Forward Inverse solver of the Radiative Transfer Equation for Zeeman polarization in geometrical scale

FIRTEZ-dz

v0.0

User manual

Pastor Yabar, A. and Borrero, J. M.

2021.05

Contents

The authors assume no responsibility or liability for any errors or omissions in the content of this site. The information contained in this site is provided on an "as is" basis with no guarantees of completeness, accuracy, usefulness or timeliness...

1 Compilation

1.1 Pre-requisites:

Before attempting the compilation of **FIRTEZ-dz**, you need the following libraries:

- **LAPACK:**
  Versions tested: 3.7.0, 3.9.0
  You can download it from here.

- **MPI:**
  Versions tested: 3.1.0 (openMPI), 3.3.2 (MPICH)
  You can download it from here.

- **FFTW3:**
  Versions tested: 3.3.8
  You can download it from here.

1.2 Compilation:

Once this software is compiled you can proceed with the compilation of **FIRTEZ-dz**. In *src/* directory there is a *makefile* file. At the beginning of the file you should define the following variables according to your case:

- *LIBS*:
  If you installed (or you already had them installed) all the above libraries in the default library path, you can leave this variable empty. Otherwise, add as many of the following lines as needed, i.e., only those for which your installation is not in the default path.
  *LIBS= *
  *-L /path_to_lapack_lib_dir/ -llapack -lblas *
  -L /path_to_fftw3_lib_dir/ -lfftw3

- *INCS*:
  If you installed (or you already had them installed) all the software (see Sec. 1.1) in the default library path, you can leave this variable empty. Otherwise add as many as the following lines as needed:
  *INCS= *
  -I /path_to_fftw3_include_dir/

Once done, save the *makefile* and you are almost ready to compile the code.

However, it is **IMPORTANT** to notice that the file *splines.f90* does only provide the name of the four subroutines and the input and output variables of each one. The body of each subroutine (*SPLIE2*, *SPLIN2*, *SPLINE*, and *SPLINT*) inside the *splines.f90* file must be filled using Numerical Recipes (it might be interesting, depending on the size of model atmospheres you are planning to use to increase the *NN* parameter of *SPLIE2* and *SPLIN2* to larger values, for instance 1000).

Once these four subroutines are written and in the same *src* directory, you can proceed typing:

$ make clean

This will remove all the **.mod*, **.o*, and **.x* files and then:

$ make FIRTEZ-dz

or:
$ make

With this the code should compile successfully, creating an executable file called: *FIRTEZ-dz.x*. By default, *gnu* fortran compiler is assumed, yet if you want to use intel's, then you compile the code typing:

$ make FIRTEZ-dz COMP=intel

or
$ make COMP=intel

1.3 Tests:

The standard distribution includes some tests to check the correct compilation of **FIRTEZ-dz**, and also, they are used in the python notebook (*firtez_dz_usage_example.ipynb*) included in the *python* directory. Also, inside the directory of each test, it is possible to find the *control_file.dat* that can help when looking through the *input file* in the next section (see Sect. 2). In order to run all the tests, move to the *tests* directory and execute the shell script *run_tests.sh* by typing:

$ ./run_tests.sh

2 Execution

The execution of the FIRTEZ-dz code is performed as:

$ *mpirun -n # ./firtez-dz.x control_file.dat*

Where # must be substituted for whatever number of processes you want to run. Currently, FIRTEZ-dz has been tested in clusters with up to 1000 nodes without issues.

For remote jobs with long execution times we recommend to run FIRTEZ-dz as:

$ *nohup mpirun -n # ./firtez-dz.x control_file.dat &*

This way the job will not stop should your internet connection drop.

While FIRTEZ-dz is running, it also produces a lot of standard output on the console. Some of this output includes error messages and how to solve them. Therefore we do not recommend to run in silent mode (*>> /dev/null*). Then again, for remote jobs this might be your only option if the connection is intermitted. In this case, if you still want to run have a log file (i.e. *output.log*) to see what could have gone wrong, we recommend to run the job as:

$ *nohup mpirun -n # ./firtez-dz.x control_file.dat — tee output.log &*

When executing the FIRTE-DZ code, besides the executable itself (*firtez-dz.x*), we will need access to two additional files: the input/control file and the spectral line database file. The input/control file is commonly named *control_file.dat* but you can choose to rename it as long as you provide the new name in the execution line above. The spectral line database file must always be named as *lines_database.dat* as this is hardcoded inside FIRTEZ-dz. However, you can modify this file (see below) to add/remove spectral lines.

2.1 Input/Control file:

It is a plain text file. It contains a number of **fields**, some of which are mandatory while others are optional (we will let you know later). Each field is defined starting with its name (in capital letters) followed by colon *:* and ends whenever another field starts. Anything written after *END:* is ignored. An example is provided after explaining each of the possible fields.

The following list covers ALL the fields the code works with, anything outside is omitted. The order with which the various **fields** appear does not matter.

- *LINES:* **Mandatory**
  It specifies the line/lines from the ones provided in the spectral lines database file (see Sec. 2.3). It might contain as many spectral regions as desired. The format to supply them is as

follows:

index nnn init ddd

where the various parameters are:

- *index* refers to the spectral line index or indexes of the spectral line database to be synthesised together. If one is supplied the RTE is solved for that line. If several indexes (separated by comma) are supplied, then the RTE is solved for the whole set simultaneously.
- *nnn* (integer) specifies the number of spectral points to be solved.
- *init* (float) specifies the difference with respect to the reference wavelength set in the spectral line database (in mÅ) of the first wavelength point to consider. For the case of blended spectral regions, the reference wavelength is the one appearing in the spectral line database for the very first of the spectral lines specified.
- *ddd* (float) is the wavelength step in between each wavelength point in mÅ.

An example of such field would be as follows:

Input file Ex. 2.1: *Lines*

```
...
LINES:
2 11 -100.0 20.0
4,5 71 -200.0 40.0
...
```

In the first line, we specify that the Stokes vector corresponding to the spectral line identified with index 2 in *lines_database.dat* is to be calculated in 11 spectral points, starting from -200.0 mÅ of the line core with a spectral sampling of *20.0* mÅ. If we now look into *lines_database.dat* we see that FIRTEZ-dz will then calculate the Stokes vector for Fe I 6173.3354 Å at positions: $6173.2354, 6173.2554, ..., 6173.3354, ..., 6173.4154, 6173.4354$ Å.

In the second line, we use spectral line 4 in *lines_database.dat*. Note that now, this line is followed by a comma plus an additional number. This indicates that spectral line 5 in *lines_database.dat* is to be computed along with 4 while considering them as being blended. This spectral region is sampled at 71 spectral points starting from -200.0 mÅ of the line core of the first (index 4) with a spectral sampling of 40.0 mÅ. Therefore, FIRTEZ-dz will calculate the Stokes vector for Fe I 6301.5010 Å and Fe I 6302.4937 Å at positions: $6301.3010, 6301.3210, 6301.3410, ..., 6301.5010, 6301.5210, ..., 6302.9010, 6302.9210$ Å. Note that the final wavelengths are well past (i.e. on the red wing) of the second spectral line.

- *BOX:* **Optional**

  This field specifies the spatial dimension of the considered atmosphere to be handled. It has three mandatory lines that refer, respectively, to the x, y, and z spatial dimension and the three of them follow the same format:

  nn dd

  where *nn* is the number of points (integer) in each dimension and *dd* is the step (real number, in km) sampling along each spatial direction. As an example, let us consider the following piece of input file:

  Input file Ex. 2.2: *Box*

  ```
  ...
  BOX:
  256 48.0
  256 48.0
  128 12.0
  ...
  ```

  The example above considers a three dimensional domain driscretized in 256, 256, and 128 grid points along the x, y, and z dimensions, respectively. The last dimension, z **is always considered the direction of the gravity vector**. The grid sizes are 48 km along x and y, but 12 km along z.

- *FILEPROFILE:* **Mandatory** if *MODE: INVERSION* or *MODE: SYNTHESIS* mode, otherwise **Inactive**

  It is a string containing the name of the file where Stokes vector is to be written. It is an optional field BUT it is mandatory for the synthesis and inversion modes. For example, if we would like to store the synthetic profiles for a supplied atmosphere in a file named *synthetic_profile.bin*, we would write the following:

  Input file Ex. 2.3: *Fileprofile*

  ```
  ...
  FILEPROFILE:
  synthetic_profile.bin
  ...
  ```

- *FILEMODEL:* **Mandatory**

This field is also a string containing the name of the model atmosphere to deal with. It is a MANDATORY field. For example, if our atmosphere is in *model_atmosphere.bin*, then we would have:

Input file Ex. 2.4: *Box*

```
...
FILEMODEL:
model_atmosphere.bin
...
```

- *HYDROSTATIC:* **Optional (default:** *NO***)**

This is an optional field. If present it can be *YES* or *NO*. If not present, default is *NO*. If *YES*, the atmosphere is assumed to be in hydrostatic equilibrium. In this case, FIRTEZ-dz will take the uppermost value of the pressure in the atmospheric model, $P_{\rm g}(z_{\rm max})$), under *FILEMODEL* as a boundary condition, and then it integrate downwards the eqution of hydrosatic equilibrium, using a Runge-Kutta 4, to obtain the gas pressure and density at all other atmospheric heights. If *NO* or not present, it takes the supplied gas pressures and densities to proceed with the resolution of the radiative transfer equation[1]. For example, if we want to assume hydrostatic equilibrium, we would have the following:

Input file Ex. 2.5: *Hydrostatic*

```
...
HYDROSTATIC:
YES
...
```

- *MODE:* **Mandatory**

This input tells the code what has to be done. There are currently 4 possibilities:

  - *SYNTHESIS*
    It uses the model atmosphere supplied to solve the radiative transfer equation and get the synthetic Stokes spectra. In this mode, *FILEPROFILE:* is mandatory.
  - *INVERSION*
    It uses the model atmosphere supplied and the profiles supplied to retrieve the model atmosphere that best fits the provided Stokes spectra. In this mode, *FILEPROFILE:* and *INVERSION SETUP:* are mandatory.

[1]Notice that either gas pressure or density are modified so that the equation of state is fulfilled at every height.

– *TAU*
It uses the provided model atmosphere (*FILEMODEL:*) to estimate the $\lg \tau_5$ scale (optical depth) from the supplied temperature and gas pressure or density. Under this mode FIRTEZ-dz does not solve the radiative transfer equation, therefore no Stokes vector under *FILEPROFILE:* is needed.

– *HYDROSTATIC*
Under this mode, FIRTEZ-dz takes the value of the gas pressure at the uppermost z-slab, $P_{\rm g}(z_{\rm max})$, of the provided atmosphere (*FILEMODEL:*) and uses it as a boundary condition to determine the gas pressure elsewhere by solving downwards the equation of hydrostatic equilibrium. It requires that *HYDROSTATIC:* field is *YES*. Under this mode FIRTEZ-dz does not solve the radiative transfer equation, therefore no Stokes vector under *FILEPROFILE:* is needed.

- *STOKES SETUP:* **Optional (default: all active)**

It is useless except for *MODE: SYNTHESIS* or *MODE: INVERSION*. It specifies the Stokes parameters to be synthesis/inverted and, if *MODE:INVERSION*, also specifies the inversion weights. The *STOKES SETUP:* field can include up to 4 lines, with any/all of the following: *STKI*, *STKQ*, *STKU*, and/or *STKV*. By default all of them are synthesised/inverted and, if *MODE:INVERSION*, the default weights are set to 1. For example, if one wants to synthesis Stokes I and V for a given model atmosphere, then *STOKES SETUP:* field should appear as follows:

Input file Ex. 2.6: *Stokes setup (1)*

```
...
MODE:
SYNTHESIS
STOKES SETUP:
STKI
STKV
...
```

And if for instance, what we want is to invert all the Stokes parameters with weights *1.0*, *1.0*, *2.0*, *2.0* for Stokes I, V, Q, and U, respectively we would write:

Input file Ex. 2.7: *Stokes setup (2)*

```
...
MODE:
INVERSION
STOKES SETUP:
STKI 1.0
STKV 1.0
STKQ 2.0
STKU 2.0
...
```

where the values *1.0* for *STKI* and *STKV* can be omitted because they are the default values. We note that weights appear quadratically in the definition of χ^2:

$$\chi^2 \propto \sum_{I,Q,U,V} \frac{\sum_\lambda (O_\lambda - I_\lambda^{syn})^2 * w_{I,Q,U,V}^2}{\sigma_{I,Q,U,V}^2} \tag{1}$$

- *INVERSION SETUP:* **Mandatory** if *MODE: INVERSION*, otherwise **Inactive**

This field is mandatory for *INVERSION* mode. It can take any or almost all of the combinations between: *TEM*, *P0*, *PGAS*, *RHO*, *BX*, *BY*, *BZ*, and/or *VLOS*. It is NOT possible to set *TEM*, *PGAS*, and *RHO* at the same time since it might lead to combination of thermodynamical parameters that do not fulfil the equation of state. It is also not possible to invert at the same time *P0* with *PGAS* or *RHO*, since the former requires of assuming hydrostatic equilibrium. If *P0* is active with *PGAS*, *RHO*, or both, execution will stop. If *P0* is active and *HYDROSTATIC* is *NO* the code will stop execution. For the inversion, by default, the code will assume that one wants to invert all the available slabs. It is possible to limit the number of perturbations to be calculated by physical parameter by adding an accompanying number to each physical parameter.

For example, let say we want to invert the temperature, the three magnetic field components and line-of-sight velocity for a given profile. But, we want to let all the temperatures slabs as free parameters, only perturb each of the three magnetic field components at two points and the line-of-sight velocity at five points, then we should supply the following *INVERSION SETUP* field:

Input file Ex. 2.8: *Inversion setup*

```
...
INVERSION SETUP:
TEM
BX 2
BY 2
BZ 2
VLOS 5
...
```

Also it is possible to use *INVERSION SETUP:* field with *SYNTHESIS* mode when one wants to write down the response functions for a model atmosphere. For this only case, it is possible to set at the same time all of the following options: *TEM*, *PGAS*, *RHO*, *BX*, *BY*, *BZ*, and/or *VLOS*.

- *CONTINUUM NORMALIZATION:* **Optional (default: active)**
  Optional field, useful for *SYNTHESIS* and *INVERSION* modes. Specifies whether continuum normalization to HSRA at disk center must be set or not. Default is *TRUE*. To disable it, one may use:

Input file Ex. 2.9: *Continuum normalization*

```
...
CONTINUUM NORMALIZATION:
FALSE
END:
...
```

- *COUPLED INVERSION:* **Optional (default: inactive)**
  Optional field only valid in *INVERSION* mode. This field is used for the inversion using Michiel van Noort's two-dimensional coupled inversion (van Noort, M. 2012). It requires at least one line, though some additional lines might be included in this field. The mandatory line specifies the file to contain the two-dimensional point-spread-function file name and should be written as follows:

Input file Ex. 2.10: *Coupled inversion (1)*

```
...
COUPLED INVERSION:
FILENAME psf2d.bin
FILEPROFILE:
...
```

In addition to that, there might be three additional input lines specifying the size of the block to consider (*BLCKSZ*), the number of threads to use for the matrix inversion (*NMTHRD*), or/and the radius (in pixels) of the point-spread-function to consider during the inversion *PSFRAD*.

Input file Ex. 2.11: *Coupled inversion (2)*

```
...
COUPLED INVERSION:
FILENAME psf2d.bin
BLCKSZ 20
NMTHRD 5
PSFRAD 6
FILEPROFILE:
...
```

If any of the optional lines are missing, default values are taken: $BLCKSZ = 10$, $NMTHRD =$ as many as MPI slaves, and $PSFRAD = 3$. Be aware that since these values depend on the point-spread-function used, default values might not be optimal for your case.

- *WRITE RESPONSE FUNCTION:* **Optional (default: inactive)**
  This is only useful for the *SYNTHESIS* mode. It goes in combination with *INVERSION SETUP:* and it can be either *NO* (default) or *YES*. This option MUST be used very carefully since it writes down files that can be extremely huge. For instance, if we are writing down the response functions for all the available physical parameters of a model atmosphere with 512 times 512 times 492 points at 100 spectral points, we will require more than 670 GB!. If one really wants to write down the response functions (provided that *MODE* is set to *SYNTHESIS*) then input file must contain:

Input file Ex. 2.12: *Write response function*

```
...
WRITE RESPONSE FUNCTION:
YES
MODELNAME:
...
```

And in addition, one can specify in *INVERSION SETUP:* for what physical parameters response functions must be calculated. For an example on how to use this field have a look at 2.20

- *LINE SPREAD FUNCTION:* **Optional (default: inactive)**
  This field is optional and has effect only in *SYNTHESIS* and *INVERSION* mode. It specifies the gaussian function to use to simulate the effect of the instrumental transmission function. It has to be specified for each of the spectral ranges considered. The format to provide this input parameter is:

ss ww,
where *ss* is the standard deviation of the instrumental line-spread-function (in mÅ) and *ww* is the offset from the zero wavelength reference (in mÅ). Following the example shown in Ex. 2.1, we would like to set an instrumental line-spread-function of Gaussian shape with a standard deviation of *3.5* for the first two spectral ranges (starting spectral line index *4*) and *7.6* for the last one (spectral line index *9*), and additionally, the offset for the middle spectral region is going to be 1.4 mÅ:

Input file Ex. 2.13: *Line spread function*

```
...
LINE SPREAD FUNCTION:
3.5 0.
3.5 1.4
7.6 0
CONTINUUM NORMALIZATION:
...
```

It is important to note that there MUST be the exactly same number of inputs as for the *LINES* field, otherwise, the code will stop.

- *MISC SETUP:* **Optional (default: see below)**
  In addition to the previous inputs, there are some additional possibilities:

  1. *SVDTOL* **Optional (default:** 10^{-4}**)**
     Used in *INVERSION* mode. It defines the threshold used for the number of eigen-values used during the inversion of the dampened Hessian matrix.
  2. *NOISE* **Optional (default:** 10^{-4}**)**
     Used in *INVERSION* mode. It defines the standard deviation/noise of the data to be used during the inversion and error calculation.
  3. *TEMP* **Optional (default: Inactive)**
     Used in *INVERSION* mode. It pre-calculates an initial temperature perturbation based on the average intensity of the first spectral region provided.
  4. *PGAS* **Optional (default: Inactive)**
     Used in *INVERSION* mode. It pre-calculates an initial gas pressure perturbation so that $\lg \tau_5$ is formed inside the provided height range.
  5. *BVEC* **Optional (default: Inactive)**
     Used in *INVERSION* mode. It pre-calculates an initial magnetic field vector perturbation based on the four Stokes spectra of the first spectral region provided (Only works if inverting the four Stokes parameters, otherwise ignored).
  6. *VLOS* **Optional (default: Inactive)**
     Used in *INVERSION* mode. It pre-calculates an initial line-of-sight velocity component perturbation based on the Stokes I spectra of the first spectral region provided (Only works if Stokes I is inverted, otherwise ignored).

7. *FULL_STOKES* **Optional (default: Inactive)**
   Used in *INVERSION* and *SYNTHESIS* mode. It stores (and writes down) the Stokes parameters for the whole three-dimensional volume considered. This option may considerably increase the memory requirement.
8. *SLDPATH* **Optional (default: ./)**
   It specifies the directory where the file with the spectral line data is.
9. *MODPATH* **Optional (default: ./)**
   It specifies the directory where the file with the model atmosphere is.
10. *DATAPATH* **Optional (default: ./)**
    It specifies the directory where the file with the stokes spectra is (only useful in inversion mode).
11. *OUTPATH* **Optional (default: ./)**
    It specifies the directory where the output data will be written, all of it, i.e. the model atmosphere, the synthetic profiles, the response functions (if required).

- *END:*
  It specifies the end of the input file that must be read by the code. Anything below is ignored.

2.2 Input file examples:

These examples are provided in the *tests* directory together with the standard release.

In the first one, we just re-calculate the gas pressure and density for a model atmosphere:

Input file Ex. 2.14: *Example 1: Get gas pressure and density in HE*

```
FILEMODEL:
test_atmos.bin
MODE:
hydrostatic
HYDROSTATIC:
yes
MISC SETUP:
MODPATH ../../default/
SLDPATH ../../default/
END:
```

In this second example, we calculate the optical depth scale at 5000 Å for the input model atmosphere used in the previous example, i.e. without the assumption of hydrostatic equilibrium.

Input file Ex. 2.15: *Example 2: Get optical depth*

```
FILEMODEL:
test_atmos.bin
MODE:
tau
HYDROSTATIC:
no
MISC SETUP:
MODPATH ../../default/
SLDPATH ../../default/
END:
```

Now, we combine the two previous cases calculating the optical depth scale of a model atmosphere whose gas pressure and density are previously calculated assuming hydrostatic equilibrium.

Input file Ex. 2.16: *Example 3: Get optical depth, HE*

```
FILEMODEL:
test_atmos.bin
MODE:
tau
HYDROSTATIC:
yes
MISC SETUP:
MODPATH ../../default/
SLDPATH ../../default/
END:
```

Now we consider the case of synthesising some Stokes profiles. To do so, we are going to synthesis the full Stokes vector for the blended lines 6301.5 and 6302.5 Å.

Input file Ex. 2.17: *Example 4: Synthesis*

```
LINES:
4,5 500 -700.0 5.0
FILEPROFILE:
syn_profile.bin
FILEMODEL:
test_atmos.bin
MODE:
synthesis
HYDROSTATIC:
no
STOKES SETUP:
STKI
STKQ
STKU
STKV
MISC SETUP:
MODPATH ../../default/
SLDPATH ../../default/
OUTPATH ./out_dir/
END:
```

Now we proceed exactly on the same way, but assuming hydrostatic equilibrium.

Input file Ex. 2.18: *Example 5: Synthesis, HE*

```
LINES:
4,5 500 -700.0 5.0
FILEPROFILE:
syn_profile.bin
FILEMODEL:
test_atmos.bin
MODE:
synthesis
HYDROSTATIC:
yes
MISC SETUP:
MODPATH ../../default/
SLDPATH ../../default/
OUTPATH ./out_dir/
END:
```

Note here that we do not specify any Stokes parameter to be synthesis as by default all of them are calculated.

In the following example 2.19 we consider the inversion of all the Stokes profiles for the temperature and the LOS velocity:

Input file Ex. 2.19: *Example 6: Inversion*

```
LINES:
4,5 500 -700.0 5.0
FILEPROFILE:
syn_profile.bin
FILEMODEL:
test_atmos.bin
MODE:
inversion
HYDROSTATIC:
yes
STOKES SETUP:
STKI
STKQ
STKU
STKV
INVERSION SETUP:
TEM
VLOS
MISC SETUP:
DATAPATH ../test_syn1/out_dir/
MODPATH ../../default/
SLDPATH ../../default/
OUTPATH ./out_dir/
END:
```

The result of the inversion is stores in two files. For the model, the output will be written in the *OUTPATH* directory with the same name as the input one preceded by the suffix: "out_" (i.e. in this case, it would be written in "out_dir" with the name: "out_test_atmos.bin"). Similarly, the synthetic profiles of the fitting, will be stored in the same directory (*OUTPATH*) and the name is the same as the input profiles with the additional "out_" suffix (in this example: "out_syn_profile.bin").

In order to get the response functions, one has to set *MODE:* to *SYNTHESIS* and *WRITE RESPONSE FUNCTION:* to *YES*. Additionally, we have to include *INVERSION SETUP:* in order to tell the code to what physical parameter we want the response functions to. In the example below, we would get the response functions to temperature (*TEM*), to LOS velocity (*VLOS*), and to the z component of the magnetic field vector (*BZ*).//

Input file Ex. 2.20: *Example 7: Response functions*

```
LINES:
4,5 500 -700.0 5.0
FILEPROFILE:
syn_profile.bin
FILEMODEL:
test_atmos.bin
MODE:
synthesis
WRITE RESPONSE FUNCTION:
YES
INVERSION SETUP:
TEM
VLOS
BZ
MISC SETUP:
DATAPATH ../test_syn1/out_dir/
MODPATH ../../default/
SLDPATH ../../default/
OUTPATH ./out_dir/
END:
```

Now that we have covered the input file, let us have a look at the spectral lines database file.

2.3 Spectral lines database file:

Spectral line database is the file in which spectral lines atomic data is supplied to the code. It is very similar to SIR code LINES file, with some differences though. Each row specifies the data for a given atomic transition and the various columns are:

1. Index: An integer number identifying each atomic data entry. It can be any integer.
2. Atomic number (Z).
3. Ionisation stage: Only lines in neutral, once, and twice ionization stages are allowed. Adimensional.
4. Central wavelength of the transition: float number, in angstrom.
5. Lower energy level: the format is as follows: X.XCY.Y, where X.X refers to the spin momentum, C refers to the angular momentum, and Y.Y to the J quantum number of the lower level.
6. Upper energy level: the format is as follows: X.XCY.Y, where X.X refers to the spin momentum, C refers to the angular momentum, and Y.Y to the J quantum number of the upper level.

7. Oscillator strength ($\log gf$): real number (might be positive or negative)
8. Ionization energy (in eV) of the lower level: real number
9. Barklem collisional theory velocity parameter (α): real number, adimensional.
10. Barklem collisional theory cross-section: real number in atomic units.
11. Barklem collisional theory transition type: two character string.
12. Barklem collisional theory lower level energy limit: real number in cm^{-1}.
13. Barklem collisional theory upper level energy limit: real number in cm^{-1}.

Some comments on these last five parameters (the ones related to ABO collisional theory) are required. If the user provides the transition type and the other are set to 0.0, then the tables published by Barklem and co-authors are interpolated to the effective principal quantum number (n^*). If the user supplies the velocity parameter (α) and the cross-section and the last two fields are 0.0, then, independently of the transition type supplied (yet two string characters must be supplied), the values provided for the velocity parameter (α) and the cross-section are used. Finally, if the last two (lower and upper energy limits) are non-zero, transition type parameter is MANDATORY. In this case, the last two parameters (lower and upper energy limits) are used to calculate the effective principal quantum number (n^*) and then, using the transition type parameter, the ABO tables are interpolated accordingly. The values for the velocity parameter (α) and cross-section for each entry of the spectral line database is stored in the output log file *info.log*.

Let use FeI 6173.3 Å spectral line as an example.
The first possibility might be to provide the transition type, i.e. *sp* as:

Spectral line database Ex. 2.1: *Spectral line database ex. 1*

```
72 026 1 6173.3354 5.0P1.0 5.0D0.0 -2.880 2.223 0.000 0000.0 sp 0.0 0.0
```

If one does so, one would retrieve a velocity parameter (α) of 0.2406 and a cross-section value of 327.2273 and the values are wrong. It is so because for the energy levels involved in this transition the energy limit and the ionization energy are not the same. In this case, there are two possibilities: 1- We know the values for the velocity parameter (α) and the cross-section:

Spectral line database Ex. 2.2: *Spectral line database ex. 2*

```
75 026 1 6173.3354 5.0P1.0 5.0D0.0 -2.880 2.223 0.2660 280.5707 aa 0.0 0.0
```

In this case, the three last parameters are ignored by the code. Or, 2- we calculate the energy limit of each level involved in the transition (in cm^{-1}):

Spectral line database Ex. 2.3: *Spectral line database ex. 3*

```
73 026 1 6173.3354 5.0P1.0 5.0D0.0 -2.880 2.223 0.000 0000.0 sp 77212.151 65610.304
```

As you can check at the beginning of *info.log* the "Collisional cross section" and "Velocity parameter" are the same as specified in Ex. 2.2. These last two cases are correct, yet the first one leads to incorrect results.