IDL Library for Plasma Diagnostics and Abundance Analysis

API Documentation for proEQUIB

Contents

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:

1. `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.

2. `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.

3. `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

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:	25
.sav files:	0
Routines:	25
Lines:	1,780

Part II

API

Directory: ./

Overview

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.

Version

0.0.3

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.

Version

0.0.3

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

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.

Version

0.0.3

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

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

0.0.3

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.

Version

0.0.3

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.

Version

0.0.3

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.

Version

0.0.3

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 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.

Version

0.0.3

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+):   0.00041257847
```

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

0.0.6

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)
```

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

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:       2602.2294
```

Author

Ashkbiz Danehkar

Copyright

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().

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

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):   3.6039600e-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().

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

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, temp_list=array, Omij=array/
   object, Aij=array/object, Elj=array, Glj=array, level_num=int, temp_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

temp_list IN REQUIRED TYPE=array
temperature intervals (array)

Omij IN REQUIRED TYPE=array/object
Collision Strengths (Omega_ij)

Aij IN REQUIRED TYPE=array/object
Transition Probabilities (A_ij)

Elj IN REQUIRED TYPE=array
Energy Levels (E_j)

Glj IN REQUIRED TYPE=array
Ground Levels (G_j)

level_num IN REQUIRED TYPE=int
Number of levels

temp_num IN REQUIRED TYPE=int
Number of temperature intervals

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

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().

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

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)
```

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

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().

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

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

0.0.1

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

0.0.1

gamma_hb_4861.pro

GAMMA_HB_4861

private

This function determines the value of gamma(HBeta 4861 A) = log10(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 = gamma_hb_4861(temperature=float, density=float, h_i_aeff_data=array/object)
```

Returns

type=double. This function returns the value of gamma(HBeta 4861) = log10(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.

Version

0.0.2

gamma_he_ii_4686.pro

GAMMA_HE_II_4686

private

This function determines the value of gamma(He II 4686 A) = log10(4pi j(He II 4686 A)/Np Ne) for the given temperature and density by using the helium emissivities from Storey & Hummer, 1995MNRAS.272...41S.

```
result = gamma_he_ii_4686(temperature=float, density=float, he_ii_aeff_data=array/object)
```

Returns

type=double. This function returns the value of gamma(He II 4686) = log10(4pi j(He II 4686)/Np Ne).

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

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.

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

02/03/2017, A. Danehkar, Integration with AtomNeb.

Version

0.0.2

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

History

Originally from IRAF STSDAS SYNPHOT redlaw.x, ebmvx-func.x

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

Version

0.0.1

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

0.0.1

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

0.0.1

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

0.0.1

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

0.0.1

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

0.0.1

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

0.0.1

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

0.0.1