PyNeb_manual7

May 5, 2017

```
In [1]: %matplotlib inline
        import pyneb as pn
```

0.1 Exploring line intensities with the EmisGrid class

0.1.1 The EmisGrid class

Most plots are obtained by operating on emission maps, which are grids of emissivities as a function of temperature and density generated by the **EmisGrid** class.

EmisGrid instantiates an atom and computes the emissivities of all its lines for the (tem, den) values of a regularly spaced grid (may be log or linear in the case of the density). Each line is represented in a 2D array (a grid), and there are as many arrays transitions in the atom. The results can be operated on, saved for a later use in a cPickle file, or restored.

The following command instantiates an [O III] atom and computes the emissivity of all its lines in a 30x30 grid:

```
In [2]: O3_EG = pn.EmisGrid('O', 3, n_tem=30, n_den=30)
```

The arguments are described in more details in the Reference Manual. Here is the list:

```
In [3]: O3_EG = pn.EmisGrid(elem='O', spec=3, n_tem=100, n_den=100,
                            tem_min=5000., tem_max=20000., den_min=10.,
                            den_max=1.e8, restore_file=None, atomObj=None)
```

The emissivity grid of a specific line can be obtained by means of:

```
In [4]: O3_5007 = O3_EG.getGrid(wave=5007)
```

```
In [5]: O3_5007.shape
```

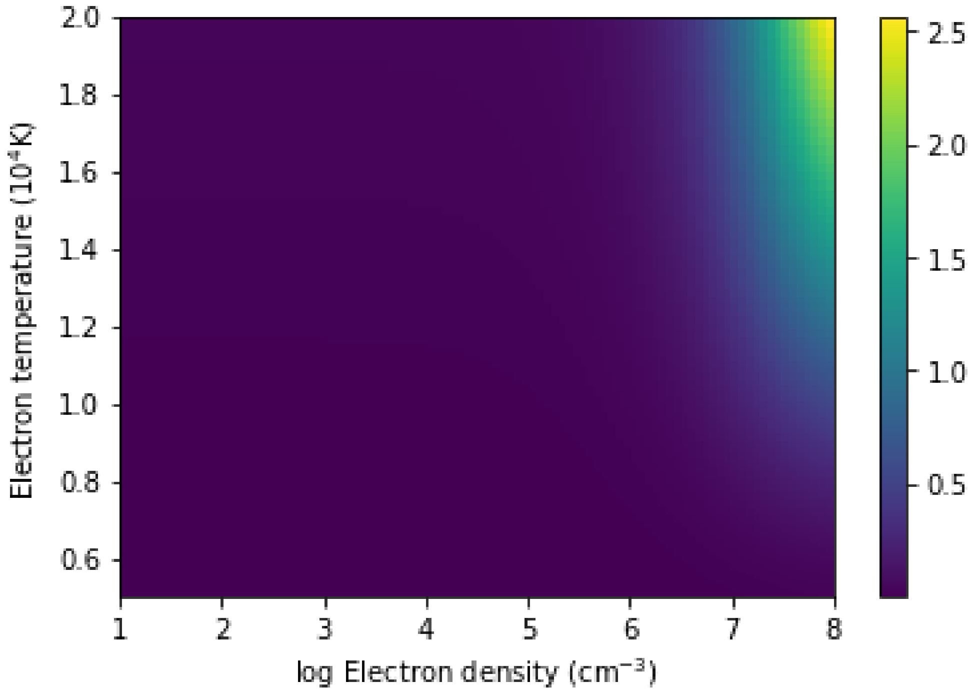
```
Out[5]: (100, 100)
```

The emissivity grid of a combination of lines can also be computed:

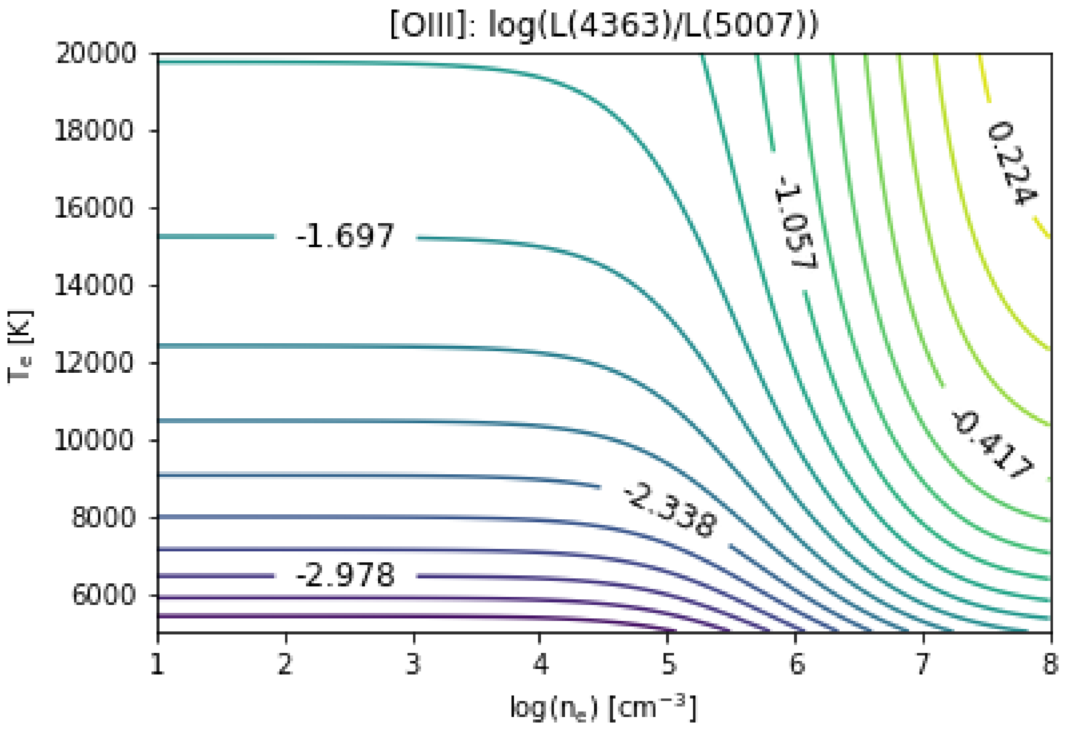
```
In [6]: O3_Te = O3_EG.getGrid(to_eval = 'L(4363)/L(5007)')
```

There are two plotting tools integrated in the **EmisGrid** object:

```
In [7]: O3_EG.plotImage(to_eval = 'L(4363)/L(5007)')
```



```
In [8]: O3_EG.plotContours(to_eval = 'L(4363)/L(5007)')
```



See the Reference Manual for more option and the **Diagnostic** class for producing plots combining different atoms.

0.1.2 Instantiating various EmisGrid objects with getEmisGridDict

It is quite common to have to instantiate various **EmisGrid** objects, especially if you want to make a diagnostic diagram. This can easily be done using the **getEmisGridDict** method, used for example as follows:

```
In [10]: emisgrids = pn.getEmisGridDict(atom_list=['O2', 'O3', 'N2'])
```

```
warng getEmisGridDict: Emission map not found: ./pypics//emis_O2.pypic
warng getEmisGridDict: Emission map not found: ./pypics//emis_O3.pypic
warng getEmisGridDict: Emission map not found: ./pypics//emis_N2.pypic
```

```
In [11]: emisgrids
```

```
Out[11]: {'N2': <pyneb.core.emisGrid.EmisGrid at 0x7fa4b374fe10>,
          'O2': <pyneb.core.emisGrid.EmisGrid at 0x7fa4b374fba8>,
          'O3': <pyneb.core.emisGrid.EmisGrid at 0x7fa4b370d710>}
```

This command generates a dictionary of emission grids for [O II], [O III] and [N II]. The resulting maps are saved in a directory defined by default when PyNeb is started, in the pn.config.pypic_path variable. It first tries to use the $HOME/.pypics directory; if it fails, it tries to use /tmp/pypic; if it fails too, the value is set to None and a user-defined value has to be provided by changing pn.config.pypic_path or using the pypic_path parameter when calling getEmisGridDict.

If a Diagnostic object is already available (see next Section), it can be used to determine the relevant atoms for which a grid must be computed or restored:

```
In [12]: diags = pn.Diagnostics() # See next section
         emisgrids = pn.getEmisGridDict(atomDict=diags.atomDict, den_max=1e6)
```

This **EmisGrid** dictionary will be very useful to plot diagnostic diagrams with the **Diagnostic** object, as is described in the next section.

0.2 The Diagnostics class

Diagnostics is the class used to evaluate temperatures and densities from line ratios. It is also the class that plots the diagnostic Te-Ne diagrams. The object is instantiated like this:

```
In [13]: diags = pn.Diagnostics()         # instantiate the Diagnostic class
```

An optional parameter **addAll=True** (default is **False**) lets the object load all the available diagnostics. Most of the time this option is not used and the diagnostics are added manually as they are needed:

```
In [14]: diags.addDiag(['[NI] 5198/5200','[NII] 5755/6548','[OII] 3726/3729'])
```

Each diagnostic is defined by a label and is associated to a tuple containing 3 elements: the atom corresponding to the diagnostic lines, the algebraic definition of the line ratios and the algebraic definition of the error of the diagnostic, which depends on the error of each line involved. In the present case, the diagnostic is the ratios of two [O III] lines, 4363/5007, and the error is the quadratic sum of the relative error of each line (E(lambda)): RMS(a, b) = sqrt(a**2** + **b**2).

Users can also define their own diagnostics, for example using:

```
In [15]: diags.addDiag('[OIII] 4363/4959', ('O3', 'L(4363)/L(4959)', 'RMS([E(4363),E(4959)])'))
```

Notice that the diagnostics are defined so that they tend to increase with the main parameter they trace: [OIII] 4363/5007 increases with the electron temperature.

The diagnostics contained in a **Diagnostics** object are listed by means of the **diags.getDiagLabels()** and **diags.getDiags()** methods. Once added to the **Diagnostics** object, they can be used either to compute **Te** and **Ne** via **getCrossTemDen** or to plot diagrams (see below). A diagnostic can be removed from the list with the **delDiag** method.

0.2.1 Determination of temperature and density

The **getCrossTemDen** method cross-converges the temperature and density derived from two sensitive line ratios (not necessarily from the same atom), by inputting the quantity derived with one line ratio into the other and then iterating. When the iteration process ends, the two diagnostics give self-consistent results. The first line ratio must be a temperature-sensitive one and the second a density-sensitive one. The temperature and density can be individual numbers as well as arrays (provided they are equal in shape).

```
In [16]: diags.getCrossTemDen('[NII] 5755/6548', '[SII] 6731/6716', 0.02, 1.0)
```

```
Out[16]: (7612.554292918343, 444.7122084104995)
```

```
In [17]: for diag in sorted(pn.diags_dict.keys()):
             print('"{0}" : {1}'.format(diag, pn.diags_dict[diag]))
```

```
"[ArIII] (7751+7136)/9m" : ('Ar3', '(L(7751)+L(7136))/L(90000)', 'RMS([E(90000), E(7751)*L(7751)/(L(775
"[ArIII] 5192/7136" : ('Ar3', 'L(5192)/L(7136)', 'RMS([E(7136), E(5192)])')
"[ArIII] 5192/7300+" : ('Ar3', 'L(5192)/(L(7751)+L(7136))', 'RMS([E(7751)*L(7751)/(L(7751)+L(7136)), E(7
"[ArIII] 7136/9m" : ('Ar3', 'L(7136)/L(90000)', 'RMS([E(90000), E(7136)])')
"[ArIII] 9.0m/21.8m" : ('Ar3', 'L(89897)/L(218000)', 'RMS([E(89897), E(218000)])')
"[ArIV] 2860+/4720+" : ('Ar4', '(L(2854)+L(2868))/(L(4711)+L(4740))', 'RMS([E(4711)*L(4711)/(L(4711)+L(4
"[ArIV] 4740/4711" : ('Ar4', 'L(4740)/L(4711)', 'RMS([E(4711), E(4740)])')
"[ArIV] 7230+/4720+" : ('Ar4', '(L(7170)+L(7263))/(L(4711)+L(4740))', 'RMS([E(4711)*L(4711)/(L(4711)+L(4
"[ArV] 4626/6600+" : ('Ar5', 'L(4626)/(L(6435)+L(7005))', 'RMS([E(6435)*L(6435)/(L(6435)+L(7005)), E(700
"[CIII] 1909/1907" : ('C3', 'L(1909)/L(1907)', 'RMS([E(1909), E(1907)])')
"[ClIII] 5538/5518" : ('Cl3', 'L(5538)/L(5518)', 'RMS([E(5518), E(5538)])')
"[ClIV] 5323/7531" : ('Cl4', 'L(5323)/L(7531)', 'RMS([E(7531), E(5323)])')
"[ClIV] 5323/7700+" : ('Cl4', 'L(5323)/(L(7531)+L(8046))', 'RMS([E(7531)*L(7531)/(L(7531)+L(8046)), E(80
"[FeIII] 4659/4009" : ('Fe3', 'L(4659)/L(4009)', 'RMS([E(4659), E(4009)])')
"[FeIII] 4659/4701" : ('Fe3', 'L(4659)/L(4701)', 'RMS([E(4659), E(4701)])')
"[FeIII] 4659/4734" : ('Fe3', 'L(4659)/L(4734)', 'RMS([E(4659), E(4734)])')
"[FeIII] 4701/4009" : ('Fe3', 'L(4701)/L(4009)', 'RMS([E(4701), E(4009)])')
"[FeIII] 4701/4734" : ('Fe3', 'L(4701)/L(4734)', 'RMS([E(4701), E(4734)])')
"[FeIII] 4734/4009" : ('Fe3', 'L(4734)/L(4009)', 'RMS([E(4734), E(4009)])')
"[FeIII] 4881/4009" : ('Fe3', 'L(4881)/L(4009)', 'RMS([E(4881), E(4009)])')
"[FeIII] 4881/4659" : ('Fe3', 'L(4881)/L(4659)', 'RMS([E(4881), E(4659)])')
"[FeIII] 4881/4701" : ('Fe3', 'L(4881)/L(4701)', 'RMS([E(4881), E(4701)])')
"[FeIII] 4881/4734" : ('Fe3', 'L(4881)/L(4734)', 'RMS([E(4881), E(4734)])')
"[FeIII] 4881/4931" : ('Fe3', 'L(4881)/L(4931)', 'RMS([E(4881), E(4931)])')
"[FeIII] 4881/5011" : ('Fe3', 'L(4881)/L(5011)', 'RMS([E(4881), E(5011)])')
"[FeIII] 4925/4009" : ('Fe3', 'L(4925)/L(4009)', 'RMS([E(4925), E(4009)])')
"[FeIII] 4925/4659" : ('Fe3', 'L(4925)/L(4659)', 'RMS([E(4925), E(4659)])')
"[FeIII] 4925/4701" : ('Fe3', 'L(4925)/L(4701)', 'RMS([E(4925), E(4701)])')
"[FeIII] 4925/4734" : ('Fe3', 'L(4925)/L(4734)', 'RMS([E(4925), E(4734)])')
"[FeIII] 4925/4881" : ('Fe3', 'L(4925)/L(4881)', 'RMS([E(4925), E(4881)])')
"[FeIII] 4925/4931" : ('Fe3', 'L(4925)/L(4931)', 'RMS([E(4925), E(4931)])')
"[FeIII] 4925/5011" : ('Fe3', 'L(4925)/L(5011)', 'RMS([E(4925), E(5011)])')
"[FeIII] 4931/4009" : ('Fe3', 'L(4931)/L(4009)', 'RMS([E(4931), E(4009)])')
"[FeIII] 4931/4659" : ('Fe3', 'L(4931)/L(4659)', 'RMS([E(4931), E(4659)])')
"[FeIII] 4931/4701" : ('Fe3', 'L(4931)/L(4701)', 'RMS([E(4931), E(4701)])')
"[FeIII] 4931/4734" : ('Fe3', 'L(4931)/L(4734)', 'RMS([E(4931), E(4734)])')
"[FeIII] 4987/4009" : ('Fe3', 'L(4987)/L(4009)', 'RMS([E(4987), E(4009)])')
"[FeIII] 4987/4659" : ('Fe3', 'L(4987)/L(4659)', 'RMS([E(4987), E(4659)])')
"[FeIII] 4987/4701" : ('Fe3', 'L(4987)/L(4701)', 'RMS([E(4987), E(4701)])')
"[FeIII] 4987/4734" : ('Fe3', 'L(4987)/L(4734)', 'RMS([E(4987), E(4734)])')
"[FeIII] 4987/4881" : ('Fe3', 'L(4987)/L(4881)', 'RMS([E(4987), E(4881)])')
```

```
"[FeIII] 4987/4925" : ('Fe3', 'L(4987)/L(4925)', 'RMS([E(4987), E(4925)])')
"[FeIII] 4987/4931" : ('Fe3', 'L(4987)/L(4931)', 'RMS([E(4987), E(4931)])')
"[FeIII] 4987/5011" : ('Fe3', 'L(4987)/L(5011)', 'RMS([E(4987), E(5011)])')
"[FeIII] 5011/4009" : ('Fe3', 'L(5011)/L(4009)', 'RMS([E(5011), E(4009)])')
"[FeIII] 5011/4659" : ('Fe3', 'L(5011)/L(4659)', 'RMS([E(5011), E(4659)])')
"[FeIII] 5011/4701" : ('Fe3', 'L(5011)/L(4701)', 'RMS([E(5011), E(4701)])')
"[FeIII] 5011/4734" : ('Fe3', 'L(5011)/L(4734)', 'RMS([E(5011), E(4734)])')
"[FeIII] 5011/4931" : ('Fe3', 'L(5011)/L(4931)', 'RMS([E(5011), E(4931)])')
"[FeIII] 5270/4009" : ('Fe3', 'L(5270)/L(4009)', 'RMS([E(5270), E(4009)])')
"[FeIII] 5270/4659" : ('Fe3', 'L(5270)/L(4659)', 'RMS([E(5270), E(4659)])')
"[FeIII] 5270/4701" : ('Fe3', 'L(5270)/L(4701)', 'RMS([E(5270), E(4701)])')
"[FeIII] 5270/4734" : ('Fe3', 'L(5270)/L(4734)', 'RMS([E(5270), E(4734)])')
"[FeIII] 5270/4881" : ('Fe3', 'L(5270)/L(4881)', 'RMS([E(5270), E(4881)])')
"[FeIII] 5270/4925" : ('Fe3', 'L(5270)/L(4925)', 'RMS([E(5270), E(4925)])')
"[FeIII] 5270/4931" : ('Fe3', 'L(5270)/L(4931)', 'RMS([E(5270), E(4931)])')
"[FeIII] 5270/4987" : ('Fe3', 'L(5270)/L(4987)', 'RMS([E(5270), E(4987)])')
"[FeIII] 5270/5011" : ('Fe3', 'L(5270)/L(5011)', 'RMS([E(5270), E(5011)])')
"[NIII] 1751+/57.4m" : ('N3', 'B("1751A+")/L(574000)', 'RMS([E(574000), BE("1751A+")])')
"[NII] 121m/20.5m" : ('N2', 'L(1214747)/L(2054427)', 'RMS([E(2054427)/E(1214747)])')
"[NII] 5755/6548" : ('N2', 'L(5755)/L(6548)', 'RMS([E(6548), E(5755)])')
"[NII] 5755/6584" : ('N2', 'L(5755)/L(6584)', 'RMS([E(6584), E(5755)])')
"[NII] 5755/6584+" : ('N2', 'L(5755)/(L(6548)+L(6584))', 'RMS([E(6548)*L(6548)/(L(6548)+L(6584)), E(658
"[NI] 5198/5200" : ('N1', 'I(3, 1)/I(2, 1)', 'RMS([E(5200), E(5198)])')
"[NeIII] 15.6m/36.0m" : ('Ne3', 'L(156000)/L(360000)', 'RMS([E(156000), E(360000)])')
"[NeIII] 3343/3930+" : ('Ne3', 'L(3343)/(L(3869)+L(3968))', 'RMS([E(3869)*L(3869)/(L(3869)+L(3968)), E(
"[NeIII] 3344/3930+" : ('Ne3', 'L(3343)/(L(3869)+L(3968))', 'RMS([E(3869)*L(3869)/(L(3869)+L(3968)), E(
"[NeIII] 3869/15.6m" : ('Ne3', 'L(3869)/L(156000)', 'RMS([E(156000), E(3869)])')
"[NeIII] 3930+/15.6m" : ('Ne3', '(L(3869)+L(3968))/L(156000)', 'RMS([E(156000), E(3869)*L(3869)/(L(3869)
"[NeV] 14.3m/24.2m" : ('Ne5', 'L(143000)/L(242000)', 'RMS([E(143000), E(242000)])')
"[NeV] 1575/3426" : ('Ne5', 'L(1575)/L(3426)', 'RMS([E(1575), E(3426)])')
"[NeV] 2973/3370+" : ('Ne5', 'L(2973)/(L(3426)+L(3346))', 'RMS([E(3426)*L(3426)/(L(3426)+L(3346)), E(33
"[NeV] 3426/24.2m" : ('Ne5', 'L(3426)/L(242000)', 'RMS([E(3426), E(242000)])')
"[NiIII] 6000/6401" : ('Ni3', 'L(6000)/L(6401)', 'RMS([E(6000), E(6401)])')
"[NiIII] 6000/6682" : ('Ni3', 'L(6000)/L(6682)', 'RMS([E(6000), E(6682)])')
"[NiIII] 6000/6797" : ('Ni3', 'L(6000)/L(6797)', 'RMS([E(6000), E(6797)])')
"[NiIII] 6000/7125" : ('Ni3', 'L(6000)/L(7125)', 'RMS([E(6000), E(7125)])')
"[NiIII] 6000/7890" : ('Ni3', 'L(6000)/L(7890)', 'RMS([E(6000), E(7890)])')
"[NiIII] 6000/8500" : ('Ni3', 'L(6000)/L(8500)', 'RMS([E(6000), E(8500)])')
"[NiIII] 6401/7125" : ('Ni3', 'L(6401)/L(7125)', 'RMS([E(6401), E(7125)])')
"[NiIII] 6401/7890" : ('Ni3', 'L(6401)/L(7890)', 'RMS([E(6401), E(7890)])')
"[NiIII] 6401/8500" : ('Ni3', 'L(6401)/L(8500)', 'RMS([E(6401), E(8500)])')
"[NiIII] 6534/6401" : ('Ni3', 'L(6534)/L(6401)', 'RMS([E(6534), E(6401)])')
"[NiIII] 6534/6682" : ('Ni3', 'L(6534)/L(6682)', 'RMS([E(6534), E(6682)])')
"[NiIII] 6534/6797" : ('Ni3', 'L(6534)/L(6797)', 'RMS([E(6534), E(6797)])')
"[NiIII] 6534/7125" : ('Ni3', 'L(6534)/L(7125)', 'RMS([E(6534), E(7125)])')
"[NiIII] 6534/7890" : ('Ni3', 'L(6534)/L(7890)', 'RMS([E(6534), E(7890)])')
"[NiIII] 6534/8500" : ('Ni3', 'L(6534)/L(8500)', 'RMS([E(6534), E(8500)])')
"[NiIII] 6682/7125" : ('Ni3', 'L(6682)/L(7125)', 'RMS([E(6682), E(7125)])')
"[NiIII] 6682/7890" : ('Ni3', 'L(6682)/L(7890)', 'RMS([E(6682), E(7890)])')
"[NiIII] 6682/8500" : ('Ni3', 'L(6682)/L(8500)', 'RMS([E(6682), E(8500)])')
"[NiIII] 6797/7125" : ('Ni3', 'L(6797)/L(7125)', 'RMS([E(6797), E(7125)])')
"[NiIII] 6797/7890" : ('Ni3', 'L(6797)/L(7890)', 'RMS([E(6797), E(7890)])')
"[NiIII] 6797/8500" : ('Ni3', 'L(6797)/L(8500)', 'RMS([E(6797), E(8500)])')
"[NiIII] 6946/6401" : ('Ni3', 'L(6946)/L(6401)', 'RMS([E(6946), E(6401)])')
```

```
"[NiIII] 6946/6682" : ('Ni3', 'L(6946)/L(6682)', 'RMS([E(6946), E(6682)])')
"[NiIII] 6946/6797" : ('Ni3', 'L(6946)/L(6797)', 'RMS([E(6946), E(6797)])')
"[NiIII] 6946/7125" : ('Ni3', 'L(6946)/L(7125)', 'RMS([E(6946), E(7125)])')
"[NiIII] 6946/7890" : ('Ni3', 'L(6946)/L(7890)', 'RMS([E(6946), E(7890)])')
"[NiIII] 6946/8500" : ('Ni3', 'L(6946)/L(8500)', 'RMS([E(6946), E(8500)])')
"[OIII] 1664+/5007" : ('O3', '(B("1664A+"))/L(5007)', 'RMS([BE("1664A+"), E(5007)])')
"[OIII] 1666/4363" : ('O3', 'L(1666)/L(4363)', 'RMS([E(4363), E(1666)])')
"[OIII] 1666/5007" : ('O3', 'L(1666)/L(5007)', 'RMS([E(5007), E(1666)])')
"[OIII] 1666/5007+" : ('O3', 'L(1666)/(L(5007)+L(4959))', 'RMS([E(5007)*L(5007)/(L(5007)+L(4959)), E(49
"[OIII] 4363/5007" : ('O3', 'L(4363)/L(5007)', 'RMS([E(5007), E(4363)])')
"[OIII] 4363/5007+" : ('O3', 'L(4363)/(L(5007)+L(4959))', 'RMS([E(5007)*L(5007)/(L(5007)+L(4959)), E(49
"[OIII] 5007/88m" : ('O3', 'L(5007)/L(883000)', 'RMS([E(883000), E(5007)])')
"[OIII] 51m/88m" : ('O3', 'L(518000)/L(883000)', 'RMS([E(883000), E(518000)])')
"[OII] 3726/3729" : ('O2', 'L(3726)/L(3729)', 'RMS([E(3729), E(3726)])')
"[OII] 3727+/7325+" : ('O2', '(L(3726)+L(3729))/(B("7319A+")+B("7330A+"))', 'RMS([E(3726)*L(3726)/(L(37
"[OII] 3727+/7325+b" : ('O2', '(L(3726)+L(3729))/(I(4,2)+I(4,3)+I(5,2)+I(5,3))', 'RMS([E(3726)*L(3726)/
"[OIV] 1400+/25.9m" : ('O4', 'B("1400A+")/L(259000)', 'RMS([BE("1400A+"), E(259000)])')
"[OIV] 1401/1405" : ('O4', 'L(1401)/L(1405)', 'RMS([E(1401), E(1405)])')
"[OI] 5577/6300" : ('O1', 'L(5577)/L(6300)', 'RMS([E(6300), E(5577)])')
"[OI] 5577/6300+" : ('O1', 'L(5577)/(L(6300)+L(6364))', 'RMS([E(6300)*L(6300)/(L(6300)+L(6364)), E(6364
"[OI] 5577/6302" : ('O1', 'L(5577)/L(6300)', 'RMS([E(6300), E(5577)])')
"[OI] 63m/147m" : ('O1', 'L(632000)/L(1455000)', 'RMS([E(632000), E(1455000)])')
"[SIII] 18.7m/33.5m" : ('S3', 'L(187000)/L(335000)', 'RMS([E(335000), E(187000)])')
"[SIII] 6312/18.7m" : ('S3', 'L(6312)/L(187000)', 'RMS([E(187000), E(6312)])')
"[SIII] 6312/9069" : ('S3', 'L(6312)/L(9069)', 'RMS([E(9069), E(6312)])')
"[SIII] 6312/9200+" : ('S3', 'L(6312)/(L(9069)+L(9531))', 'RMS([E(9069)*L(9069)/(L(9069)+L(9531)), E(95
"[SIII] 9069/18.7m" : ('S3', 'L(9069)/L(187000)', 'RMS([E(187000), E(9069)])')
"[SII] 4069/4076" : ('S2', 'L(4069)/L(4076)', 'RMS([E(4069), E(4076)])')
"[SII] 4072+/6720+" : ('S2', '(L(4069)+L(4076))/(L(6716)+L(6731))', 'RMS([E(6716)*L(6716)/(L(6716)+L(67
"[SII] 6731/6716" : ('S2', 'L(6731)/L(6716)', 'RMS([E(6716), E(6731)])')
```

0.2.2 Diagnostic diagram combining various atoms

The plotting tool included in the **Diagnostics** class requires an **EmisGrid** dictionary (as returned by **pn.getEmisGridDict**; see the previous section) and an **Observation** object. The plot is obtained by:

```
In [19]: obs = pn.Observation()
         obs.readData('observations1.dat', fileFormat='lines_in_rows', err_default=0.05) # fill obs wit
         obs.def_EBV(label1="H1r_6563A", label2="H1r_4861A", r_theo=2.85)
         obs.correctData(normWave=4861.)
```

```
In [20]: diags = pn.Diagnostics()
         diags.addDiagsFromObs(obs)
         print(diags.diags)
```

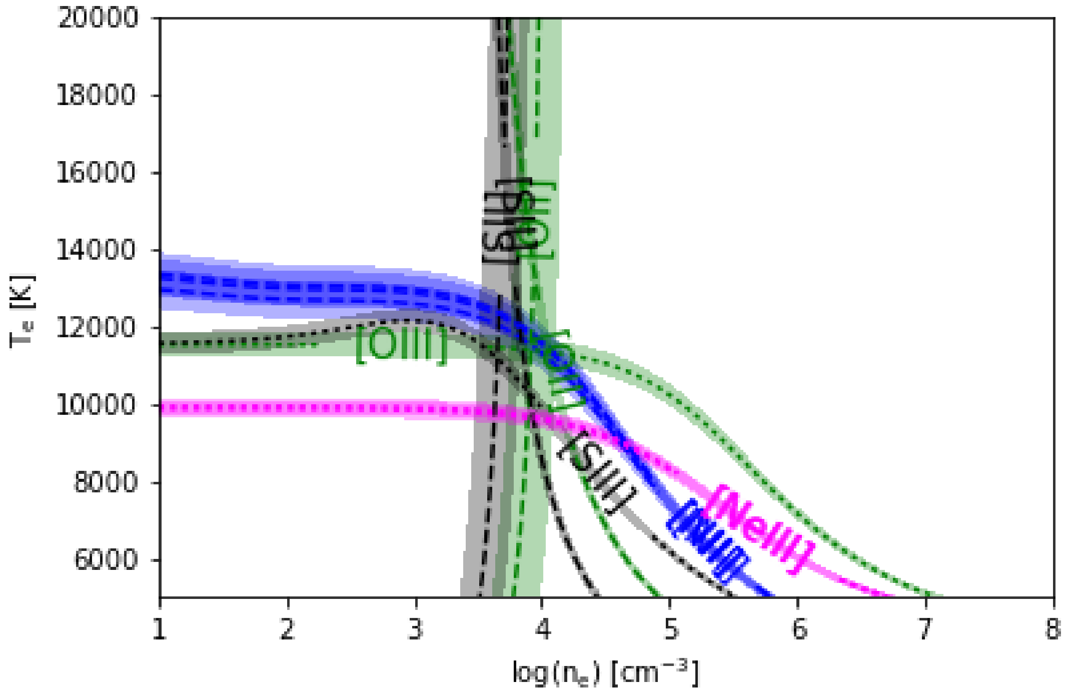
```
{'[NII] 5755/6548': ('N2', 'L(5755)/L(6548)', 'RMS([E(6548), E(5755)])'), '[NII] 5755/6584': ('N2', 'L(5
```

```
In [21]: print(obs.getIntens())
```

```
{'S4_10.5m': array([ 7.]), 'Ne2_12.8m': array([ 8.30000019]), 'Ne3_15.6m': array([ 34.09999847]), 'S3_18
```

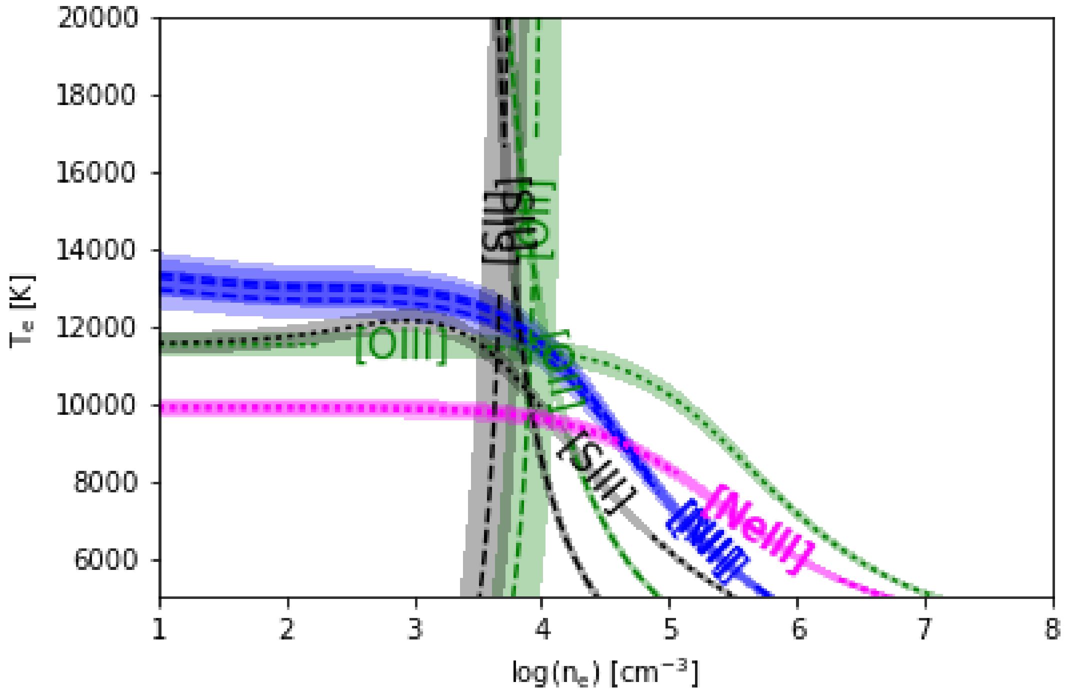
```
In [23]: import matplotlib as mpl
         %config InlineBackend.figure_format = 'png'
         mpl.rc("savefig", dpi=150)
         emisgrids = pn.getEmisGridDict(atomDict=diags.atomDict)
         diags.plot(emisgrids, obs)
```

```
warng getEmisGridDict: Emission map not found: ./pypics//emis_S2.pypic
warng getEmisGridDict: Emission map not found: ./pypics//emis_S3.pypic
warng getEmisGridDict: Emission map not found: ./pypics//emis_Ne3.pypic
```



If there is more than one spectrum in the **Observation** object, the index of the observation to be used is given by **i_obs**:

```
In [24]: diags.plot(emisgrids, obs, i_obs = 0)
```

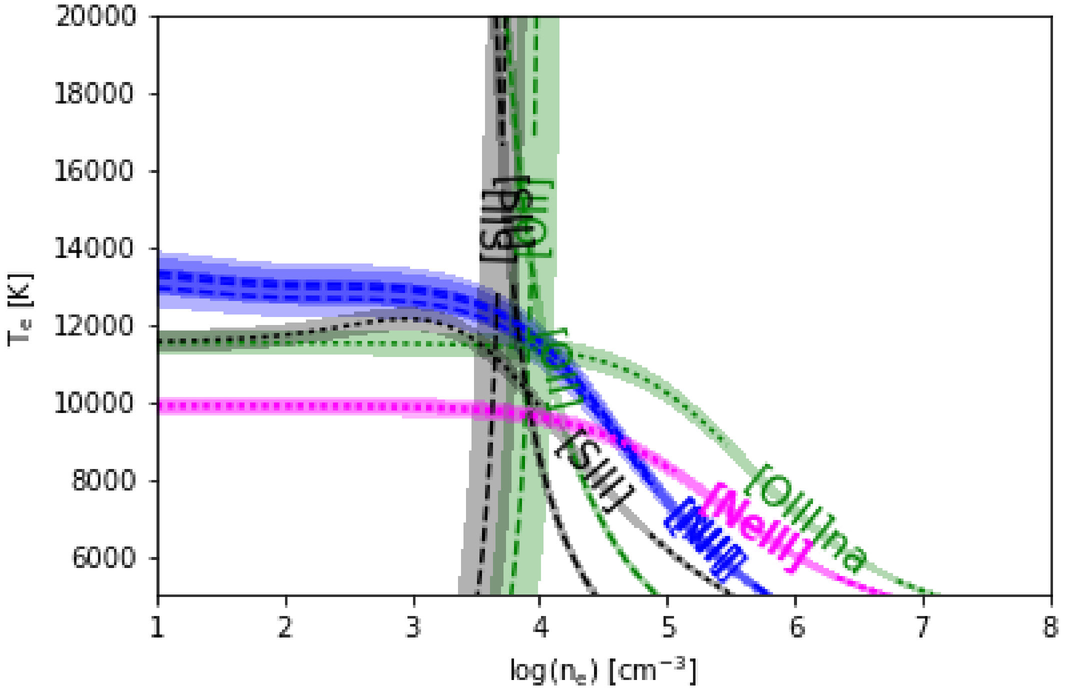


```
In [25]: diags.diags

Out[25]: {'[NII] 5755/6548': ('N2', 'L(5755)/L(6548)', 'RMS([E(6548), E(5755)])'),
          '[NII] 5755/6584': ('N2', 'L(5755)/L(6584)', 'RMS([E(6584), E(5755)])'),
          '[NII] 5755/6584+': ('N2',
           'L(5755)/(L(6548)+L(6584))',
           'RMS([E(6548)*L(6548)/(L(6548)+L(6584)), E(6584)*L(6584)/(L(6584)+L(6548)), E(5755)])'),
          '[NeIII] 3869/15.6m': ('Ne3',
           'L(3869)/L(156000)',
           'RMS([E(156000), E(3869)])'),
          '[NeIII] 3930+/15.6m': ('Ne3',
           '(L(3869)+L(3968))/L(156000)',
           'RMS([E(156000), E(3869)*L(3869)/(L(3869)+L(3968)), E(3968)*L(3968)/(L(3869)+L(3968))])'),
          '[OIII] 4363/5007': ('O3', 'L(4363)/L(5007)', 'RMS([E(5007), E(4363)])'),
          '[OII] 3726/3729': ('O2', 'L(3726)/L(3729)', 'RMS([E(3729), E(3726)])'),
          '[OII] 3727+/7325+': ('O2',
           '(L(3726)+L(3729))/(B("7319A+")+B("7330A+"))',
           'RMS([E(3726)*L(3726)/(L(3726)+L(3729)), E(3729)*L(3729)/(L(3726)+L(3729)),BE("7319A+")*B("73
          '[SIII] 6312/18.7m': ('S3', 'L(6312)/L(187000)', 'RMS([E(187000), E(6312)])'),
          '[SII] 4069/4076': ('S2', 'L(4069)/L(4076)', 'RMS([E(4069), E(4076)])'),
          '[SII] 4072+/6720+': ('S2',
           '(L(4069)+L(4076))/(L(6716)+L(6731))',
           'RMS([E(6716)*L(6716)/(L(6716)+L(6731)), E(6731)*L(6731)/(L(6716)+L(6731)), E(4069)*L(4069)/
          '[SII] 6731/6716': ('S2', 'L(6731)/L(6716)', 'RMS([E(6716), E(6731)])')}
```

The label used to identify emission lines can be changed in the **diags** object, using for example:

```
In [26]: diags.addClabel('[OIII] 4363/5007', '[OIII]na')
In [27]: diags.plot(emisgrids, obs, i_obs = 0)
```


T_e [K]
log(n_e) [cm^{-3}]
[SII]
[OII]
[OIII]
[SIII]
[NII]
[NeIII]
[OIII]na