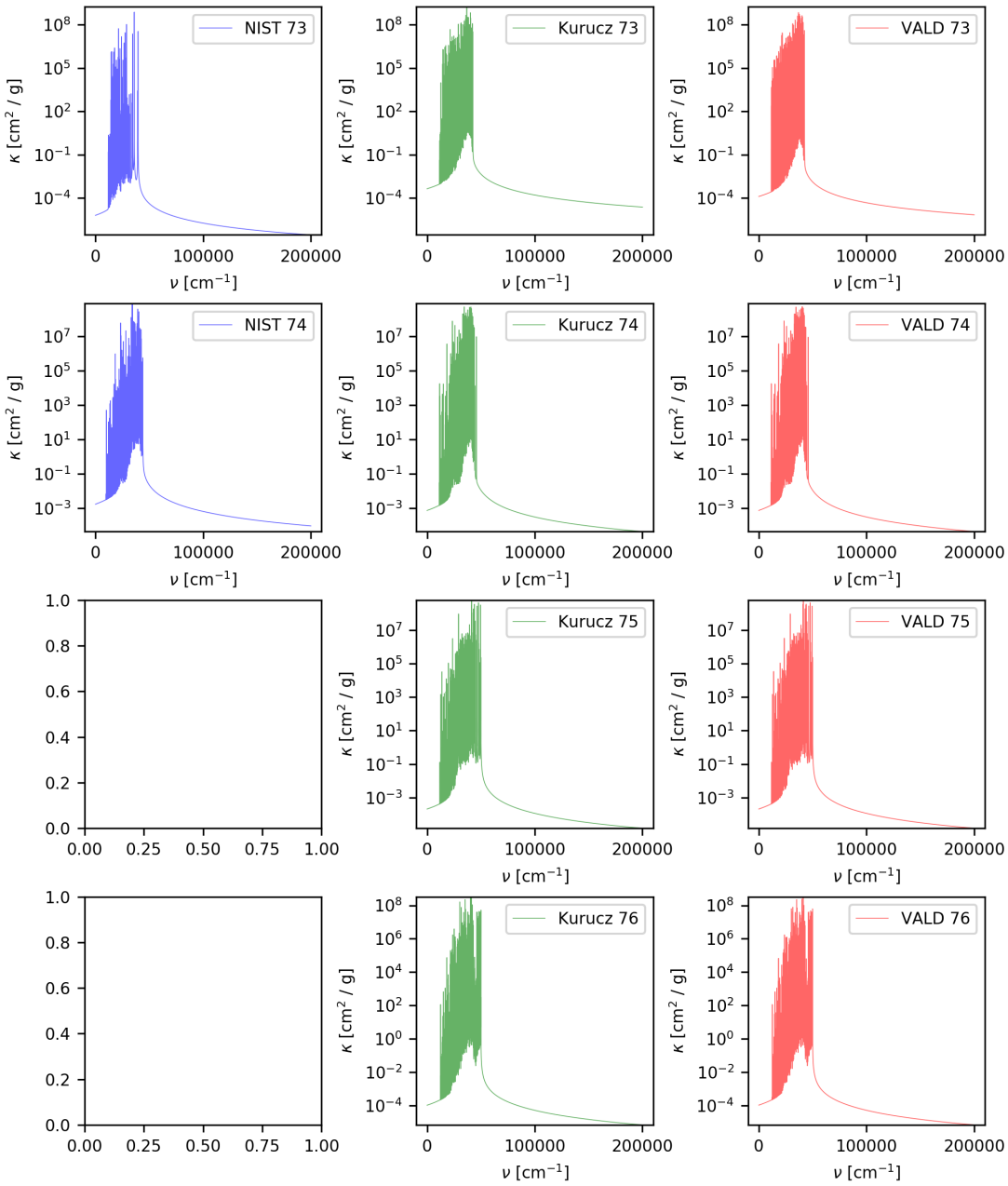

NIST 53
NIST 54
NIST 55
Kurucz 55
VALD 55
NIST 56
Kurucz 56
VALD 56
κ [cm² / g]
ν [cm⁻¹]

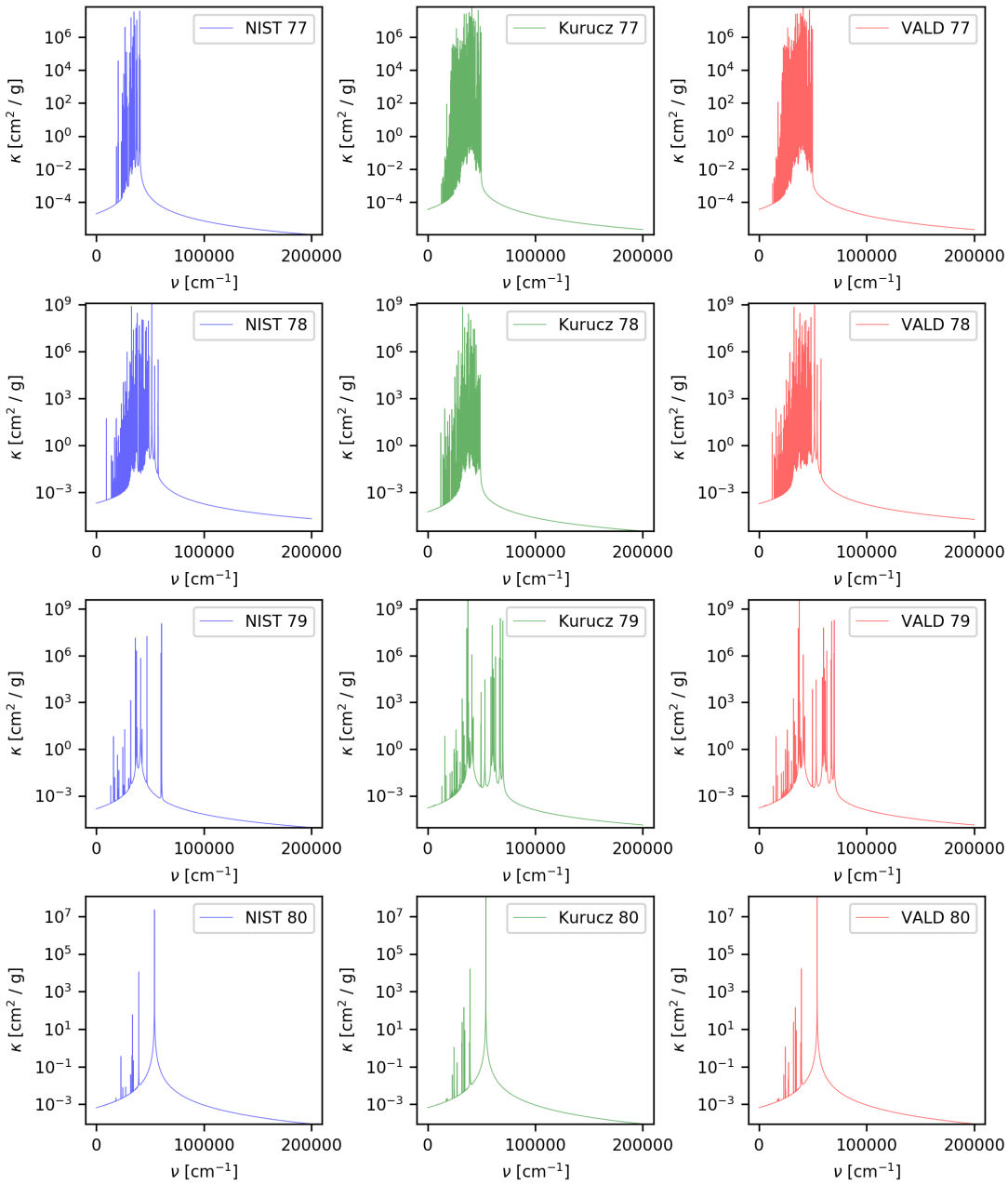

NIST 57
Kurucz 57
VALD 57
NIST 58
Kurucz 58
VALD 58
Kurucz 59
VALD 59
NIST 60
Kurucz 60
VALD 60
κ [cm² / g]
ν [cm⁻¹]

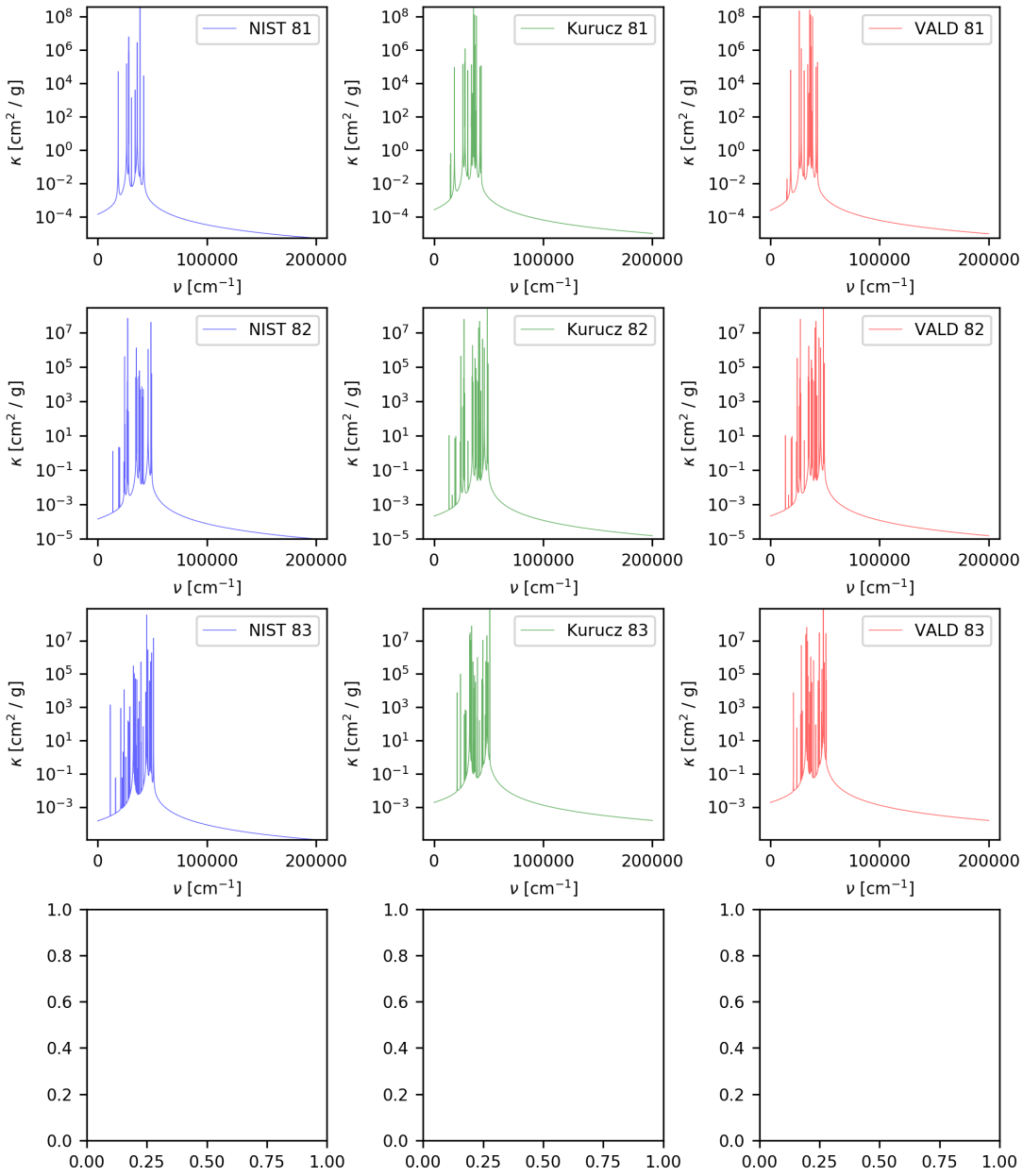

NIST 62
Kurucz 62
VALD 62
NIST 63
Kurucz 63
VALD 63
NIST 64
Kurucz 64
VALD 64
κ [cm² / g]
ν [cm⁻¹]

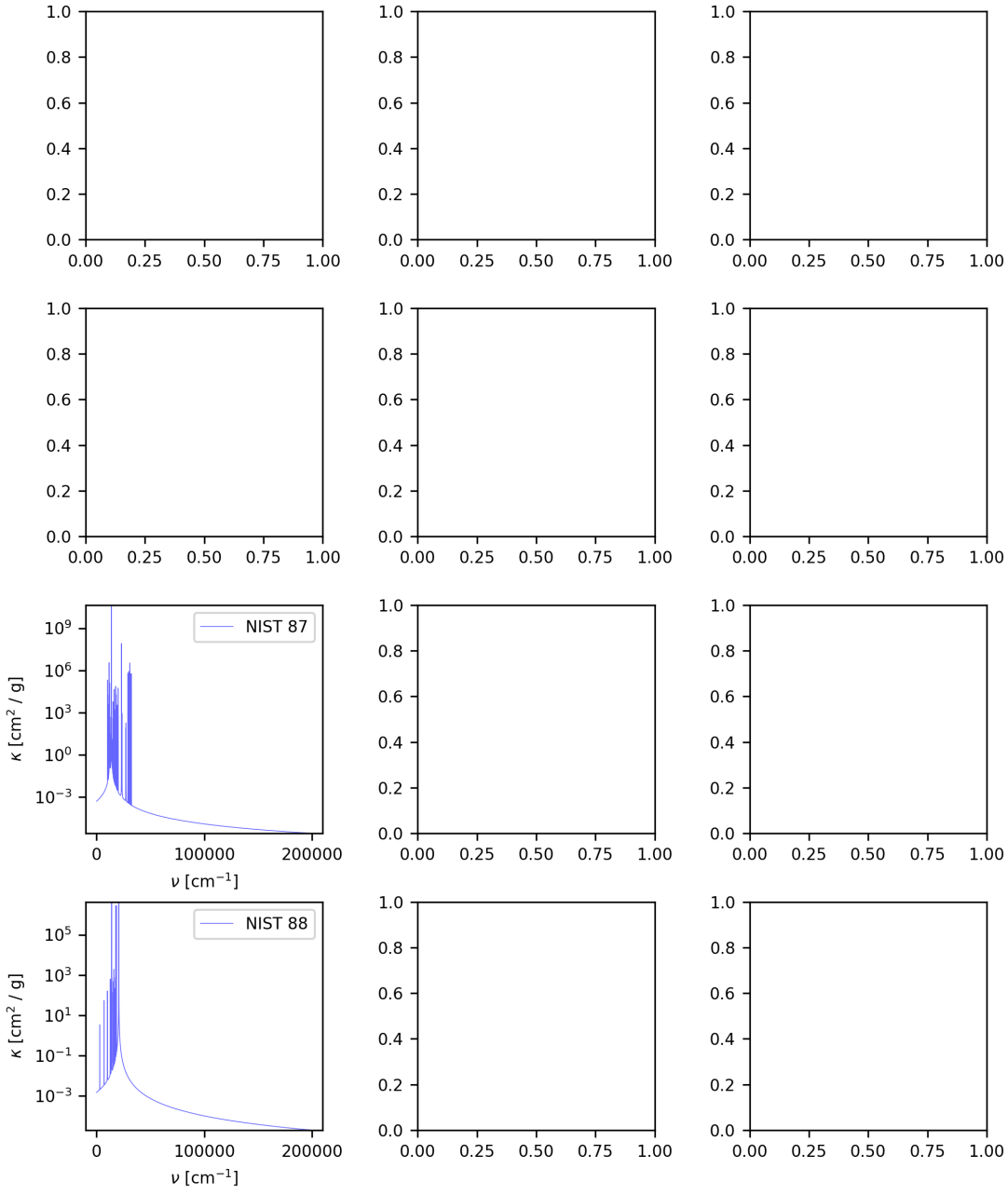

Kurucz 65
VALD 65
NIST 66
Kurucz 66
VALD 66
NIST 67
Kurucz 67
VALD 67
NIST 68
Kurucz 68
VALD 68
κ [cm² / g]
ν [cm⁻¹]

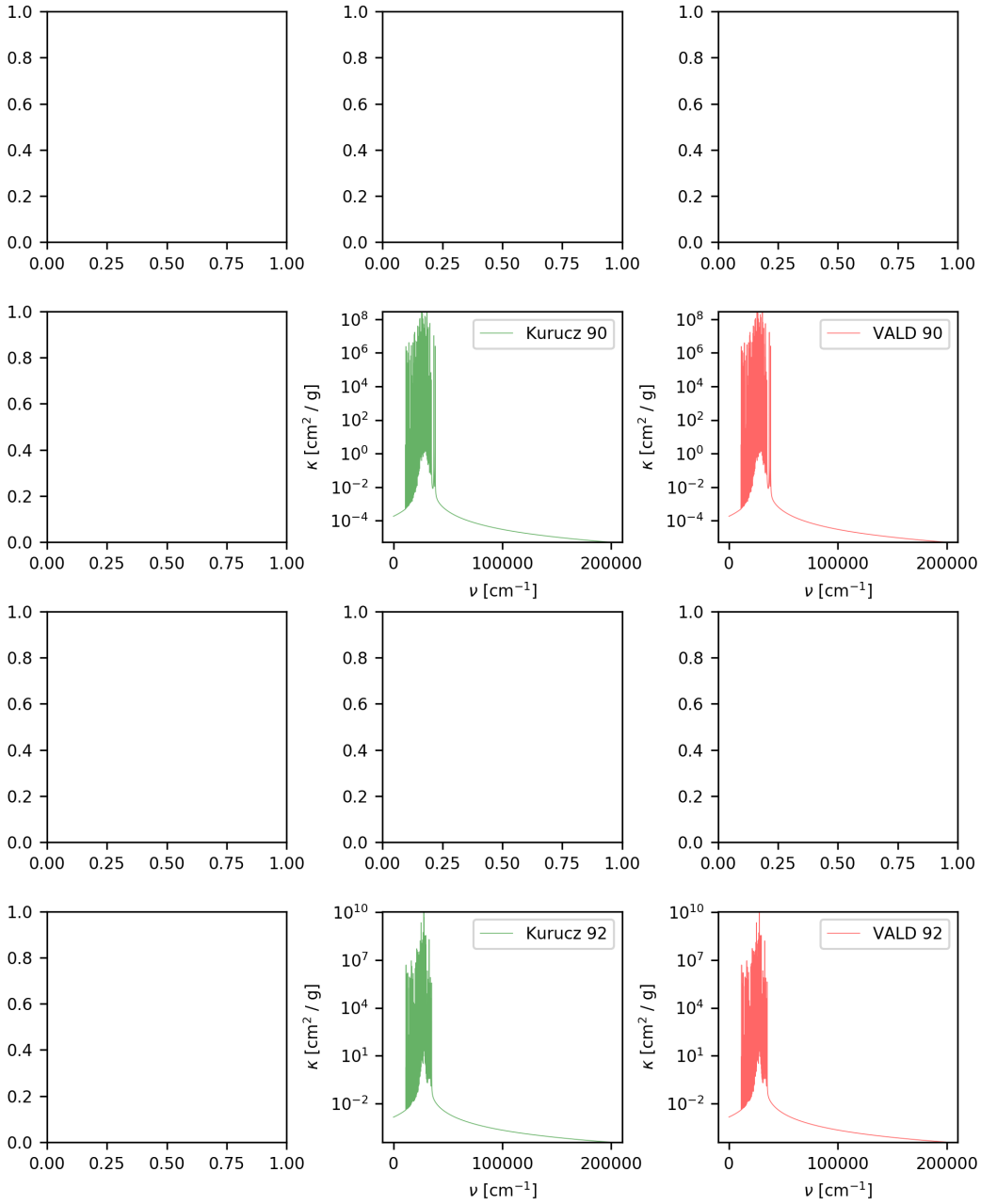

NIST 69
Kurucz 69
VALD 69
NIST 70
Kurucz 70
VALD 70
NIST 71
Kurucz 71
VALD 71
NIST 72
Kurucz 72
VALD 72
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000


NIST 73
Kurucz 73
VALD 73
NIST 74
Kurucz 74
VALD 74
Kurucz 75
VALD 75
Kurucz 76
VALD 76
κ [cm² / g]
ν [cm⁻¹]


NIST 77
Kurucz 77
VALD 77
NIST 78
Kurucz 78
VALD 78
NIST 79
Kurucz 79
VALD 79
NIST 80
Kurucz 80
VALD 80
κ [cm² / g]
ν [cm⁻¹]
0
100000
200000


NIST 81
Kurucz 81
VALD 81
NIST 82
Kurucz 82
VALD 82
NIST 83
Kurucz 83
VALD 83
κ [cm² / g]
ν [cm⁻¹]


NIST 87
NIST 88
κ [cm² / g]
ν [cm⁻¹]


Kurucz 90
VALD 90
Kurucz 92
VALD 92
κ [cm² / g]
ν [cm⁻¹]

CHAPTER

# SEVEN

# CITATIONS

If you make use of HELIOS-K in your work, please cite

Grimm et Al. (2021):

```
@ARTICLE{2021ApJS..253...30G,
       author = {{Grimm}, Simon L. and {Malik}, Matej and {Kitzmann}, Daniel and {Guzm{\
↪'a}n-Mesa}, Andrea and {Hoeijmakers}, H. Jens and {Fisher}, Chloe and {Mendon{\c{c}}a},
↪ Jo{\~a}o M. and {Yurchenko}, Sergey N. and {Tennyson}, Jonathan and {Alesina}, Fabien␣
↪and {Buchschacher}, Nicolas and {Burnier}, Julien and {Segransan}, Damien and {Kurucz},
↪ Robert L. and {Heng}, Kevin},
        title = "{HELIOS-K 2.0 Opacity Calculator and Open-source Opacity Database for␣
↪Exoplanetary Atmospheres}",
      journal = {\apjs},
     keywords = {Exoplanet atmospheres, 487, Astrophysics - Earth and Planetary␣
↪Astrophysics, Astrophysics - Instrumentation and Methods for Astrophysics},
         year = 2021,
        month = mar,
       volume = {253},
       number = {1},
          eid = {30},
        pages = {30},
          doi = {10.3847/1538-4365/abd773},
archivePrefix = {arXiv},
       eprint = {2101.02005},
 primaryClass = {astro-ph.EP},
       adsurl = {https://ui.adsabs.harvard.edu/abs/2021ApJS..253...30G},
      adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

or

Grimm & Heng (2015):

```
@ARTICLE{2015ApJ...808..182G,
       author = {{Grimm}, Simon L. and {Heng}, Kevin},
        title = "{HELIOS-K: An Ultrafast, Open-source Opacity Calculator for Radiative␣
↪Transfer}",
      journal = {\apj},
     keywords = {methods: numerical, planets and satellites: atmospheres, radiative␣
↪transfer, Astrophysics - Earth and Planetary Astrophysics, Physics - Atmospheric and␣
↪Oceanic Physics},
         year = "2015",
```

(continues on next page)

(continued from previous page)

```
        month = "Aug",
       volume = {808},
       number = {2},
          eid = {182},
        pages = {182},
          doi = {10.1088/0004-637X/808/2/182},
archivePrefix = {arXiv},
       eprint = {1503.03806},
 primaryClass = {astro-ph.EP},
       adsurl = {https://ui.adsabs.harvard.edu/abs/2015ApJ...808..182G},
      adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```