IDL Library for Plasma Diagnostics and Abundance Analysis

API Documentation for proEQUIB

Contents

I Overview **5**

II API **9**

Directory: ./ 11

Overview 11
proequib.idldoc 11
calc_abund_c_ii_rl.pro 13
calc_abund_c_iii_rl.pro 15
calc_abund_he_i_rl.pro 16
calc_abund_he_ii_rl.pro 18
calc_abund_n_ii_rl.pro 20
calc_abund_n_iii_rl.pro 22
calc_abund_ne_ii_rl.pro 24
calc_abund_o_ii_rl.pro 25
calc_abundance.pro 27
calc_crit_density.pro 30
calc_density.pro 33
calc_emiss_c_ii_rl.pro 36
calc_emiss_c_iii_rl.pro 37
calc_emiss_h_beta.pro 39
calc_emiss_he_i_rl.pro 40
calc_emiss_he_ii_rl.pro 42
calc_emiss_n_ii_rl.pro 43
calc_emiss_n_iii_rl.pro 45
calc_emiss_ne_ii_rl.pro 46
calc_emiss_o_ii_rl.pro 48
calc_emissivity.pro 49
calc_populations.pro 52
calc_temperature.pro 55
collision__define.pro 58

deredden_flux.pro 84
deredden_relflux.pro 85
find_aeff_sh95_column.pro 87
get_omij_temp.pro 88
print_ionic.pro 90
recombination__define.pro 92
reddening__define.pro 122
redlaw.pro 138
redlaw_ccm.pro 139
redlaw_fm.pro 140
redlaw_gal.pro 142
redlaw_gal2.pro 143
redlaw_jbk.pro 144
redlaw_lmc.pro 145
redlaw_smc.pro 146

Part I

Overview

Overview

proEQUIB is an IDL library for plasma diagnostics and abundance analysis in nebular astrophysics. This library has API functions written in Interactive Data Language (IDL)/GNU Data Language (GDL) programs. It uses the AtomNeb IDL library, which can be used to determine interstellar extinctions, electron temperatures, electron densities, and ionic abundances from collisionally excited lines (CEL) and recombination lines (RL).

proEQUIB mainly contains the follwing API functions written purely in IDL/GDL:

* `API functions for collisionally excited lines (CEL)` have been developed based on the algorithm of the FORTRAN program EQUIB written in FORTRAN by Howarth & Adams (1981). The program EQUIB calculates atomic level populations and line emissivities in statistical equilibrium in multi-level atoms for different physical conditions of the stratification layers where the chemical elements are ionized. Using the IDL/GDL implementation of the program EQUIB, electron temperatures, electron densities, and ionic abundances are determined from the measured fluxes of collisionally excited lines.

* `API functions for recombination lines (RL)` have been developed based on the algorithm of the recombination scripts by X. W. Liu and Y. Zhang included in the FORTRAN program MOCASSIN. These API functiosn are used to determine ionic abundances from recombination lines for some heavy element ions.

* `API functions for reddening and extinctions` have been developed according to the methods of the reddening law functions from STSDAS IRAF Package, which are used to obtain interstellar extinctions and deredden measured fluxes based on different reddening laws.

Dependencies

* This package requires the following packages:

- The IDL Astronomy User's Library
- The AtomNeb IDL Library
- IDL MCMC Hammer library (currently not used!)

* To get this package with all the dependent packages, you can simply use git command as follows:

```
git clone --recursive https://github.com/equib/proEQUIB.git
```

GDL Installation

* The GNU Data Language (GDL) can be installed on

- Linux (Fedora):

```
sudo dnf install gdl
```

- Linux (Ubuntu):

```
sudo apt-get install gnudatalanguage
```

- OS X:

```
brew install gnudatalanguage
```

- Windows: using the GNU Data Language for Win32 (Unofficial Version) or compiling the GitHub source with Visual Studio 2015 as seen in appveyor.yml.

* To setup proEQUIB in GDL, add its path to .gdl_startup in the home directory:

```
!PATH=!PATH + ':/home/proEQUIB/pro/'
!PATH=!PATH + ':/home/proEQUIB/externals/misc/'
!PATH=!PATH + ':/home/proEQUIB/externals/astron/pro/'
!PATH=!PATH + ':/home/proEQUIB/externals/atomneb/pro/'
```

```
Set ‘‘GDL_STARTUP‘‘ in ‘‘.bashrc‘‘ (bash):

export GDL_STARTUP=~/.gdl_startup
```

or in .tcshrc (cshrc):

```
setenv GDL_STARTUP ~/.gdl_startup
```

* This package needs GDL version 0.9.8 or later.

IDL Installation

* To install proEQUIB in IDL, add its path to your IDL path. For more information about the path management in IDL, read the IDL path management by Harris Geospatial or the IDL library installation by David Fanning.

* This package needs IDL version 7.1 or later.

Project statistics

Directories:	1
.pro files:	39
.sav files:	0
Routines:	110
Lines:	3,049

Part II

API

Directory: ./

Overview

proequib.idldoc

proEQUIB is an IDL library for plasma diagnostics and abundance analysis in nebular astrophysics. This library has API functions written in Interactive Data Language_ (IDL)/GNU Data Language_ (GDL) programs. It uses the AtomNeb IDL library_, which can be used to determine interstellar extinctions, electron temperatures, electron densities, and ionic abundances from collisionally excited lines (CEL) and recombination lines (RL).

proEQUIB mainly contains the follwing API functions written purely in IDL/GDL:

* **API functions for collisionally excited lines (CEL)** have been developed based on the algorithm of the FORTRAN program EQUIB_ written in FORTRAN by Howarth & Adams (1981)_. The program EQUIB calculates atomic level populations and line emissivities in statistical equilibrium in multi-level atoms for different physical conditions of the stratification layers where the chemical elements are ionized. Using the IDL/GDL implementation of the program EQUIB_, electron temperatures, electron densities, and ionic abundances are determined from the measured fluxes of collisionally excited lines.

* **API functions for recombination lines (RL)** have been developed based on the algorithm of the recombination scripts by X. W. Liu and Y. Zhang included in the FORTRAN program MOCASSIN_. These API functiosn are used to determine ionic abundances from recombination lines for some heavy element ions.

* **API functions for reddening and extinctions** have been developed according to the methods of the reddening law functions from STSDAS IRAF Package_, which are used to obtain interstellar extinctions and deredden measured fluxes based on different reddening laws.

Dependencies
* This package requires the following packages:
- The IDL Astronomy User's Library_
- The AtomNeb IDL Library_
- IDL MCMC Hammer library_ (currently not used!)
* To get this package with all the dependent packages, you can simply use git command as follows:

```
git clone --recursive https://github.com/equib/proEQUIB.git
```

GDL Installation
* The GNU Data Language (GDL) can be installed on
- Linux (Fedora):

```
sudo dnf install gdl
```

- Linux (Ubuntu):

```
sudo apt-get install gnudatalanguage
```

- OS X:

```
brew install gnudatalanguage
```

- Windows: using the GNU Data Language for Win32_ (Unofficial Version) or compiling the GitHub source_ with Visual Studio 2015 as seen in appveyor.yml_.
* To setup **proEQUIB** in GDL, add its path to .gdl_startup in the home directory:

```
!PATH=!PATH + ':/home/proEQUIB/pro/'
 !PATH=!PATH + ':/home/proEQUIB/externals/misc/'
 !PATH=!PATH + ':/home/proEQUIB/externals/astron/pro/'
 !PATH=!PATH + ':/home/proEQUIB/externals/atomneb/pro/'

Set ''GDL_STARTUP'' in ''.bashrc'' (bash):
```

```
export GDL_STARTUP=~/.gdl_startup
```

or in .tcshrc (cshrc):

```
setenv GDL_STARTUP ~/.gdl_startup
```

* This package needs GDL version 0.9.8 or later.
IDL Installation
* To install **proEQUIB** in IDL, add its path to your IDL path.
For more information about the path management in IDL, read
`the IDL path management<https://www.harrisgeospatial.com/Support/Self-Help-Tools/Help-Articles/Help-Articles-`
by Harris Geospatial or the IDL library installation _ by David
Fanning.
* This package needs IDL version 7.1 or later.

calc_abund_c_ii_rl.pro

CALC_ABUND_C_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of C II recombination line by using the recombination coefficients from from Davey et al. (2000) 2000A&AS..142...85D.

```
result = calc_abund_c_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =float, c_ii_rc_data=array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

c_ii_rc_data IN REQUIRED TYPE=array/object
C II recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='c'
 IDL> ion='iii' ; C II
 IDL> c_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> c_ii_6151_flux = 0.028
 IDL> wavelength=6151.43
 IDL> Abund_c_ii=calc_abund_c_ii_rl(temperature=temperature, density=density, $
 IDL>                              wavelength=wavelength, line_flux=c_ii_6151_flux, $
 IDL>                              c_ii_rc_data=c_ii_rc_data, h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'N(C^2+)/N(H+):', Abund_c_ii
    N(C^2+)/N(H+):    0.00063404650
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for C II lines from
Davey et al. 2000A&AS..142...85D.

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, added to MOCASSIN.

10/05/2013, A. Danehkar, Translated to IDL code.

15/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_ii_rl()
for calculating line emissivities and separated it from calc_abund_c_ii_rl().

Version

0.3.0

calc_abund_c_iii_rl.pro

CALC_ABUND_C_III_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of C III recombination line by using the recombination coefficients from Pequignot et al. 1991A&A...251..680P.

```
  result = calc_abund_c_iii_rl(temperature=float, density=float, wavelength=float, line_flux
     =float, c_iii_rc_data=array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

c_iii_rc_data IN REQUIRED TYPE=array/object
C III recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_PPB91_file='/media/linux/proEQUIB/AtomNeb-idl/atomic-data-rc/rc_PPB91.fits'
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='c'
 IDL> ion='iv' ; C III
 IDL> c_iii_rc_data=atomneb_read_aeff_ppb91(Atom_RC_PPB91_file, atom, ion)
IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> c_iii_4647_flux = 0.107
IDL> wavelength=4647.42
IDL> Abund_c_iii=calc_abund_c_iii_rl(temperature=temperature, density=density, $
IDL>                                wavelength=wavelength, line_flux=c_iii_4647_flux, $
IDL>                                c_iii_rc_data=c_iii_rc_data, h_i_aeff_data=h_i_aeff_data)
IDL> print, 'N(C^3+)/N(H+):', Abund_c_iii
   N(C^3+)/N(H+):    0.00017502840
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for C
III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

18/05/2013, A. Danehkar, Translated to IDL code.

06/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_iii_rl()
for calculating line emissivities and separated it from calc_abund_c_iii_rl().

Version

0.3.0

calc_abund_he_i_rl.pro

CALC_ABUND_HE_I_RL

This function determines the ionic abundance from the observed
flux intensity for the given wavelength of He I recombination
line by using the recombination coefficients from Porter et al.
2012MNRAS.425L..28P.

```
result = calc_abund_he_i_rl(temperature=float, density=float, linenum=int, line_flux=
   float, he_i_aeff_data=array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature in required type=float
: electron temperature

density in required type=float
: electron density

linenum in required type=int
: Line Number for Wavelength
Wavelength=4120.84:linenum=7,
Wavelength=4387.93: linenum=8,
Wavelength=4437.55: linenum=9,
Wavelength=4471.50: linenum=10,
Wavelength=4921.93: linenum=12,
Wavelength=5015.68: linenum=13,
Wavelength=5047.74: linenum=14,
Wavelength=5875.66: linenum=15,
Wavelength=6678.16: linenum=16,
Wavelength=7065.25: linenum=17,
Wavelength=7281.35: linenum=18.

line_flux in required type=float
: line flux intensity

he_i_aeff_data in required type=array/object
: He I recombination coefficients

h_i_aeff_data in required type=array/object
: H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_He_I_file= filepath('rc_he_ii_PFSD12.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='he'
 IDL> ion='ii' ; He I
 IDL> he_i_rc_data=atomneb_read_aeff_he_i_pfsd12(Atom_RC_He_I_file, atom, ion)
 IDL> he_i_aeff_data=he_i_rc_data[0].Aeff
 IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> he_i_4471_flux= 2.104
IDL> linenum=10; 4471.50
IDL> Abund_he_i=calc_abund_he_i_rl(temperature=temperature, density=density, $
                                 linenum=linenum, line_flux=he_i_4471_flux, $
                                 he_i_aeff_data=he_i_aeff_data, h_i_aeff_data=h_i_aeff_data)
IDL> print, 'N(He^+)/N(H^+):', Abund_he_i
   N(He^+)/N(H^+):     0.040848393
```

Author

Ashkbiz Danehkar

Copyright

History

Based on improved He I emissivities in the case B from
Porter et al. 2012MNRAS.425L..28P

15/12/2013, A. Danehkar, IDL code written.

20/03/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_i_rl()
for calculating line emissivities and separated it from calc_abund_he_i_rl().

Version

0.3.0

calc_abund_he_ii_rl.pro

CALC_ABUND_HE_II_RL

This function determines the ionic abundance from the observed flux intensity for the He II recombination line 4686 A by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
result = calc_abund_he_ii_rl(temperature=float, density=float, line_flux=float, he_ii_aeff_data
   =array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
: electron temperature

density IN REQUIRED TYPE=float
: electron density

line_flux IN REQUIRED TYPE=float
: line flux intensity

he_ii_aeff_data IN REQUIRED TYPE=array/object
: He II recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
: H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_He_I_file= filepath('rc_he_ii_PFSD12.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='he'
 IDL> ion='iii' ; He II
 IDL> he_ii_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> he_ii_aeff_data=he_ii_rc_data[0].Aeff
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> he_ii_4686_flux = 135.833
 IDL> Abund_he_ii=calc_abund_he_ii_rl(temperature=temperature, density=density, $
 IDL>                                 line_flux=he_ii_4686_flux, $
 IDL>                                 he_ii_aeff_data=he_ii_aeff_data, h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'N(He^2+)/N(H^+):', Abund_he_ii
    N(He^2+)/N(H^+):      0.11228817
```

Author

Ashkbiz Danehkar

Copyright

History

Based on He II emissivities from Storey & Hummer, 1995MNRAS.272...41S.

15/12/2013, A. Danehkar, IDL code written.

02/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_ii_rl()
for calculating line emissivities and separated it from calc_abund_he_ii_rl().

Version

0.3.0

calc_abund_n_ii_rl.pro

CALC_ABUND_N_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of N II recombination line by using the recombination coefficients from Escalante & Victor 1990ApJS...73..513E.

```
result = calc_abund_n_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =float, n_ii_rc_br=array/object, n_ii_rc_data=array/object, h_i_aeff_data=array/object
   )
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

n_ii_rc_br IN REQUIRED TYPE=array/object
N II branching ratios (Br)

n_ii_rc_data IN REQUIRED TYPE=array/object
N II recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='n'
 IDL> ion='iii' ; N II
 IDL> n_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
 IDL> n_ii_rc_data_br=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion, /br)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> n_ii_4442_flux = 0.017
 IDL> wavelength=4442.02
 IDL> Abund_n_ii=calc_abund_n_ii_rl(temperature=temperature, density=density, $
 IDL>                              wavelength=wavelength, line_flux=n_ii_4442_flux, $
 IDL>                              n_ii_rc_br=n_ii_rc_data_br, n_ii_rc_data=n_ii_rc_data, $
 IDL>                              h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'N(N^2+)/N(H+):', Abund_n_ii
    N(N^2+)/N(H+):   0.00069297541
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Effective recombination coefficients for N II lines from Escalante & Victor 1990ApJS...73..513E.

Adopted from MIDAS Rnii script written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MOCASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_ii_rl()
for calculating line emissivities and separated it from calc_abund_n_ii_rl().

Version

0.3.0

calc_abund_n_iii_rl.pro

CALC_ABUND_N_III_RL

This function determines the ionic abundance from the observed
flux intensity for the given wavelength of N III recombination
line by using the recombination coefficients from Pequignot et al.
1991A&A...251..680P.

```
result = calc_abund_n_iii_rl(temperature=float, density=float, wavelength=float, line_flux
   =float, n_iii_rc_data=array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

n_iii_rc_data IN REQUIRED TYPE=array/object
N III recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_PPB91_file='/media/linux/proEQUIB/AtomNeb-idl/atomic-data-rc/rc_PPB91.fits'
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='n'
 IDL> ion='iv' ; N III
 IDL> n_iii_rc_data=atomneb_read_aeff_ppb91(Atom_RC_PPB91_file, atom, ion)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> n_iii_4641_flux = 0.245
 IDL> wavelength=4640.64
 IDL> Abund_n_iii=calc_abund_n_iii_rl(temperature=temperature, density=density, $
 IDL>                                 wavelength=wavelength, line_flux=n_iii_4641_flux, $
 IDL>                                 n_iii_rc_data=n_iii_rc_data, h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'N(N^3+)/N(H+):', Abund_n_iii
    N(N^3+)/N(H+):    6.3366175e-05
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for N
III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

10/05/2013, A. Danehkar, IDL code written.

20/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_iii_rl()
for calculating line emissivities and separated it from calc_abund_n_iii_rl().

Version

0.3.0

calc_abund_ne_ii_rl.pro

CALC_ABUND_NE_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of Ne II recombination line by using the recombination coefficients from Kisielius et al. (1998) & Storey (unpublished).

```
result = calc_abund_ne_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =float, ne_ii_rc_data=array/object, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

ne_ii_rc_data IN REQUIRED TYPE=array/object
Ne II recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='ne'
 IDL> ion='iii' ; Ne II
 IDL> ne_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> ne_ii_3777_flux = 0.056
IDL> wavelength=3777.14
IDL> Abund_ne_ii=calc_abund_ne_ii_rl(temperature=temperature, density=density, $
IDL>                                 wavelength=wavelength, line_flux=ne_ii_3777_flux, $
IDL>                                 ne_ii_rc_data=ne_ii_rc_data, h_i_aeff_data=h_i_aeff_data)
IDL> print, 'N(Ne^2+)/N(H+):', Abund_ne_ii
   N(Ne^2+)/N(H+):    0.00043376850
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for Ne II lines from Kisielius et al. 1998A&AS..133..257K & Storey (unpublished).

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, scripts added to MOCASSIN.

14/05/2013, A. Danehkar, Translated to IDL code.

10/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_ne_ii_rl() for calculating line emissivities and separated it from calc_abund_ne_ii_rl().

Version

0.3.0

calc_abund_o_ii_rl.pro

CALC_ABUND_O_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of O II recombination line by using the recombination coefficients from Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

```
result = calc_abund_o_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =float, o_ii_rc_br=array/object, o_ii_rc_data=array/object, h_i_aeff_data=array/object
   )
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

o_ii_rc_br IN REQUIRED TYPE=array/object
O II branching ratios (Br)

o_ii_rc_data IN REQUIRED TYPE=array/object
O II recombination coefficients

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='o'
 IDL> ion='iii' ; O II
 IDL> o_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
 IDL> o_ii_rc_data_br=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion, /br)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> o_ii_4614_flux = 0.009
 IDL> wavelength=4613.68
IDL> Abund_o_ii=calc_abund_o_ii_rl(temperature=temperature, density=density, $
IDL>                              wavelength=wavelength, line_flux=o_ii_4614_flux, $
IDL>                              o_ii_rc_br=o_ii_rc_data_br, o_ii_rc_data=o_ii_rc_data, $
IDL>                              h_i_aeff_data=h_i_aeff_data)
IDL> print, 'N(O^2+)/N(H+):', Abund_o_ii
   N(O^2+)/N(H+):    0.0018886330
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for O II lines from
Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

Adopted from MIDAS script Roii.prg written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MO-
CASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_o_ii_rl()
for calculating line emissivities and separated it from calc_abund_o_ii_rl().

Version

0.3.0

calc_abundance.pro

CALC_ABUNDANCE

This function determines the ionic abundance from the observed
flux intensity for specified ion with level(s) by solving atomic level
populations and line emissivities in statistical equilibrium for
given electron density and temperature.

```
result = calc_abundance(temperature=float, density=float, line_flux=float, atomic_levels=
   string, elj_data=array/object, omij_data=array/object, aij_data=array/object, h_i_aeff_data
   =array/object)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature in required type=float
electron temperature

density in required type=float
electron density

line_flux in required type=float
line flux intensity

atomic_levels in required type=string
level(s) e.g '1,2/', '1,2,1,3/'

elj_data in required type=array/object
energy levels (Ej) data

omij_data in required type=array/object
collision strengths (omega_ij) data

aij_data in required type=array/object
transition probabilities (Aij) data

h_i_aeff_data in required type=array/object
H I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='o'
 IDL> ion='iii'
 IDL> o_iii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> o_iii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> o_iii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> hi_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=hi_rc_data[0].Aeff
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
IDL> atomic_levels='3,4/'
IDL> iobs5007=double(1200.0)
IDL> Abb5007=double(0.0)
IDL> Abb5007=calc_abundance(temperature=temperature, density=density, $
IDL>                        line_flux=iobs5007, atomic_levels=atomic_levels,$
IDL>                        elj_data=o_iii_elj, omij_data=o_iii_omij, $
IDL>                        aij_data=o_iii_aij, h_i_aeff_data=hi_rc_data[0].Aeff)
IDL> print, 'N(O^2+)/N(H+):', Abb5007
   N(O^2+)/N(H+):   0.00041256231
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

21/11/2016, A. Danehkar, Made a new function calc_emissivity() for calculating line emissivities and separated it from calc_abundance().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_abundance().

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

calc_crit_density.pro

CALC_CRIT_DENSITY

This function calculates critical densities in statistical equilibrium for given electron temperature.

```
result = calc_crit_density(temperature=float, elj_data=array/object, omij_data=array/
    object, aij_data=array/object, level_num=int, irats=int)
```

Returns

type=array/object. This function returns the critical densities.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

elj_data IN REQUIRED TYPE=array/object
energy levels (Ej) data

omij_data IN REQUIRED TYPE=array/object
collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
transition probabilities (Aij) data

level_num IN TYPE=int
Number of levels

irats IN TYPE=int
Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='s'
 IDL> ion='ii'
 IDL> s_ii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> s_ii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> s_ii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)\
 IDL> temperature=double(10000.0)
 IDL> N_crit=calc_crit_density(temperature=temperature, $
 IDL>                         elj_data=s_ii_elj, omij_data=s_ii_omij, $
 IDL>                         aij_data=s_ii_aij)
 IDL> print, 'Critical Densities:', N_crit
    Critical Densities:       0.0000000       5007.8396       1732.8414       1072685.0       2220758.1
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_populations().

27/02/2019, A. Danehkar, Simplify the calc_populations() routine for external usage.

01/03/2019, A. Danehkar, Create the calc_crit_density() routine from the calc_populations() routine.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

calc_density.pro

CALC_DENSITY

This function determines electron density from given flux intensity ratio for specified ion with upper level(s) lower level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron temperature.

```
result = calc_density(line_flux_ratio=float, temperature=float, upper_levels=string,
   lower_levels=string, elj_data=array/object, omij_data=array/object, aij_data=array/
   object [, low_density=float] [, high_density=float] [, num_density=integer] [, min_temperature
   =float])
```

Returns

type=double. This function returns the electron density.

Keywords

line_flux_ratio in required type=float
flux intensity ratio

temperature in required type=float
electron temperature

upper_levels in required type=string
upper atomic level(s) e.g '1,2/', '1,2,1,3/'

lower_levels in required type=string
lower atomic level(s) e.g '1,2/', '1,2,1,3/'

elj_data in required type=array/object
energy levels (Ej) data

omij_data in required type=array/object
collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
transition probabilities (Aij) data

low_density IN OPTIONAL TYPE=float
lower density range

high_density IN OPTIONAL TYPE=float
upper density range

num_density IN OPTIONAL TYPE=integer
number of the iteration step

min_temperature IN OPTIONAL TYPE=float
minimum temperature

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='s'
 IDL> ion='ii'
 IDL> s_ii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> s_ii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> s_ii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)\
 IDL> upper_levels='1,2/'
 IDL> lower_levels='1,3/'
 IDL> temperature=double(7000.0);
 IDL> line_flux_ratio=double(1.506);
 IDL> density=calc_density(line_flux_ratio=line_flux_ratio, temperature=temperature, $
 IDL>                      upper_levels=upper_levels, lower_levels=lower_levels, $
 IDL>                      elj_data=s_ii_elj, omij_data=s_ii_omij, $
 IDL>                      aij_data=s_ii_aij)
 IDL> print, "Electron Density:", density
    Electron Density:       2312.6395
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_density().

27/02/2019, A. Danehkar, Fix a bug in the atomic level assumption, and use the simplified calc_populations() routine.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

24/05/2019, A. Danehkar, Add the optional density range.

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

calc_emiss_c_ii_rl.pro

CALC_EMISS_C_II_RL

This function calculates the emissivity for the given wavelength of C II recombination line by using the recombination coefficients from from Davey et al. (2000) 2000A&AS..142...85D.

```
result = calc_emiss_c_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =line_flux, c_ii_rc_data=array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux

c_ii_rc_data IN REQUIRED TYPE=array/object
C II recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
IDL>
IDL> atom='c'
IDL> ion='iii' ; C II
IDL> c_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> wavelength=6151.43
IDL> emiss_c_ii=calc_emiss_c_ii_rl(temperature=temperature, density=density, $
IDL>                              wavelength=wavelength, $
IDL>                              c_ii_rc_data=c_ii_rc_data)
IDL> print, 'Emissivity:', emiss_c_ii
   Emissivity:   5.4719511e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for C II lines from
Davey et al. 2000A&AS..142...85D.

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, added to MOCASSIN.

10/05/2013, A. Danehkar, Translated to IDL code.

15/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_ii_rl()
for calculating line emissivities and separated it from calc_abund_c_ii_rl().

Version

0.3.0

calc_emiss_c_iii_rl.pro

CALC_EMISS_C_III_RL

This function calculates the emissivity for the given wavelength of
C III recombination line by using the recombination coefficients
from Pequignot et al. 1991A&A...251..680P.

```
result = calc_emiss_c_iii_rl(temperature=float, density=float, wavelength=float, c_iii_rc_data
   =array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

c_iii_rc_data IN REQUIRED TYPE=array/object
C III recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_PPB91_file='/media/linux/proEQUIB/AtomNeb-idl/atomic-data-rc/rc_PPB91.fits'
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='c'
 IDL> ion='iv' ; C III
 IDL> c_iii_rc_data=atomneb_read_aeff_ppb91(Atom_RC_PPB91_file, atom, ion)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> wavelength=4647.42
 IDL> emiss_c_iii=calc_emiss_c_iii_rl(temperature=temperature, density=density, $
 IDL>                                 wavelength=wavelength, $
 IDL>                                 c_iii_rc_data=c_iii_rc_data)
 IDL> print, 'Emissivity:', emiss_c_iii
    Emissivity:   7.5749632e-25
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for C III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

18/05/2013, A. Danehkar, Translated to IDL code.

06/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_iii_rl() for calculating line emissivities and separated it from calc_abund_c_iii_rl().

Version

0.3.0

calc_emiss_h_beta.pro

CALC_EMISS_H_BETA

private

This function calculates the emissivity for H_beta 4861A Emis(Hbeta)= 4pi j(HBeta 4861 A)/Np Ne) for the given temperature and density by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
  result = calc_emiss_h_beta(temperature=float, density=float, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the H beta emissivity 4pi j(HBeta 4861)/Np Ne).

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

h_i_aeff_data IN REQUIRED TYPE=array/object
H I recombination coefficients

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on H I emissivities from Storey & Hummer, 1995MNRAS.272...41S.

25/08/2012, A. Danehkar, IDL code written.

11/03/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Change from logarithmic to linear

Version

0.3.0

calc_emiss_he_i_rl.pro

CALC_EMISS_HE_I_RL

This function calculates the emissivity for the given wavelength of He I recombination line by using the recombination coefficients from Porter et al. 2012MNRAS.425L..28P.

```
result = calc_emiss_he_i_rl(temperature=float, density=float, linenum=int, he_i_aeff_data
   =array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

linenum IN REQUIRED TYPE=int
Line Number for Wavelength
Wavelength=4120.84:linenum=7,
Wavelength=4387.93: linenum=8,
Wavelength=4437.55: linenum=9,
Wavelength=4471.50: linenum=10,
Wavelength=4921.93: linenum=12,
Wavelength=5015.68: linenum=13,
Wavelength=5047.74: linenum=14,
Wavelength=5875.66: linenum=15,

Wavelength=6678.16: linenum=16,
Wavelength=7065.25: linenum=17,
Wavelength=7281.35: linenum=18.

he_i_aeff_data IN REQUIRED TYPE=array/object
He I recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_He_I_file= filepath('rc_he_ii_PFSD12.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='he'
 IDL> ion='ii' ; He I
 IDL> he_i_rc_data=atomneb_read_aeff_he_i_pfsd12(Atom_RC_He_I_file, atom, ion)
 IDL> he_i_aeff_data=he_i_rc_data[0].Aeff
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> linenum=10; 4471.50
 IDL> emiss_he_i=calc_emiss_he_i_rl(temperature=temperature, density=density, $
                                   linenum=linenum, $
                                   he_i_aeff_data=he_i_aeff_data)
 IDL> print, 'Emissivity:', emiss_he_i
    Emissivity:   6.3822830e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on improved He I emissivities in the case B from
Porter et al. 2012MNRAS.425L..28P

15/12/2013, A. Danehkar, IDL code written.

20/03/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_i_rl()
for calculating line emissivities and separated it from calc_abund_he_i_rl().

Version

0.3.0

calc_emiss_he_ii_rl.pro

CALC_EMISS_HE_II_RL

This functioncalculates the emissivity for the He II recombination line 4686 A by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
result = calc_emiss_he_ii_rl(temperature=float, density=float, he_ii_aeff_data=array/
   object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

he_ii_aeff_data IN REQUIRED TYPE=array/object
He II recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_He_I_file= filepath('rc_he_ii_PFSD12.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='he'
 IDL> ion='iii' ; He II
 IDL> he_ii_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> he_ii_aeff_data=he_ii_rc_data[0].Aeff
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> he_ii_4686_flux = 135.833
 IDL> emiss_he_ii=calc_emiss_he_ii_rl(temperature=temperature, density=density, $
 IDL>                                 he_ii_aeff_data=he_ii_aeff_data)
 IDL> print, 'Emissivity:', emiss_he_ii
    Emissivity:   1.4989134e-24
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on He II emissivities from Storey & Hummer, 1995MN-RAS.272...41S.

15/12/2013, A. Danehkar, IDL code written.

02/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Change from logarithmic to linear

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_ii_rl() for calculating line emissivities and separated it from calc_abund_he_ii_rl().

Version

0.3.0

calc_emiss_n_ii_rl.pro

CALC_EMISS_N_II_RL

This function calculates the emissivity for the given wavelength of N II recombination line by using the recombination coefficients from Escalante & Victor 1990ApJS...73..513E.

```
result = calc_emiss_n_ii_rl(temperature=float, density=float, wavelength=float, n_ii_rc_br
   =array/object, n_ii_rc_data=array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

n_ii_rc_br IN REQUIRED TYPE=array/object
N II branching ratios (Br)

n_ii_rc_data IN REQUIRED TYPE=array/object
N II recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> atom='h'
 IDL> ion='ii' ; H I
 IDL> h_i_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
 IDL> h_i_aeff_data=h_i_rc_data[0].Aeff
 IDL> atom='n'
 IDL> ion='iii' ; N II
 IDL> n_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
 IDL> n_ii_rc_data_br=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion, /br)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> wavelength=4442.02
 IDL> emiss_n_ii=calc_emiss_n_ii_rl(temperature=temperature, density=density, $
 IDL>                              wavelength=wavelength, $
 IDL>                              n_ii_rc_br=n_ii_rc_data_br, n_ii_rc_data=n_ii_rc_data, $
 IDL>                              h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'Emissivity:', emiss_n_ii
    Emissivity:   3.0397397e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Effective recombination coefficients for N II lines
from Escalante & Victor 1990ApJS...73..513E.

Adopted from MIDAS Rnii script written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MOCASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_ii_rl()
for calculating line emissivities and separated it from calc_abund_n_ii_rl().

Version

0.3.0

calc_emiss_n_iii_rl.pro

CALC_EMISS_N_III_RL

This function calculates the emissivity for the given wavelength of
N III recombination line by using the recombination coefficients
from Pequignot et al. 1991A&A...251..680P.

```
result = calc_emiss_n_iii_rl(temperature=float, density=float, wavelength=float, n_iii_rc_data
   =array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

n_iii_rc_data IN REQUIRED TYPE=array/object
N III recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_PPB91_file='/media/linux/proEQUIB/AtomNeb-idl/atomic-data-rc/rc_PPB91.fits'
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='n'
 IDL> ion='iv' ; N III
 IDL> n_iii_rc_data=atomneb_read_aeff_ppb91(Atom_RC_PPB91_file, atom, ion)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
IDL> wavelength=4640.64
IDL> emiss_n_iii=calc_abund_n_iii_rl(temperature=temperature, density=density, $
IDL>                                 wavelength=wavelength, $
IDL>                                 n_iii_rc_data=n_iii_rc_data)
IDL> print, 'Emissivity:', emiss_n_iii
   Emissivity:   4.7908644e-24
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for N
III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

10/05/2013, A. Danehkar, IDL code written.

20/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_iii_rl()
for calculating line emissivities and separated it from calc_abund_n_iii_rl().

Version

0.3.0

calc_emiss_ne_ii_rl.pro

CALC_EMISS_NE_II_RL

This function calculates the emissivity for the given wavelength of
Ne II recombination line by using the recombination coefficients
from Kisielius et al. (1998) & Storey (unpublished).

```
result = calc_emiss_ne_ii_rl(temperature=float, density=float, wavelength=float, line_flux
   =line_flux, ne_ii_rc_data=array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
: electron density

wavelength IN REQUIRED TYPE=float
: Line Wavelength in Angstrom

line_flux

ne_ii_rc_data IN REQUIRED TYPE=array/object
: Ne II recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='ne'
 IDL> ion='iii' ; Ne II
 IDL> ne_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> wavelength=3777.14
 IDL> emiss_ne_ii=calc_emiss_ne_ii_rl(temperature=temperature, density=density, $
 IDL>                                 wavelength=wavelength, $
 IDL>                                 ne_ii_rc_data=ne_ii_rc_data, h_i_aeff_data=h_i_aeff_data)
 IDL> print, 'Emissivity:', emiss_ne_ii
    Emissivity:   1.5996881e-25
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for Ne
II lines from Kisielius et al. 1998A&AS..133..257K & Storey
(unpublished).

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, scripts added to MOCASSIN.

14/05/2013, A. Danehkar, Translated to IDL code.

10/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_ne_ii_rl()
for calculating line emissivities and separated it from calc_abund_ne_ii_rl().

Version

0.3.0

calc_emiss_o_ii_rl.pro

CALC_EMISS_O_II_RL

This function calculates the emissivity for the given wavelength of O II recombination line by using the recombination coefficients from Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

```
result = calc_emiss_o_ii_rl(temperature=float, density=float, wavelength=float, o_ii_rc_br
   =array/object, o_ii_rc_data=array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

o_ii_rc_br IN REQUIRED TYPE=array/object
O II branching ratios (Br)

o_ii_rc_data IN REQUIRED TYPE=array/object
O II recombination coefficients

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_All_file= filepath('rc_collection.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
 IDL>
 IDL> atom='o'
 IDL> ion='iii' ; O II
IDL> o_ii_rc_data=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion)
IDL> o_ii_rc_data_br=atomneb_read_aeff_collection(Atom_RC_All_file, atom, ion, /br)
IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> wavelength=4613.68
IDL> emiss_o_ii=calc_emiss_o_ii_rl(temperature=temperature, density=density, $
IDL>                               wavelength=wavelength, $
IDL>                               o_ii_rc_br=o_ii_rc_data_br, o_ii_rc_data=o_ii_rc_data, $
IDL>                               h_i_aeff_data=h_i_aeff_data)
IDL> print, 'Emissivity:', emiss_o_ii
   Emissivity:   5.9047319e-27
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for O II lines from
Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

Adopted from MIDAS script Roii.prg written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MO-
CASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_o_ii_rl()
for calculating line emissivities and separated it from calc_abund_o_ii_rl().

Version

0.3.0

calc_emissivity.pro

CALC_EMISSIVITY

This function calculates line emissivities for specified ion with level(s) by solving atomic level populations and in statistical equilibrium for given electron density and temperature.

```
result = calc_emissivity(temperature=float, density=float, atomic_levels=string, elj_data
   =array/object, omij_data=array/object, aij_data=array/object)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
: electron temperature

density IN REQUIRED TYPE=float
: electron density

atomic_levels REQUIRED TYPE=string
: level(s) e.g '1,2/', '1,2,1,3/'

elj_data IN REQUIRED TYPE=array/object
: energy levels (Ej) data

omij_data IN REQUIRED TYPE=array/object
: collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
: transition probabilities (Aij) data

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='o'
 IDL> ion='iii'
 IDL> o_iii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> o_iii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> o_iii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> atomic_levels='3,4/'
 IDL> emiss5007=double(0.0)
 IDL> emiss5007=calc_emissivity(temperature=temperature, density=density, $
 IDL>                           atomic_levels=atomic_levels, $
 IDL>                           elj_data=o_iii_elj, omij_data=o_iii_omij, $
 IDL>                           aij_data=o_iii_aij
 IDL> print, 'Emissivity(O III 5007):', emiss5007
    Emissivity(O III 5007):   3.6041012e-21
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

21/11/2016, A. Danehkar, Made a new function calc_emissivity() for calculating line emissivities and separated it from calc_abundance().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_emissivity().

27/06/2019, A. Danehkar, Use the simplified calc_populations() routine.

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

calc_populations.pro

CALC_POPULATIONS

This function solves atomic level populations in statistical equilibrium for given electron temperature and density.

```
result = calc_populations(temperature=float, density=float, elj_data=array/object, omij_data
   =array/object, aij_data=array/object, eff_Omij=array/object, level_num=int, irats=int)
```

Returns

type=array/object. This function returns the atomic level populations.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

elj_data IN REQUIRED TYPE=array/object
energy levels (Ej) data

omij_data IN REQUIRED TYPE=array/object
collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
transition probabilities (Aij) data

eff_Omij IN TYPE=array/object
effective collision strengths (Omij_T) at given temperature

level_num IN TYPE=int
Number of levels

irats IN TYPE=int
Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='s'
 IDL> ion='ii'
 IDL> s_ii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> s_ii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> s_ii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)\
 IDL> density = double(1000)
 IDL> temperature=double(10000.0);
 IDL> Nlj=calc_populations(temperature=temperature, density=density, $
 IDL>                      elj_data=s_ii_elj, omij_data=s_ii_omij, $
 IDL>                      aij_data=s_ii_aij)
 IDL> print, 'Atomic Level Populations:', Nlj
    Atomic Level Populations:    0.96992832    0.0070036315     0.023062261   2.6593671e-06   3.1277019e
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LA-PACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_populations().

27/02/2019, A. Danehkar, Simplify the calc_populations() routine for external usage.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

calc_temperature.pro

CALC_TEMPERATURE

This function determines electron temperature from given flux intensity ratio for specified ion with upper level(s) lower level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron density.

```
result = calc_temperature(line_flux_ratio=float, density=float, upper_levels=string,
   lower_levels=string, elj_data=array/object, omij_data=array/object, aij_data=array/
   object [, low_temperature=float] [, high_temperature=float] [, num_temperature=integer
   ] [, min_density=float])
```

Returns

type=double. This function returns the electron temperature.

Keywords

line_flux_ratio IN REQUIRED TYPE=float
flux intensity ratio

density IN REQUIRED TYPE=float
electron density

upper_levels IN REQUIRED TYPE=string
upper atomic level(s) e.g '1,2/', '1,2,1,3/'

lower_levels IN REQUIRED TYPE=string
lower atomic level(s) e.g '1,2/', '1,2,1,3/'

elj_data IN REQUIRED TYPE=array/object
energy levels (Ej) data

omij_data IN REQUIRED TYPE=array/object
collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
transition probabilities (Aij) data

low_temperature IN OPTIONAL TYPE=float
lower temperature range

high_temperature IN OPTIONAL TYPE=float
upper temperature range

num_temperature IN OPTIONAL TYPE=integer
number of the iteration step

min_density IN OPTIONAL TYPE=float
lower density range

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='s'
 IDL> ion='ii'
 IDL> s_ii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> s_ii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> s_ii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)
 IDL> upper_levels='1,2,1,3/'
 IDL> lower_levels='1,5/'
 IDL> density = double(2550)
 IDL> line_flux_ratio=double(10.753)
 IDL> temperature=calc_temperature(line_flux_ratio=line_flux_ratio, density=density, $
 IDL>                             upper_levels=upper_levels, lower_levels=lower_levels, $
 IDL>                             elj_data=s_ii_elj, omij_data=s_ii_omij, $
 IDL>                             aij_data=s_ii_aij)
 IDL> print, "Electron Temperature:", temperature
    Electron Temperature:       7920.2865
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_temperature().

27/02/2019, A. Danehkar, Fix a bug in the atomic level assumption, and use the simplified calc_populations() routine.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

24/05/2019, A. Danehkar, Add the optional temperature range.

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.3.0

collision__define.pro

Class description for collision

Inheritance

- ION_UNIT

Fields

ATOM_AIJ_FILE ''
ATOM_ELJ_FILE ''
ATOM_OMIJ_FILE ''
ATOM_RC_SH95_FILE ''
DATA_AIJ PTR_NEW()
DATA_DIR ''
DATA_ELJ PTR_NEW()
DATA_OMIJ PTR_NEW()
DATA_RC_DIR ''
HI_RC_DATA PTR_NEW()
LEVEL 0L

Fields in ION_UNIT

ATOM ''
BASE_DIR ''
ION ''

COLLISION::INIT

"Unit for Collisionally Excited Lines": This obejct library can be used to determine electron temperature, electron density, ionic abundance from the observed flux of collisionally excited lines (CEL) for specified ion with level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron density and temperature.

```
result = collision::init()
```

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> upper_levels='1,2,1,3/'
 IDL> lower_levels='1,5/'
 IDL> density = double(2550)
 IDL> line_flux_ratio=double(10.753)
 IDL> temperature=s2->calc_temperature(line_flux_ratio=line_flux_ratio, density=density, $
 IDL>   upper_levels=upper_levels, lower_levels=lower_levels)
 IDL> print, "Electron Temperature:", temperature
    Electron Temperature:       7920.2865

 IDL> upper_levels='1,2/'
 IDL> lower_levels='1,3/'
 IDL> diagtype='D'
 IDL> temperature=double(7000.0);
 IDL> line_flux_ratio=double(1.506);
 IDL> density=s2->calc_density(line_flux_ratio=line_flux_ratio, temperature=temperature, $
 IDL>   upper_levels=upper_levels, lower_levels=lower_levels)
 IDL> print, "Electron Density:", density
    Electron Density:       2312.6164

 IDL> density = double(1000)
 IDL> temperature=double(10000.0);
 IDL> Nlj=s2->calc_populations(temperature=temperature, density=density)
 IDL> print, 'Atomic Level Populations:', Nlj
    Atomic Level Populations:      0.96992796    0.0070037404     0.023062517   2.6594158e-06   3.127759

 IDL> temperature=double(10000.0)
 IDL> N_crit=s2->calc_crit_density(temperature=temperature)
 IDL> print, 'Critical Densities:', N_crit
    Critical Densities:       0.0000000       5007.8396       1732.8414       1072685.0       2220758.1

 IDL> temperature=double(10000.0)
 IDL> Omij_T=s2->get_omij_temp(temperature=temperature)
 IDL> print, 'Effective Collision Strengths: '
 IDL> print, Omij_T
    Effective Collision Strengths:
    0.0000000       0.0000000       0.0000000       0.0000000       0.0000000
    2.7800000       0.0000000       0.0000000       0.0000000       0.0000000
    4.1600000       7.4600000       0.0000000       0.0000000       0.0000000
```

```
   1.1700000       1.8000000       2.2000000       0.0000000       0.0000000
   2.3500000       3.0000000       4.9900000       2.7100000       0.0000000

IDL> s2->print_ionic, temperature=temperature, density=density
   Temperature =   10000.0 K
   Density =    1000.0 cm-3

   Level    Populations   Critical Densities
   Level 1:   9.699E-01   0.000E+00
   Level 2:   7.004E-03   5.008E+03
   Level 3:   2.306E-02   1.733E+03
   Level 4:   2.659E-06   1.073E+06
   Level 5:   3.128E-06   2.221E+06

   1.231E-03
   6732.69A
   (2-->1)
   2.544E-20

   3.338E-04   3.452E-07
   6718.31A     314.47um
   (3-->1)     (3-->2)
   2.276E-20   5.029E-26

   1.076E-01   1.812E-01   7.506E-02
   4077.51A       1.03um       1.04um
   (4-->1)     (4-->2)     (4-->3)
   1.394E-21   9.258E-22   3.823E-22

   2.670E-01   1.644E-01   1.938E-01   0.000E+00
   4069.76A       1.03um       1.03um     214.14um
   (5-->1)     (5-->2)     (5-->3)     (5-->4)
   4.076E-21   9.927E-22   1.166E-21   0.000E+00

   H-beta emissivity: 1.237E-25 N(H+) Ne  [erg/s]

IDL> o3=obj_new('collision')
IDL> o3->set,['o','iii']
IDL>
IDL> levels5007='3,4/'
IDL> temperature=double(10000.0)
IDL> density=double(5000.0)
IDL> iobs5007=double(1200.0)
IDL> Abb5007=double(0.0)
IDL>
IDL> emis=o3->calc_emissivity(temperature=temperature, density=density, $
IDL>   atomic_levels=levels5007)
IDL> print, 'Emissivity(O III 5007):', emis
   Emissivity(O III 5007):   3.6041012e-21

IDL> Abb5007=o3->calc_abundance(temperature=temperature, density=density, $
IDL>   line_flux=iobs5007, atomic_levels=levels5007)
IDL> print, 'N(O^2+)/N(H+):', Abb5007
   N(O^2+)/N(H+):   0.00041256231

IDL> Nlj=o3->calc_populations(temperature=temperature, density=density)
IDL> print, 'Atomic Level Populations:', Nlj
   Atomic Level Populations:      0.15564960      0.42689831      0.41723001   0.00022205964   1.522458

IDL> N_crit=o3->calc_crit_density(temperature=temperature)
IDL> print, 'Critical Densities:', N_crit
   Critical Densities:       0.0000000       490.78115       3419.4864       685276.77       25472367.

IDL> temperature=double(10000.0)
IDL> Omij_T=o3->get_omij_temp(temperature=temperature, level_num=8)
IDL> print, 'Effective Collision Strengths: '
IDL> print, Omij_T
   Effective Collision Strengths:
   0.0000000       0.0000000       0.0000000       0.0000000       0.0000000       0.0000000       0.00
   0.54300000       0.0000000       0.0000000       0.0000000       0.0000000       0.0000000       0.0
   0.27000000       1.2900000       0.0000000       0.0000000       0.0000000       0.0000000       0.0
   0.25300000      0.76000000       1.2700000       0.0000000       0.0000000       0.0000000       0.0
   0.032300000     0.097200000      0.16200000      0.57800000       0.0000000       0.0000000       0.
   0.13300000      0.39600000      0.66000000   1.9400000e-05       0.0000000       0.0000000       0.0
   0.098800000       1.6300000      0.89000000      0.72700000    0.0029900000       1.4400000       0.
   0.66000000      0.62900000      0.28100000      0.29400000     0.024200000      0.46200000       1.0

IDL> o3->print_ionic, temperature=temperature, density=density
   Temperature =   10000.0 K
   Density =    5000.0 cm-3

   Level    Populations   Critical Densities
   Level 1:   1.556E-01   0.000E+00
   Level 2:   4.269E-01   4.908E+02
   Level 3:   4.172E-01   3.419E+03
   Level 4:   2.221E-04   6.853E+05
   Level 5:   1.522E-08   2.547E+07

   2.597E-05
   88.34um
   (2-->1)
   4.986E-23
```

```
0.000E+00   9.632E-05
32.66um      51.81um
(3-->1)     (3-->2)
0.000E+00   3.081E-22

2.322E-06   6.791E-03   2.046E-02
4932.60A    4960.29A    5008.24A
(4-->1)     (4-->2)     (4-->3)
4.153E-25   1.208E-21   3.604E-21

0.000E+00   2.255E-01   6.998E-04   1.685E+00
2315.58A    2321.67A    2332.12A    4364.45A
(5-->1)     (5-->2)     (5-->3)     (5-->4)
0.000E+00   5.875E-24   1.815E-26   2.335E-23

H-beta emissivity: 1.239E-25 N(H+) Ne  [erg/s]
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

21/11/2016, A. Danehkar, Made a new function calc_emissivity() for calculating line emissivities and separated it from calc_abundance().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_abundance(), calc_density(), and calc_temperature().

27/02/2019, A. Danehkar, Fix a bug in the atomic level assumption, and use the simplified calc_populations() routine.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

24/05/2019, A. Danehkar, Add the optional density range to calc_density(), and the optional temperature range to calc_temperature().

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::FREE

```
result = collision::free()
```

COLLISION::SET

```
collision::set, atom_ion, level=level
```

Parameters

atom_ion

Keywords

level

COLLISION::CALC_TEMPERATURE

This function determines electron temperature from given flux intensity ratio for specified ion with upper level(s) lower level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron density.

```
result = collision::calc_temperature(line_flux_ratio=float, density=float, upper_levels=
   string, lower_levels=string [, low_temperature=float] [, high_temperature=float] [,
   num_temperature=integer] [, min_density=float])
```

Returns

type=double. This function returns the electron temperature.

Keywords

line_flux_ratio IN REQUIRED TYPE=float
flux intensity ratio

density IN REQUIRED TYPE=float
electron density

upper_levels IN REQUIRED TYPE=string
upper atomic level(s) e.g '1,2/', '1,2,1,3/'

lower_levels IN REQUIRED TYPE=string
lower atomic level(s) e.g '1,2/', '1,2,1,3/'

low_temperature IN OPTIONAL TYPE=float
lower temperature range

high_temperature IN OPTIONAL TYPE=float
upper temperature range

num_temperature IN OPTIONAL TYPE=integer
number of the iteration step

min_density IN OPTIONAL TYPE=float
lower density range

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> upper_levels='1,2,1,3/'
 IDL> lower_levels='1,5/'
 IDL> density = double(2550)
 IDL> line_flux_ratio=double(10.753)
 IDL> temperature=s2->calc_temperature(line_flux_ratio=line_flux_ratio, density=density, $
 IDL>   upper_levels=upper_levels, lower_levels=lower_levels)
 IDL> print, "Electron Temperature:", temperature
    Electron Temperature:       7920.2865
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_temperature().

27/02/2019, A. Danehkar, fix a bug in the atomic level assumption, and use the simplified calc_populations() routine.

04/03/2019, A. Danehkar, use the get_omij_temp() routine.

24/05/2019, A. Danehkar, add the optional temperature range.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::CALC_DENSITY

This function determines electron density from given flux intensity ratio for specified ion with upper level(s) lower level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron temperature.

```
result = collision::calc_density(line_flux_ratio=float, temperature=float, upper_levels=
   string, lower_levels=string [, low_density=float] [, high_density=float] [, num_density
   =integer] [, min_temperature=float])
```

Returns

type=double. This function returns the electron density.

Keywords

line_flux_ratio IN REQUIRED TYPE=float
flux intensity ratio

temperature IN REQUIRED TYPE=float
electron temperature

upper_levels IN REQUIRED TYPE=string
upper atomic level(s) e.g '1,2/', '1,2,1,3/'

lower_levels IN REQUIRED TYPE=string
lower atomic level(s) e.g '1,2/', '1,2,1,3/' transition probabilities (Aij) data

low_density IN OPTIONAL TYPE=float
lower density range

high_density IN OPTIONAL TYPE=float
upper density range

num_density IN OPTIONAL TYPE=integer
number of the iteration step

min_temperature IN OPTIONAL TYPE=float
minimum temperature

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> upper_levels='1,2/'
 IDL> lower_levels='1,3/'
 IDL> diagtype='D'
IDL> temperature=double(7000.0);
IDL> line_flux_ratio=double(1.506);
IDL> density=s2->calc_density(line_flux_ratio=line_flux_ratio, temperature=temperature, $
IDL>   upper_levels=upper_levels, lower_levels=lower_levels)
IDL> print, "Electron Density:", density
   Electron Density:       2312.6164
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_density().

27/02/2019, A. Danehkar, fix a bug in the atomic level assumption, and use the simplified calc_populations() routine.

04/03/2019, A. Danehkar, use the get_omij_temp() routine.

24/05/2019, A. Danehkar, add the optional density range.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::CALC_POPULATIONS

This function solves atomic level populations in statistical equilibrium for given electron temperature and density.

```
result = collision::calc_populations(temperature=float, density=float, eff_Omij=array/
   object, level_num=int, irats=int)
```

Returns

type=array/object. This function returns the atomic level populations.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
: electron density

eff_Omij IN TYPE=array/object
: effective collision strengths (Omij_T) at given temperature

level_num IN TYPE=int
: Number of levels

irats IN TYPE=int
: Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> density = double(1000)
 IDL> temperature=double(10000.0);
 IDL> Nlj=s2->calc_populations(temperature=temperature, density=density)
 IDL> print, 'Atomic Level Populations:', Nlj
    Atomic Level Populations:      0.96992796    0.0070037404     0.023062517   2.6594158e-06   3.127759
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_populations().

27/02/2019, A. Danehkar, Simplify the calc_populations() routine for external usage.

04/03/2019, A. Danehkar, Use the get_omij_temp() routine.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.0.6

COLLISION::CALC_CRIT_DENSITY

This function calculates critical densities in statistical equilibrium for given electron temperature.

```
  result = collision::calc_crit_density(temperature=float, level_num=int, irats=int)
```

Returns

type=array/object. This function returns the critical densities.

Keywords

temperature IN REQUIRED TYPE=float
: electron temperature

level_num IN TYPE=int
: Number of levels

irats IN TYPE=int
: Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> temperature=double(10000.0)
 IDL> N_crit=s2->calc_crit_density(temperature=temperature)
 IDL> print, 'Critical Densities:', N_crit
    Critical Densities:       0.0000000       5007.8396       1732.8414       1072685.0       2220758.1
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_populations().

27/02/2019, A. Danehkar, simplify the calc_populations() routine for external usage.

01/03/2019, A. Danehkar, create the calc_crit_density() routine from the calc_populations() routine.

04/03/2019, A. Danehkar, use the get_omij_temp() routine.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::CALC_EMISSIVITY

This function calculates line emissivities for specified ion with level(s) by solving atomic level populations and in statistical equilibrium for given electron density and temperature.

```
result = collision::calc_emissivity(temperature=float, density=float, atomic_levels=
   string)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

atomic_levels REQUIRED TYPE=string
level(s) e.g '1,2/', '1,2,1,3/'

Examples

For example:

```
IDL> o3=obj_new('collision')
 IDL> o3->set,['o','iii']
 IDL>
 IDL> levels5007='3,4/'
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> iobs5007=double(1200.0)
 IDL> Abb5007=double(0.0)
 IDL>
 IDL> emis=o3->calc_emissivity(temperature=temperature, density=density, $
 IDL>   atomic_levels=levels5007)
 IDL> print, 'Emissivity(O III 5007):', emis
```

```
    Emissivity(O III 5007):   3.6041012e-21
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

21/11/2016, A. Danehkar, Made a new function calc_emissivity() for calculating line emissivities and separated it from calc_abundance().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_emissivity().

27/06/2019, A. Danehkar, use the simplified calc_populations() routine.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::CALC_ABUNDANCE

This function determines the ionic abundance from the observed flux intensity for specified ion with level(s) by solving atomic level populations and line emissivities in statistical equilibrium for given electron density and temperature.

```
result = collision::calc_abundance(temperature=float, density=float, line_flux=float,
    atomic_levels=string)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float

electron temperature

density IN REQUIRED TYPE=float

electron density

line_flux IN REQUIRED TYPE=float

line flux intensity

atomic_levels IN REQUIRED TYPE=string
level(s) e.g '1,2/', '1,2,1,3/'

Examples

For example:

```
IDL> o3=obj_new('collision')
 IDL> o3->set,['o','iii']
 IDL>
 IDL> levels5007='3,4/'
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> iobs5007=double(1200.0)
 IDL> Abb5007=double(0.0)
 IDL>
 IDL> Abb5007=o3->calc_abundance(temperature=temperature, density=density, $
 IDL>   line_flux=iobs5007, atomic_levels=levels5007)
 IDL> print, 'N(O^2+)/N(H+):', Abb5007
    N(O^2+)/N(H+):   0.00041256231
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

15/09/2013, A. Danehkar, Translated from FORTRAN to IDL code.

20/10/2016, A. Danehkar, Replaced str2int with strnumber.

20/10/2016, A. Danehkar, Replaced CFY, SPLMAT, and CFD with IDL function INTERPOL(/SPLINE).

20/10/2016, A. Danehkar, Replaced LUSLV with IDL LAPACK function LA_LINEAR_EQUATION.

15/11/2016, A. Danehkar, Replaced LA_LINEAR_EQUATION (not work in GDL) with IDL function LUDC & LUSOL.

19/11/2016, A. Danehkar, Replaced INTERPOL (not accurate) with SPL_INIT & SPL_INTERP.

20/11/2016, A. Danehkar, Made a new function calc_populations() for solving atomic level populations and separated it from calc_abundance(), calc_density() and calc_temperature().

21/11/2016, A. Danehkar, Made a new function calc_emissivity() for calculating line emissivities and separated it from calc_abundance().

10/03/2017, A. Danehkar, Integration with AtomNeb, now uses atomic data input elj_data, omij_data, aij_data.

12/06/2017, A. Danehkar, Cleaning the function, and remove unused varibales from calc_abundance().

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

FORTRAN HISTORY:

03/05/1981, I.D.Howarth, Version 1.

05/05/1981, I.D.Howarth, Minibug fixed!

07/05/1981, I.D.Howarth, Now takes collision rates or strengths.

03/08/1981, S.Adams, Interpolates collision strengths.

07/08/1981, S.Adams, Input method changed.

19/11/1984, R.E.S.Clegg, SA files entombed in scratch disk. Logical filenames given to SA's data files.

08/1995, D.P.Ruffle, Changed input file format. Increased matrices.

02/1996, X.W.Liu, Tidy up. SUBROUTINES SPLMAT, HGEN, CFY and CFD modified such that matrix sizes (i.e. maximum of Te and maximum no of levels) can now be cha by modifying the parameters NDIM1, NDIM2 and N in the Main program. EASY! Now takes collision rates as well. All variables are declared explicitly Generate two extra files (ionpop.lis and ionra of plain stream format for plotting.

06/1996, C.J.Pritchet, Changed input data format for cases IBIG=1,2. Fixed readin bug for IBIG=2 case. Now reads reformatted upsilons (easier to see and the 0 0 0 data end is excluded for these c The A values have a different format for IBIG=.

2006, B.Ercolano, Converted to F90.

Version

0.2.0

COLLISION::PRINT_IONIC

This function prints the atom's transitions information, atomic level populations, critical densities, and emissivities for given temperature and density.

```
collision::print_ionic, temperature=float, density=float, /printEmissivity, /printPopulations
   , /printCritDensity
```

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

printEmissivity IN TYPE=boolean
Set for printing Emissivities

printPopulations IN TYPE=boolean
Set for printing Populations

printCritDensity IN TYPE=boolean
Set for printing Critical Densities

Examples

For example:

```
IDL> o3=obj_new('collision')
 IDL> o3->set,['o','iii']
 IDL>
 IDL> levels5007='3,4/'
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL> iobs5007=double(1200.0)
 IDL> Abb5007=double(0.0)
 IDL>
 IDL> o3->print_ionic, temperature=temperature, density=density
    Temperature =   10000.0 K
    Density =    5000.0 cm-3

    Level    Populations   Critical Densities
    Level 1:   1.556E-01   0.000E+00
    Level 2:   4.269E-01   4.908E+02
    Level 3:   4.172E-01   3.419E+03
    Level 4:   2.221E-04   6.853E+05
    Level 5:   1.522E-08   2.547E+07

    2.597E-05
    88.34um
    (2-->1)
    4.986E-23

    0.000E+00   9.632E-05
```

```
32.66um      51.81um
(3-->1)     (3-->2)
0.000E+00   3.081E-22

2.322E-06   6.791E-03   2.046E-02
4932.60A    4960.29A    5008.24A
(4-->1)     (4-->2)     (4-->3)
4.153E-25   1.208E-21   3.604E-21

0.000E+00   2.255E-01   6.998E-04   1.685E+00
2315.58A    2321.67A    2332.12A    4364.45A
(5-->1)     (5-->2)     (5-->3)     (5-->4)
0.000E+00   5.875E-24   1.815E-26   2.335E-23

H-beta emissivity: 1.239E-25 N(H+) Ne  [erg/s]
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

04/03/2019, A. Danehkar, create the print_ionic() routine.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

COLLISION::GET_OMIJ_TEMP

This function derives the effective collision strengths (Omij_T) from the collision strengths (omega_ij) data for the given temperature.

```
result = collision::get_omij_temp(temperature=float, level_num=int, irats=int)
```

Returns

type=array/object. This function returns the effective collision strengths (Omij_T).

Keywords

temperature IN REQUIRED TYPE=float
    electron temperature

level_num IN TYPE=int
    Number of levels

irats IN TYPE=int
    Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> s2=obj_new('collision')
 IDL> s2->set,['s','ii']
 IDL>
 IDL> temperature=double(10000.0)
 IDL> Omij_T=s2->get_omij_temp(temperature=temperature)
 IDL> print, 'Effective Collision Strengths: '
 IDL> print, Omij_T
    Effective Collision Strengths:
    0.0000000       0.0000000       0.0000000       0.0000000       0.0000000
    2.7800000       0.0000000       0.0000000       0.0000000       0.0000000
    4.1600000       7.4600000       0.0000000       0.0000000       0.0000000
    1.1700000       1.8000000       2.2000000       0.0000000       0.0000000
    2.3500000       3.0000000       4.9900000       2.7100000       0.0000000
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

04/03/2019, A. Danehkar, create the get_omij_temp() routine.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.3.0

COLLISION::SET_LEVEL_NUM

```
collision::set_level_num, level_num
```

Parameters

level_num

COLLISION::GET_LEVEL_NUM

```
result = collision::get_level_num()
```

COLLISION::SET_DATA_DIR

```
collision::set_data_dir, data_dir
```

Parameters

data_dir

COLLISION::GET_DATA_DIR

```
result = collision::get_data_dir()
```

COLLISION::SET_DATA_RC_DIR

```
collision::set_data_rc_dir, data_rc_dir
```

Parameters

data_rc_dir

COLLISION::GET_DATA_RC_DIR

```
result = collision::get_data_rc_dir()
```

COLLISION::SET_ATOM_ELJ_FILE

```
collision::set_Atom_Elj_file, Atom_Elj_file
```

Parameters

Atom_Elj_file

COLLISION::GET_ATOM_ELJ_FILE

```
result = collision::get_Atom_Elj_file()
```

COLLISION::SET_ATOM_OMIJ_FILE

```
collision::set_Atom_Omij_file, Atom_Omij_file
```

Parameters

Atom_Omij_file

COLLISION::GET_ATOM_OMIJ_FILE

```
result = collision::get_Atom_Omij_file()
```

COLLISION::SET_ATOM_AIJ_FILE

```
collision::set_Atom_Aij_file, Atom_Aij_file
```

Parameters

Atom_Aij_file

COLLISION::GET_ATOM_AIJ_FILE

```
result = collision::get_Atom_Aij_file()
```

COLLISION::SET_ATOM_RC_SH95_FILE

```
collision::set_Atom_RC_SH95_file, Atom_RC_SH95_file
```

Parameters

Atom_RC_SH95_file

COLLISION::GET_ATOM_RC_SH95_FILE

```
result = collision::get_Atom_RC_SH95_file()
```

COLLISION__DEFINE

```
collision__define
```

deredden_flux.pro

DEREDDEN_FLUX

This function dereddens absolute flux intensity based on the reddening law.

```
result = deredden_flux(wavelength, flux, m_ext [, ext_law=string] [, rv=float] [, fmlaw=
   string])
```

Returns

type=double. This function returns the deredden flux intensity.

Parameters

wavelength IN REQUIRED TYPE=float/array
Wavelength in Angstrom

flux IN REQUIRED TYPE=float
absolute flux intensity

m_ext IN REQUIRED TYPE=float
logarithmic extinction

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL
the extinction law:
'GAL' for Howarth Galactic.
'GAL2' for Savage and Mathis.
'CCM' for CCM galactic.
'JBK' for Whitford, Seaton, Kaler.
'FM' for Fitxpatrick.
'SMC' for Prevot SMC.
'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1
the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:
'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).
'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).
'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> ext_law='GAL'
 IDL> R_V=3.1
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> flux_deredden=deredden_flux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V) ; deredden absolute
 IDL> print, 'dereddened flux(6563):', flux_deredden
    dereddened flux(6563):       4.7847785
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

31/08/2012, A. Danehkar, IDL code.

Version

0.3.0

deredden_relflux.pro

DEREDDEN_RELFLUX

This function dereddens flux intensity relative to Hb=100, based on the reddening law.

```
result = deredden_relflux(wavelength, relflux, m_ext [, ext_law=string] [, rv=float] [,
   fmlaw=string])
```

Returns

type=double. This function returns the deredden flux intensity relative to Hb=100.

Parameters

wavelength IN REQUIRED TYPE=float/array

Wavelength in Angstrom

relflux IN REQUIRED TYPE=float

flux intensity relative to Hb=100

m_ext IN REQUIRED TYPE=float

logarithmic extinction

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL

the extinction law:

'GAL' for Howarth Galactic.

'GAL2' for Savage and Mathis.

'CCM' for CCM galactic.

'JBK' for Whitford, Seaton, Kaler.

'FM' for Fitxpatrick.

'SMC' for Prevot SMC.

'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> ext_law='GAL'
 IDL> R_V=3.1
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> flux_deredden=deredden_relflux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V) ; deredden absolu
 IDL> print, 'dereddened relative flux(6563):', flux_deredden
    dereddened relative flux(6563):       0.47847785
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

31/08/2012, A. Danehkar, IDL code.

Version

0.3.0

find_aeff_sh95_column.pro

FIND_AEFF_SH95_COLUMN

private

This function locates and returns the data location of the given low energy level, high energy level, and the level number within the database of H I emissivities given by from Storey & Hummer, 1995MNRAS.272...41S.

```
  result = find_aeff_sh95_column(lo_lev, hi_lev, lev_num)
```

Returns

type=double. This function returns the data location .

Parameters

lo_lev IN REQUIRED TYPE=float
low energy level

hi_lev IN REQUIRED TYPE=float
high energy level

lev_num IN REQUIRED TYPE=float
level number

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on H I emissivities from Storey & Hummer, 1995MN-RAS.272...41S.

25/08/2012, A. Danehkar, IDL code written.

11/03/2017, A. Danehkar, Integration with AtomNeb.

Version

0.3.0

get_omij_temp.pro

GET_OMIJ_TEMP

This function derives the effective collision strengths (Omij_T) from the collision strengths (omega_ij) data for the given temperature.

```
result = get_omij_temp(temperature=float, omij_data=array/object, elj_data=elj_data,
   level_num=int, irats=int)
```

Returns

type=array/object. This function returns the effective collision strengths (Omij_T).

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

omij_data IN REQUIRED TYPE=array/object
collision strengths (omega_ij) data

elj_data

level_num IN TYPE=int

Number of levels

irats IN TYPE=int

Else Coll. rates = tabulated values * 10 ** irats

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> atom='s'
 IDL> ion='ii'
 IDL> s_ii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
 IDL> s_ii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
 IDL> s_ii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)\
 IDL> temperature=double(10000.0);
 IDL> Omij_T=get_omij_temp(temperature=temperature, omij_data=s_ii_omij)
 IDL> print, 'Effective Collision Strengths: '
 IDL> print, Omij_T
    Effective Collision Strengths:
    0.0000000       0.0000000       0.0000000       0.0000000       0.0000000
    2.7800000       0.0000000       0.0000000       0.0000000       0.0000000
    4.1600000       7.4600000       0.0000000       0.0000000       0.0000000
    1.1700000       1.8000000       2.2000000       0.0000000       0.0000000
    2.3500000       3.0000000       4.9900000       2.7100000       0.0000000
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

04/03/2019, A. Danehkar, create the get_omij_temp() routine.

Version

0.3.0

print_ionic.pro

PRINT_IONIC

This function prints the atom's transitions information, atomic level populations, critical densities, and emissivities for given temperature and density.

```
print_ionic, temperature=float, density=float, elj_data=array/object, omij_data=array/
   object, aij_data=array/object, h_i_aeff_data=array/object, /printEmissivity, /printPopulations
   , /printCritDensity
```

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

elj_data IN REQUIRED TYPE=array/object
energy levels (Ej) data

omij_data IN REQUIRED TYPE=array/object
collision strengths (omega_ij) data

aij_data IN REQUIRED TYPE=array/object
transition probabilities (Aij) data

h_i_aeff_data IN TYPE=array/object
H I recombination coefficients

printEmissivity IN TYPE=boolean
Set for printing Emissivities

printPopulations IN TYPE=boolean
Set for printing Populations

printCritDensity IN TYPE=boolean
Set for printing Critical Densities

Examples

For example:

```
IDL> base_dir = file_dirname(file_dirname((routine_info('$MAIN$', /source)).path))
 IDL> data_dir = ['atomic-data', 'chianti70']
 IDL> Atom_Elj_file = filepath('AtomElj.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Omij_file = filepath('AtomOmij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> Atom_Aij_file = filepath('AtomAij.fits', root_dir=base_dir, subdir=data_dir )
 IDL> data_rc_dir = ['atomic-data-rc']
 IDL> Atom_RC_SH95_file= filepath('rc_SH95.fits', root_dir=base_dir, subdir=data_rc_dir )
IDL> atom='o'
IDL> ion='iii'
IDL> o_iii_elj=atomneb_read_elj(Atom_Elj_file, atom, ion, level_num=5) ; read Energy Levels (Ej)
IDL> o_iii_omij=atomneb_read_omij(Atom_Omij_file, atom, ion) ; read Collision Strengths (Omegaij)
IDL> o_iii_aij=atomneb_read_aij(Atom_Aij_file, atom, ion) ; read Transition Probabilities (Aij)
IDL> atom='h'
IDL> ion='ii' ; H I
IDL> hi_rc_data=atomneb_read_aeff_sh95(Atom_RC_SH95_file, atom, ion)
IDL> temperature=double(10000.0);
IDL> density = double(1000.)
IDL> print_ionic, temperature=temperature, density=density, $, $
IDL>              elj_data=o_iii_elj, omij_data=o_iii_omij, $
IDL>              aij_data=o_iii_aij, h_i_aeff_data=hi_rc_data[0].Aeff
   Temperature =   10000.0 K
   Density =    1000.0 cm-3

   Level    Populations   Critical Densities
   Level 1:   3.063E-01   0.000E+00
   Level 2:   4.896E-01   4.908E+02
   Level 3:   2.041E-01   3.419E+03
   Level 4:   4.427E-05   6.853E+05
   Level 5:   2.985E-09   2.547E+07

    2.597E-05
        88.34um
       (2-->1)
    2.859E-22

    0.000E+00   9.632E-05
        32.66um      51.81um
       (3-->1)     (3-->2)
    0.000E+00   7.536E-22

    2.322E-06   6.791E-03   2.046E-02
      4932.60A    4960.29A    5008.24A
       (4-->1)     (4-->2)     (4-->3)
    4.140E-25   1.204E-21   3.593E-21

    0.000E+00   2.255E-01   6.998E-04   1.685E+00
      2315.58A    2321.67A    2332.12A    4364.45A
       (5-->1)     (5-->2)     (5-->3)     (5-->4)
    0.000E+00   5.759E-24   1.779E-26   2.289E-23

   H-beta emissivity: 1.237E-25 N(H+) Ne  [erg/s]
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

04/03/2019, A. Danehkar, create the print_ionic() routine.

Version

0.3.0

recombination__define.pro

Class description for recombination

Inheritance

- ION_UNIT

Fields

ATOM_RC_ALL_FILE ''

ATOM_RC_HE_I_FILE ''

ATOM_RC_N_II_FSL13_FILE ''

ATOM_RC_O_II_SSB17_FILE ''

ATOM_RC_PPB91_FILE ''

ATOM_RC_SH95_FILE ''

DATA_RC_DIR ''

HI_RC_DATA PTR_NEW()

LEVEL 0L

RC_DATA PTR_NEW()

RC_DATA_BR PTR_NEW()

Fields in ION_UNIT

ATOM ''

BASE_DIR ''

ION ''

RECOMBINATION::INIT

"Unit for Recombination Lines": This obejct library can be used to determine the ionic abundance from the observed flux of recombination lines (RL) by using the recombination coefficients.

```
  result = recombination::init()
```

Examples

For example:

```
IDL> he1=obj_new('recombination')
 IDL> he1->set,['he','ii'] ; He I
 IDL>
 IDL> he2=obj_new('recombination')
 IDL> he2->set,['he','iii'] ; He II
 IDL>
 IDL> c2=obj_new('recombination')
 IDL> c2->set,['c','iii'] ; C II
 IDL>
 IDL> c3=obj_new('recombination')
 IDL> c3->set,['c','iv'] ; C III
 IDL>
 IDL> n2=obj_new('recombination')
 IDL> n2->set,['n','iii'] ; N II
 IDL>
 IDL> n3=obj_new('recombination')
 IDL> n3->set,['n','iv'] ; N III
 IDL>
 IDL> o2=obj_new('recombination')
 IDL> o2->set,['o','iii'] ; O II
 IDL>
 IDL> ne2=obj_new('recombination')
 IDL> ne2->set,['ne','iii'] ; Ne II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> ; 4120.84: linenum=7
 IDL> ; 4387.93: linenum=8
 IDL> ; 4437.55: linenum=9
 IDL> ; 4471.50: linenum=10
 IDL> ; 4921.93: linenum=12
 IDL> ; 5015.68: linenum=13
 IDL> ; 5047.74: linenum=14
 IDL> ; 5875.66: linenum=15
 IDL> ; 6678.16: linenum=16
IDL> ; 7065.25: linenum=17
IDL> ; 7281.35: linenum=18
IDL> linenum=10; 4471.50
IDL> emiss_he_i=he1->calc_emissivity(temperature=temperature, density=density, linenum=linenum)
IDL> print, 'Emissivity:', emiss_he_i
   Emissivity:   6.3822830e-26

IDL> he_i_4471_flux= 2.104
IDL> Abund_he_i=he1->calc_abundance(temperature=temperature, density=density, $
IDL>                               linenum=linenum, line_flux=he_i_4471_flux)
IDL> print, 'N(He^+)/N(H^+):', Abund_he_i
   N(He^+)/N(H^+):     0.040848393

IDL> emiss_he_ii=he2->calc_emissivity(temperature=temperature, density=density)
IDL> print, 'Emissivity:', emiss_he_ii
   Emissivity:   1.4989134e-24

IDL> he_ii_4686_flux = 135.833
IDL> Abund_he_ii=he2->calc_abundance(temperature=temperature, density=density, $
IDL>                                 line_flux=he_ii_4686_flux)
IDL> print, 'N(He^2+)/N(H^+):', Abund_he_ii
   N(He^2+)/N(H^+):      0.11228817

IDL> wavelength=6151.43
IDL> emiss_c_ii=c2->calc_emissivity(temperature=temperature, density=density, $
IDL>                                   wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_c_ii
   Emissivity:   5.4719511e-26

IDL> c_ii_6151_flux = 0.028
IDL> Abund_c_ii=c2->calc_abundance(temperature=temperature, density=density, $
IDL>                               wavelength=wavelength, line_flux=c_ii_6151_flux)
IDL> print, 'N(C^2+)/N(H+):', Abund_c_ii
   N(C^2+)/N(H+):   0.00063404650

IDL> wavelength=4647.42
IDL> emiss_c_iii=c3->calc_emissivity(temperature=temperature, density=density, $
IDL>                                     wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_c_iii
   Emissivity:   7.5749632e-25

IDL> c_iii_4647_flux = 0.107
IDL> Abund_c_iii=c3->calc_abundance(temperature=temperature, density=density, $
IDL>                                 wavelength=wavelength, line_flux=c_iii_4647_flux)
IDL> print, 'N(C^3+)/N(H+):', Abund_c_iii
   N(C^3+)/N(H+):   0.00017502840
```

```
IDL> wavelength=4442.02
IDL> emiss_n_ii=n2->calc_emissivity(temperature=temperature, density=density, $
IDL>                                   wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_n_ii
   Emissivity:   3.0397397e-26

IDL> n_ii_4442_flux = 0.017
IDL> Abund_n_ii=n2->calc_abundance(temperature=temperature, density=density, $
IDL>                               wavelength=wavelength, line_flux=n_ii_4442_flux)
IDL> print, 'N(N^2+)/N(H+):', Abund_n_ii
   N(N^2+)/N(H+):   0.00069297541

IDL> wavelength=4640.64
IDL> emiss_n_iii=n3->calc_emissivity(temperature=temperature, density=density, $
IDL>                                     wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_n_iii
   Emissivity:   4.7908644e-24

IDL> n_iii_4641_flux = 0.245
IDL> Abund_n_iii=n3->calc_abundance(temperature=temperature, density=density, $
IDL>                                 wavelength=wavelength, line_flux=n_iii_4641_flux)
IDL> print, 'N(N^3+)/N(H+):', Abund_n_iii
   N(N^3+)/N(H+):   6.3366175e-05

IDL> wavelength=4613.68
IDL> emiss_o_ii=o2->calc_emissivity(temperature=temperature, density=density, $
IDL>                                   wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_o_ii
   Emissivity:   5.9047319e-27

IDL> o_ii_4614_flux = 0.009
IDL> Abund_o_ii=o2->calc_abundance(temperature=temperature, density=density, $
IDL>                               wavelength=wavelength, line_flux=o_ii_4614_flux)
IDL> print, 'N(O^2+)/N(H+):', Abund_o_ii
   N(O^2+)/N(H+):    0.0018886330

IDL> wavelength=3777.14
IDL> emiss_ne_ii=ne2->calc_emissivity(temperature=temperature, density=density, $
IDL>                                     wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_ne_ii
   Emissivity:   1.5996881e-25

IDL> ne_ii_3777_flux = 0.056
IDL> Abund_ne_ii=ne2->calc_abundance(temperature=temperature, density=density, $
IDL>                                 wavelength=wavelength, line_flux=ne_ii_3777_flux)
```

```
    IDL> print, 'N(Ne^2+)/N(H+):', Abund_ne_ii
       N(Ne^2+)/N(H+):   0.00043376850
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

calc_abund_n_ii_rl(), calc_abund_o_ii_rl(), calc_abund_ne_ii_rl() and calc_abund_c_ii_rl() are mostly based on scripts by Yong Zhang added to MOCASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E. and MIDAS script written by X.W.Liu.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::FREE

```
result = recombination::free()
```

RECOMBINATION::SET

```
recombination::set, atom_ion, new=new, wavelength_list=wavelength_list
```

Parameters

atom_ion

Keywords

new

wavelength_list

RECOMBINATION::CALC_ABUNDANCE

```
result = recombination::calc_abundance(temperature=temperature, density=density, wavelength
   =wavelength, linenum=linenum, line_flux=line_flux)
```

Keywords

temperature

density

wavelength

linenum

line_flux

RECOMBINATION::CALC_EMISSIVITY

```
result = recombination::calc_emissivity(temperature=temperature, density=density, wavelength
   =wavelength, linenum=linenum)
```

Keywords

temperature

density

wavelength

linenum

RECOMBINATION::CALC_EMISS_HE_I_RL

This function calculates the emissivity for the given wavelength
of He I recombination line by using the recombination coefficients
from Porter et al. 2012MNRAS.425L..28P.

```
result = recombination::calc_emiss_he_i_rl(temperature=float, density=float, linenum=int)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

linenum IN REQUIRED TYPE=int

Line Number for Wavelength
Wavelength=4120.84:linenum=7,
Wavelength=4387.93: linenum=8,
Wavelength=4437.55: linenum=9,
Wavelength=4471.50: linenum=10,
Wavelength=4921.93: linenum=12,
Wavelength=5015.68: linenum=13,
Wavelength=5047.74: linenum=14,
Wavelength=5875.66: linenum=15,
Wavelength=6678.16: linenum=16,
Wavelength=7065.25: linenum=17,
Wavelength=7281.35: linenum=18.

Examples

For example:

```
IDL> he1=obj_new('recombination')
 IDL> he1->set,['he','ii'] ; He I
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> linenum=10; 4471.50
 IDL> emiss_he_i=he1->calc_emiss_he_i_rl(temperature=temperature, density=density, linenum=linenum)
 IDL> print, 'Emissivity:', emiss_he_i
    Emissivity:   6.3822830e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on improved He I emissivities in the case B from
Porter et al. 2012MNRAS.425L..28P

15/12/2013, A. Danehkar, IDL code written.

20/03/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_i_rl()
for calculating line emissivities and separated it from calc_abund_he_i_rl().

10/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_HE_II_RL

This functioncalculates the emissivity for the He II recombination line 4686 A by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
result = recombination::calc_emiss_he_ii_rl(temperature=float, density=float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

Examples

For example:

```
IDL> he2=obj_new('recombination')
 IDL> he2->set,['he','iii'] ; He II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> emiss_he_ii=he2->calc_emiss_he_ii_rl(temperature=temperature, density=density)
 IDL> print, 'Emissivity:', emiss_he_ii
    Emissivity:   1.4989134e-24
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on He II emissivities from Storey & Hummer, 1995MNRAS.272...41S.

15/12/2013, A. Danehkar, IDL code written.

02/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Change from logarithmic to linear

10/07/2019, A. Danehkar, Made a new function calc_emiss_he_ii_rl()
for calculating line emissivities and separated it from calc_abund_he_ii_rl().

10/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_C_II_RL

This function calculates the emissivity for the given wavelength of C II recombination line by using the recombination coefficients from from Davey et al. (2000) 2000A&AS..142...85D.

```
result = recombination::calc_emiss_c_ii_rl(temperature=float, density=float, wavelength=
   float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> c2=obj_new('recombination')
 IDL> c2->set,['c','iii'] ; C II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> wavelength=6151.43
 IDL> emiss_c_ii=c2->calc_emiss_c_ii_rl(temperature=temperature, density=density, $
IDL>                                  wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_c_ii
   Emissivity:   5.4719511e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for C II lines from
Davey et al. 2000A&AS..142...85D.

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, added to MOCASSIN.

10/05/2013, A. Danehkar, Translated to IDL code.

15/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_ii_rl()
for calculating line emissivities and separated it from calc_abund_c_ii_rl().

10/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_C_III_RL

This function calculates the emissivity for the given wavelength of
C III recombination line by using the recombination coefficients
from Pequignot et al. 1991A&A...251..680P.

```
result = recombination::calc_emiss_c_iii_rl(temperature=float, density=float, wavelength=
    float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> c3=obj_new('recombination')
 IDL> c3->set,['c','iv'] ; C III
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> wavelength=4647.42
 IDL> emiss_c_iii=c3->calc_emiss_c_iii_rl(temperature=temperature, density=density, $
 IDL>                                     wavelength=wavelength)
 IDL> print, 'Emissivity:', emiss_c_iii
    Emissivity:   7.5749632e-25
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for C
III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

18/05/2013, A. Danehkar, Translated to IDL code.

06/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_c_iii_rl()
for calculating line emissivities and separated it from calc_abund_c_iii_rl().

10/07/2019, A. Danehkar, Move to object-oriented program-
ming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_N_II_RL

This function calculates the emissivity for the given wavelength
of N II recombination line by using the recombination coefficients
from Escalante & Victor 1990ApJS...73..513E.

```
  result = recombination::calc_emiss_n_ii_rl(temperature=float, density=float, wavelength=
     float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> n2=obj_new('recombination')
 IDL> n2->set,['n','iii'] ; N II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> wavelength=4442.02
 IDL> emiss_n_ii=n2->calc_emiss_n_ii_rl(temperature=temperature, density=density, $
 IDL>                                  wavelength=wavelength)
 IDL> print, 'Emissivity:', emiss_n_ii
    Emissivity:   3.0397397e-26
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Effective recombination coefficients for N II lines
from Escalante & Victor 1990ApJS...73..513E.

Adopted from MIDAS Rnii script written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MO-
CASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_ii_rl()
for calculating line emissivities and separated it from calc_abund_n_ii_rl().

10/07/2019, A. Danehkar, Move to object-oriented program-
ming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_N_III_RL

This function calculates the emissivity for the given wavelength of
N III recombination line by using the recombination coefficients
from Pequignot et al. 1991A&A...251..680P.

```
result = recombination::calc_emiss_n_iii_rl(temperature=float, density=float, wavelength=
   float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> n3=obj_new('recombination')
 IDL> n3->set,['n','iv'] ; N III
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
IDL> wavelength=4640.64
IDL> emiss_n_iii=n3->calc_emiss_n_iii_rl(temperature=temperature, density=density, $
IDL>                                     wavelength=wavelength)
IDL> print, 'Emissivity:', emiss_n_iii
   Emissivity:   4.7908644e-24
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for N III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

10/05/2013, A. Danehkar, IDL code written.

20/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_n_iii_rl() for calculating line emissivities and separated it from calc_abund_n_iii_rl().

10/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_O_II_RL

This function calculates the emissivity for the given wavelength of O II recombination line by using the recombination coefficients from Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

```
result = recombination::calc_emiss_o_ii_rl(temperature=float, density=float, wavelength=
    float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> o2=obj_new('recombination')
 IDL> o2->set,['o','iii'] ; O II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> wavelength=4613.68
 IDL> emiss_o_ii=o2->calc_emiss_o_ii_rl(temperature=temperature, density=density, $
 IDL>                                   wavelength=wavelength)
 IDL> print, 'Emissivity:', emiss_o_ii
    Emissivity:   5.9047319e-27
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for O II lines from
Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

Adopted from MIDAS script Roii.prg written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MO-
CASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_o_ii_rl()
for calculating line emissivities and separated it from calc_abund_o_ii_rl().

10/07/2019, A. Danehkar, Move to object-oriented program-
ming (OOP).

Version

0.0.3

RECOMBINATION::CALC_EMISS_NE_II_RL

This function calculates the emissivity for the given wavelength of Ne II recombination line by using the recombination coefficients from Kisielius et al. (1998) & Storey (unpublished).

```
result = recombination::calc_emiss_ne_ii_rl(temperature=float, density=float, wavelength=
   float)
```

Returns

type=double. This function returns the line emissivity.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

Examples

For example:

```
IDL> ne2=obj_new('recombination')
 IDL> ne2->set,['ne','iii'] ; Ne II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> wavelength=3777.14
 IDL> emiss_ne_ii=ne2->calc_emiss_ne_ii_rl(temperature=temperature, density=density, $
 IDL>                                      wavelength=wavelength)
 IDL> print, 'Emissivity:', emiss_ne_ii
    Emissivity:   1.5996881e-25
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for Ne II lines from Kisielius et al. 1998A&AS..133..257K & Storey (unpublished).

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, scripts added to MOCASSIN.

14/05/2013, A. Danehkar, Translated to IDL code.

10/04/2017, A. Danehkar, Integration with AtomNeb.

10/07/2019, A. Danehkar, Made a new function calc_emiss_ne_ii_rl() for calculating line emissivities and separated it from calc_abund_ne_ii_rl().

10/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.0.3

RECOMBINATION::CALC_ABUND_HE_I_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of He I recombination line by using the recombination coefficients from Porter et al. 2012MNRAS.425L..28P.

```
result = recombination::calc_abund_he_i_rl(temperature=float, density=float, linenum=int,
    line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

linenum IN REQUIRED TYPE=int
Line Number for Wavelength
Wavelength=4120.84:linenum=7,
Wavelength=4387.93: linenum=8,
Wavelength=4437.55: linenum=9,
Wavelength=4471.50: linenum=10,
Wavelength=4921.93: linenum=12,

Wavelength=5015.68: linenum=13,
Wavelength=5047.74: linenum=14,
Wavelength=5875.66: linenum=15,
Wavelength=6678.16: linenum=16,
Wavelength=7065.25: linenum=17,
Wavelength=7281.35: linenum=18.

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> he1=obj_new('recombination')
 IDL> he1->set,['he','ii'] ; He I
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> ; 4120.84: linenum=7
 IDL> ; 4387.93: linenum=8
 IDL> ; 4437.55: linenum=9
 IDL> ; 4471.50: linenum=10
 IDL> ; 4921.93: linenum=12
 IDL> ; 5015.68: linenum=13
 IDL> ; 5047.74: linenum=14
 IDL> ; 5875.66: linenum=15
 IDL> ; 6678.16: linenum=16
 IDL> ; 7065.25: linenum=17
 IDL> ; 7281.35: linenum=18
 IDL> he_i_4471_flux= 2.104
 IDL> linenum=10; 4471.50
 IDL> Abund_he_i=he1->calc_abundance(temperature=temperature, density=density, $
 IDL>                               linenum=linenum, line_flux=he_i_4471_flux)
 IDL> print, 'N(He^+)/N(H^+):', Abund_he_i
    N(He^+)/N(H^+):     0.040848393
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on improved He I emissivities in the case B from Porter et al. 2012MNRAS.425L..28P

15/12/2013, A. Danehkar, IDL code written.

20/03/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_HE_II_RL

This function determines the ionic abundance from the observed flux intensity for the He II recombination line 4686 A by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
result = recombination::calc_abund_he_ii_rl(temperature=float, density=float, line_flux=
   float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> he2=obj_new('recombination')
 IDL> he2->set,['he','iii'] ; He II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> he_ii_4686_flux = 135.833
IDL> Abund_he_ii=he2->calc_abundance(temperature=temperature, density=density, $
IDL>                                 line_flux=he_ii_4686_flux)
IDL> print, 'N(He^2+)/N(H^+):', Abund_he_ii
   N(He^2+)/N(H^+):      0.11228817
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on He II emissivities from Storey & Hummer, 1995MNRAS.272...41S.

15/12/2013, A. Danehkar, IDL code written.

02/04/2017, A. Danehkar, Integration with AtomNeb.

Version

0.2.0

RECOMBINATION::CALC_ABUND_C_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of C II recombination line by using the recombination coefficients from from Davey et al. (2000) 2000A&AS..142...85D.

```
result = recombination::calc_abund_c_ii_rl(temperature=float, density=float, wavelength=
   float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> c2=obj_new('recombination')
 IDL> c2->set,['c','iii'] ; C II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> c_ii_6151_flux = 0.028
 IDL> wavelength=6151.43
 IDL> Abund_c_ii=c2->calc_abundance(temperature=temperature, density=density, $
 IDL>                               wavelength=wavelength, line_flux=c_ii_6151_flux)
 IDL> print, 'N(C^2+)/N(H+):', Abund_c_ii
    N(C^2+)/N(H+):   0.00063404650
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for C II lines from
Davey et al. 2000A&AS..142...85D.

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, added to MOCASSIN.

10/05/2013, A. Danehkar, Translated to IDL code.

15/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_C_III_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of C III recombination line by using the recombination coefficients from Pequignot et al. 1991A&A...251..680P.

```
result = recombination::calc_abund_c_iii_rl(temperature=float, density=float, wavelength=
    float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> c3=obj_new('recombination')
 IDL> c3->set,['c','iv'] ; C III
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> c_iii_4647_flux = 0.107
 IDL> wavelength=4647.42
 IDL> Abund_c_iii=c3->calc_abundance(temperature=temperature, density=density, $
 IDL>                                 wavelength=wavelength, line_flux=c_iii_4647_flux)
 IDL> print, 'N(C^3+)/N(H+):', Abund_c_iii
    N(C^3+)/N(H+):   0.00017502840
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for C III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

18/05/2013, A. Danehkar, Translated to IDL code.

06/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_N_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of N II recombination line by using the recombination coefficients from Escalante & Victor 1990ApJS...73..513E.

```
result = recombination::calc_abund_n_ii_rl(temperature=float, density=float, wavelength=
   float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> n2=obj_new('recombination')
 IDL> n2->set,['n','iii'] ; N II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> n_ii_4442_flux = 0.017
 IDL> wavelength=4442.02
 IDL> Abund_n_ii=n2->calc_abundance(temperature=temperature, density=density, $
 IDL>                               wavelength=wavelength, line_flux=n_ii_4442_flux)
 IDL> print, 'N(N^2+)/N(H+):', Abund_n_ii
    N(N^2+)/N(H+):   0.00069297541
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Effective recombination coefficients for N II lines from Escalante & Victor 1990ApJS...73..513E.

Adopted from MIDAS Rnii script written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MOCASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_N_III_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of N III recombination line by using the recombination coefficients from Pequignot et al. 1991A&A...251..680P.

```
result = recombination::calc_abund_n_iii_rl(temperature=float, density=float, wavelength=
   float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
: electron temperature

density IN REQUIRED TYPE=float
: electron density

wavelength IN REQUIRED TYPE=float
: Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
: line flux intensity

Examples

For example:

```
IDL> n3=obj_new('recombination')
 IDL> n3->set,['n','iv'] ; N III
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> n_iii_4641_flux = 0.245
 IDL> wavelength=4640.64
 IDL> Abund_n_iii=n3->calc_abundance(temperature=temperature, density=density, $
 IDL>                                 wavelength=wavelength, line_flux=n_iii_4641_flux)
 IDL> print, 'N(N^3+)/N(H+):', Abund_n_iii
    N(N^3+)/N(H+):   6.3366175e-05
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for N III lines from Pequignot, Petitjean, Boisson, C. 1991A&A...251..680P.

10/05/2013, A. Danehkar, IDL code written.

20/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_O_II_RL

This function determines the ionic abundance from the observed
flux intensity for the given wavelength of O II recombination line
by using the recombination coefficients from Storey 1994A&A...282..999S
and Liu et al. 1995MNRAS.272..369L.

```
result = recombination::calc_abund_o_ii_rl(temperature=float, density=float, wavelength=
    float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float
electron temperature

density IN REQUIRED TYPE=float
electron density

wavelength IN REQUIRED TYPE=float
Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float
line flux intensity

Examples

For example:

```
IDL> o2=obj_new('recombination')
 IDL> o2->set,['o','iii'] ; O II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> o_ii_4614_flux = 0.009
 IDL> wavelength=4613.68
 IDL> Abund_o_ii=o2->calc_abundance(temperature=temperature, density=density, $
 IDL>                              wavelength=wavelength, line_flux=o_ii_4614_flux)
 IDL> print, 'N(O^2+)/N(H+):', Abund_o_ii
    N(O^2+)/N(H+):    0.0018886330
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on recombination coefficients for O II lines from Storey 1994A&A...282..999S and Liu et al. 1995MNRAS.272..369L.

Adopted from MIDAS script Roii.prg written by X.W.Liu.

Revised based on scripts by Yong Zhang added to MOCASSIN, 02/2003 Ercolano et al. 2005MNRAS.362.1038E.

10/05/2013, A. Danehkar, Translated to IDL code.

25/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

RECOMBINATION::CALC_ABUND_NE_II_RL

This function determines the ionic abundance from the observed flux intensity for the given wavelength of Ne II recombination line by using the recombination coefficients from Kisielius et al. (1998) & Storey (unpublished).

```
result = recombination::calc_abund_ne_ii_rl(temperature=float, density=float, wavelength=
   float, line_flux=float)
```

Returns

type=double. This function returns the ionic abundanc.

Keywords

temperature IN REQUIRED TYPE=float

electron temperature

density IN REQUIRED TYPE=float

electron density

wavelength IN REQUIRED TYPE=float

Line Wavelength in Angstrom

line_flux IN REQUIRED TYPE=float

line flux intensity

Examples

For example:

```
IDL> ne2=obj_new('recombination')
 IDL> ne2->set,['ne','iii'] ; Ne II
 IDL>
 IDL> temperature=double(10000.0)
 IDL> density=double(5000.0)
 IDL>
 IDL> ne_ii_3777_flux = 0.056
 IDL> wavelength=3777.14
 IDL> Abund_ne_ii=ne2->calc_abundance(temperature=temperature, density=density, $
 IDL>                                wavelength=wavelength, line_flux=ne_ii_3777_flux)
 IDL> print, 'N(Ne^2+)/N(H+):', Abund_ne_ii
    N(Ne^2+)/N(H+):   0.00043376850
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on effective radiative recombination coefficients for Ne II lines from Kisielius et al. 1998A&AS..133..257K & Storey (unpublished).

Adopted from MOCASSIN, Ercolano et al. 2005MNRAS.362.1038E.

02/2003, Yong Zhang, scripts added to MOCASSIN.

14/05/2013, A. Danehkar, Translated to IDL code.

10/04/2017, A. Danehkar, Integration with AtomNeb.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.3.0

RECOMBINATION::SET_DATA_RC_DIR

```
recombination::set_data_rc_dir, data_rc_dir
```

Parameters

data_rc_dir

RECOMBINATION::GET_DATA_RC_DIR

```
result = recombination::get_data_rc_dir()
```

RECOMBINATION::SET_ATOM_RC_ALL_FILE

```
recombination::set_Atom_RC_All_file, Atom_RC_All_file
```

Parameters

Atom_RC_All_file

RECOMBINATION::ATOM_RC_ALL_FILE

```
result = recombination::Atom_RC_All_file()
```

RECOMBINATION::SET_ATOM_RC_HE_I_FILE

```
recombination::set_Atom_RC_He_I_file, Atom_RC_He_I_file
```

Parameters

Atom_RC_He_I_file

RECOMBINATION::GET_ATOM_RC_HE_I_FILE

```
result = recombination::get_Atom_RC_He_I_file()
```

RECOMBINATION::SET_ATOM_RC_PPB91_FILE

```
recombination::set_Atom_RC_PPB91_file, Atom_RC_PPB91_file
```

Parameters

Atom_RC_PPB91_file

RECOMBINATION::GET_ATOM_RC_PPB91_FILE

```
result = recombination::get_Atom_RC_PPB91_file()
```

RECOMBINATION::SET_ATOM_RC_SH95_FILE

```
recombination::set_Atom_RC_SH95_file, Atom_RC_SH95_file
```

Parameters

Atom_RC_SH95_file

RECOMBINATION::GET_ATOM_RC_SH95_FILE

```
result = recombination::get_Atom_RC_SH95_file()
```

RECOMBINATION::SET_ATOM_RC_N_II_FSL13_FILE

```
recombination::set_Atom_RC_N_II_FSL13_file, Atom_RC_N_II_FSL13_file
```

Parameters

Atom_RC_N_II_FSL13_file

RECOMBINATION::GET_ATOM_RC_N_II_FSL13_FILE

```
result = recombination::get_Atom_RC_N_II_FSL13_file()
```

RECOMBINATION::SET_ATOM_RC_O_II_SSB17_FILE

```
recombination::set_Atom_RC_O_II_SSB17_file, Atom_RC_O_II_SSB17_file
```

Parameters

Atom_RC_O_II_SSB17_file

RECOMBINATION::GET_ATOM_RC_O_II_SSB17_FILE

```
result = recombination::get_Atom_RC_O_II_SSB17_file()
```

RECOMBINATION__DEFINE

```
recombination__define
```

reddening__define.pro

Class description for reddening

Fields

BASE_DIR ''

REDDENING::INIT

"Unit for Reddening and Dereddening": This obejct library can be used to determine the reddening law function of the line at the given wavelength for the used extinction law.

```
result = reddening::init()
```

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_gal(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.32013816

 IDL> fl=ext->redlaw_gal2(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.30925984

 IDL> fl=ext->redlaw_ccm(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.29756615

 IDL> fl=ext->redlaw_jbk(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.33113684
```

```
IDL> fmlaw='AVGLMC'
IDL> fl=ext->redlaw_fm(wavelength, fmlaw=fmlaw, rv=R_V)
IDL> print, 'fl(6563)', fl
   fl(6563)     -0.35053032

IDL> fl=ext->redlaw_smc(wavelength)
IDL> print, 'fl(6563)', fl
   fl(6563)     -0.22659261

IDL> fl=ext->redlaw_lmc(wavelength)
IDL> print, 'fl(6563)', fl
   fl(6563)     -0.30871187

IDL> fl=ext->redlaw(wavelength, rv=R_V)
IDL> print, 'fl(6563)', fl
   fl(6563)     -0.32013816

IDL> ext_law='GAL'
IDL> R_V=3.1
IDL> flux_deredden=ext->deredden_relflux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V)
IDL> print, 'dereddened flux(6563)', flux_deredden
   dereddened flux(6563)      0.47847785

IDL> flux_deredden=ext->deredden_flux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V)
IDL> print, 'dereddened flux(6563)', flux_deredden
   dereddened flux(6563)       4.7847785
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.3.0

REDDENING::REDLAW

This function determines the reddening law function of the line at the given wavelength for the used extinction law.

```
  result = reddening::redlaw(wavelength [, ext_law=string] [, rv=float] [, fmlaw=string])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array

Wavelength in Angstrom

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL

the extinction law:

'GAL' for Howarth Galactic.

'GAL2' for Savage and Mathis.

'CCM' for CCM galactic.

'JBK' for Whitford, Seaton, Kaler.

'FM' for Fitxpatrick.

'SMC' for Prevot SMC.

'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.32013816
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_GAL

This function determines the reddening law function of the line at the given wavelength for Galactic Seaton1979+Howarth1983+CCM1983.

```
result = reddening::redlaw_gal(wavelength [, rv=float])
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float

Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_gal(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.32013816
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on the UV Formulae from Seaton 1979, MNRAS, 187, 73 1979MNRAS.187P..73S, the opt/NIR from Howarth 1983, MNRAS, 203, 301 the FIR from Cardelli, Clayton and Mathis 1989, ApJ, 345, 245 1989ApJ...345..245C

Originally from IRAF STSDAS SYNPHOT ebmvxfunc.x, pyneb.extinction

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_GAL2

This function determines the reddening law function of the line at the given wavelength for Galactic Savage & Mathis 1979.

```
result = reddening::redlaw_gal2(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_gal2(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.30925984
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Savage & Mathis 1979, ARA&A, vol. 17, 73-111

Originally from IRAF STSDAS SYNPHOT ebmvxfunc.x

20/09/1994, R. A. Shaw, Initial IRAF implementation.

04/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_CCM

This function determines the reddening law function of Cardelli, Clayton & Mathis.

```
  result = reddening::redlaw_ccm(wavelength [, rv=float])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array
Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1
the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_ccm(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.29756615
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Cardelli, Clayton & Mathis 1989, ApJ 345, 245-256. 1989ApJ...345..245C

Originally from IRAF STSDAS SYNPHOT redlaw.x

18/05/1993, R. A. Shaw, Initial IRAF implementation, based upon CCM module in onedspec.deredden.

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_JBK

This function determines the reddening law function for Galactic Whitford1958 + Seaton1977 + Kaler1976.

```
result = reddening::redlaw_jbk(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_jbk(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.33113684
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Whitford (1958), extended to the UV by Seaton (1977), adapted by Kaler (1976).

Originally from IRAF STSDAS SYNPHOT redlaw.x

13/05/1993, R. A. Shaw, Initial IRAF implementation.

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_FM

This function determines the reddening law function by Fitzpatrick & Massa for the line at the given wavelength.

```
result = reddening::redlaw_fm(wavelength [, rv=float] [, fmlaw=string])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array

Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fmlaw='AVGLMC'
 IDL> fl=ext->redlaw_fm(wavelength, fmlaw=fmlaw, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.35053032
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Fitzpatrick 1999, PASP, 11, 63 1999PASP..111...63F, Fitzpatrick & Massa 1990, ApJS, 72, 163, 1990ApJS...72..163F

Adopted from NASA IDL Library & PyAstronomy.

30/12/2016, A. Danehkar, Revised in IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_SMC

This function determines the reddening law function of the line at the given wavelength for Small Magellanic Cloud.

```
  result = reddening::redlaw_smc(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float

Wavelength in Angstrom

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_smc(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.22659261
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Prevot et al. (1984), A&A, 132, 389-392 1984A%26A...132..389P

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

20/09/1994, R. A. Shaw, Initial IRAF implementation.

04/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::REDLAW_LMC

This function determines the reddening law function of the line at the given wavelength for the Large Magellanic Cloud.

```
  result = reddening::redlaw_lmc(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> fl=ext->redlaw_lmc(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.30871187
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Howarth 1983, MNRAS, 203, 301 1983MNRAS.203..301H

Originally from IRAF STSDAS SYNPHOT ebmvlfunc.x, redlaw.x

18/10/1994, R. A. Shaw, Initial IRAF implementation.

14/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::DEREDDEN_FLUX

This function dereddens absolute flux intensity based on the reddening law.

```
result = reddening::deredden_flux(wavelength, flux, m_ext [, ext_law=string] [, rv=float]
    [, fmlaw=string])
```

Returns

type=double. This function returns the deredden flux intensity.

Parameters

wavelength IN REQUIRED TYPE=float/array
Wavelength in Angstrom

flux IN REQUIRED TYPE=float
absolute flux intensity

m_ext IN REQUIRED TYPE=float
logarithmic extinction

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL
the extinction law:
'GAL' for Howarth Galactic.
'GAL2' for Savage and Mathis.
'CCM' for CCM galactic.
'JBK' for Whitford, Seaton, Kaler.
'FM' for Fitxpatrick.
'SMC' for Prevot SMC.
'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> flux_deredden=ext->deredden_flux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V)
 IDL> print, 'dereddened flux(6563)', flux_deredden
    dereddened flux(6563)       4.7847785
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

31/08/2012, A. Danehkar, IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING::DEREDDEN_RELFLUX

This function dereddens flux intensity relative to Hb=100, based on the reddening law.

```
result = reddening::deredden_relflux(wavelength, relflux, m_ext [, ext_law=string] [, rv=
   float] [, fmlaw=string])
```

Returns

type=double. This function returns the deredden flux intensity relative to Hb=100.

Parameters

wavelength IN REQUIRED TYPE=float/array

Wavelength in Angstrom

relflux IN REQUIRED TYPE=float

flux intensity relative to Hb=100

m_ext IN REQUIRED TYPE=float

logarithmic extinction

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL

the extinction law:

'GAL' for Howarth Galactic.

'GAL2' for Savage and Mathis.

'CCM' for CCM galactic.

'JBK' for Whitford, Seaton, Kaler.

'FM' for Fitxpatrick.

'SMC' for Prevot SMC.

'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL

the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> ext=obj_new('reddening')
 IDL> wavelength=6563.0
 IDL> m_ext=1.0
 IDL> flux=1.0
 IDL> R_V=3.1
 IDL>
 IDL> ext_law='GAL'
 IDL> R_V=3.1
 IDL> flux_deredden=ext->deredden_relflux(wavelength, flux, m_ext, ext_law=ext_law, rv=R_V)
 IDL> print, 'dereddened flux(6563)', flux_deredden
    dereddened flux(6563)      0.47847785
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

31/08/2012, A. Danehkar, IDL code.

08/07/2019, A. Danehkar, Move to object-oriented programming (OOP).

Version

0.2.0

REDDENING__DEFINE

```
reddening__define
```

redlaw.pro

REDLAW

This function determines the reddening law function of the line at the given wavelength for the used extinction law.

```
result = redlaw(wavelength [, ext_law=string] [, rv=float] [, fmlaw=string])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array
Wavelength in Angstrom

Keywords

ext_law IN OPTIONAL TYPE=string DEFAULT=GAL
the extinction law:
'GAL' for Howarth Galactic.
'GAL2' for Savage and Mathis.
'CCM' for CCM galactic.
'JBK' for Whitford, Seaton, Kaler.
'FM' for Fitxpatrick.
'SMC' for Prevot SMC.
'LMC' for Howarth LMC.

rv IN OPTIONAL TYPE=float DEFAULT=3.1
the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL
the fmlaw keyword is used only in the redlaw_fm function:
'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).
'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).
'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> R_V=3.1
 IDL> fl=redlaw(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.32013816
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_ccm.pro

REDLAW_CCM

This function determines the reddening law function of Cardelli, Clayton & Mathis.

```
result = redlaw_ccm(wavelength [, rv=float])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array

Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1
the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> R_V=3.1
 IDL> fl=redlaw_ccm(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.29756615
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Cardelli, Clayton & Mathis 1989, ApJ 345, 245-256. 1989ApJ...345..245C

Originally from IRAF STSDAS SYNPHOT redlaw.x

18/05/1993, R. A. Shaw, Initial IRAF implementation, based upon CCM module in onedspec.deredden.

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_fm.pro

REDLAW_FM

This function determines the reddening law function by Fitzpatrick & Massa for the line at the given wavelength.

```
result = redlaw_fm(wavelength [, rv=float] [, fmlaw=string])
```

Returns

type=double/array. This function returns the reddening law function value for the given wavelength.

Parameters

wavelength IN REQUIRED TYPE=float/array
Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1
the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

fmlaw IN OPTIONAL TYPE=string DEFAULT=GAL
the fmlaw keyword is used only in the redlaw_fm function:

'GAL' for the default fit parameters for the R-dependent Galactic extinction curve from Fitzpatrick & Massa (Fitzpatrick, 1999, PASP, 111, 63).

'LMC2' for the fit parameters are those determined for reddening the LMC2 field (inc. 30 Dor) from Misselt et al. (1999, ApJ, 515, 128).

'AVGLMC' for the fit parameters are those determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999, ApJ, 515, 128).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> R_V=3.1
 IDL> fl=redlaw_fm(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.35054942
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Fitzpatrick 1999, PASP, 11, 63 1999PASP..111...63F, Fitzpatrick & Massa 1990, ApJS, 72, 163, 1990ApJS...72..163F

Adopted from NASA IDL Library & PyAstronomy.

30/12/2016, A. Danehkar, Revised in IDL code.

Version

0.3.0

redlaw_gal.pro

REDLAW_GAL

This function determines the reddening law function of the line at the given wavelength for Galactic Seaton1979+Howarth1983+CCM1983.

```
  result = redlaw_gal(wavelength [, rv=float])
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float

Wavelength in Angstrom

Keywords

rv IN OPTIONAL TYPE=float DEFAULT=3.1

the optical total-to-selective extinction ratio, RV = A(V)/E(B-V).

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> R_V=3.1
 IDL> fl=redlaw_gal(wavelength, rv=R_V)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.32013816
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on the UV Formulae from Seaton 1979, MNRAS, 187, 73 1979MNRAS.187P..73S, the opt/NIR from Howarth 1983, MNRAS, 203, 301 the FIR from Cardelli, Clayton and Mathis 1989, ApJ, 345, 245 1989ApJ...345..245C

Originally from IRAF STSDAS SYNPHOT ebmvxfunc.x, pyneb.extinction

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_gal2.pro

REDLAW_GAL2

This function determines the reddening law function of the line at the given wavelength for Galactic Savage & Mathis 1979.

```
result = redlaw_gal2(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> fl=redlaw_gal2(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.30925984
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Savage & Mathis 1979, ARA&A, vol. 17, 73-111

Originally from IRAF STSDAS SYNPHOT ebmvxfunc.x

20/09/1994, R. A. Shaw, Initial IRAF implementation.

04/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_jbk.pro

REDLAW_JBK

This function determines the reddening law function for Galactic Whitford1958 + Seaton1977 + Kaler1976.

```
result = redlaw_jbk(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float

Wavelength in Angstrom

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> fl=redlaw_jbk(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.33113684
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Whitford (1958), extended to the UV by Seaton (1977), adapted by Kaler (1976).

Originally from IRAF STSDAS SYNPHOT redlaw.x

13/05/1993, R. A. Shaw, Initial IRAF implementation.

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_lmc.pro

REDLAW_LMC

This function determines the reddening law function of the line at the given wavelength for the Large Magellanic Cloud.

```
result = redlaw_lmc(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> fl=redlaw_lmc(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.30871187
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Formulae by Howarth 1983, MNRAS, 203, 301 1983MNRAS.203..301H

Originally from IRAF STSDAS SYNPHOT ebmvlfunc.x, redlaw.x

18/10/1994, R. A. Shaw, Initial IRAF implementation.

14/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0

redlaw_smc.pro

REDLAW_SMC

This function determines the reddening law function of the line at the given wavelength for Small Magellanic Cloud.

```
result = redlaw_smc(wavelength)
```

Returns

type=double/array. This function returns the reddening law function value(s) for the given wavelength(s).

Parameters

wavelength IN REQUIRED TYPE=float
Wavelength in Angstrom

Examples

For example:

```
IDL> wavelength=6563.0
 IDL> fl=redlaw_smc(wavelength)
 IDL> print, 'fl(6563)', fl
    fl(6563)     -0.22659261
```

Author

Ashkbiz Danehkar

Copyright

This library is released under a GNU General Public License.

History

Based on Prevot et al. (1984), A&A, 132, 389-392 1984A%26A...132..389P

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

20/09/1994, R. A. Shaw, Initial IRAF implementation.

04/03/1995, R. A. Shaw, Return A(lambda)/A(V) instead.

31/08/2012, A. Danehkar, Converted to IDL code.

Version

0.3.0