IASI Code Collection

Generated by Doxygen 1.8.11

Contents

1 Main Page

The JUelich RApid Spectral SImulation Code (JURASSIC) is a fast radiative transfer model for the mid-infrared spectral region.This reference manual provides information on the algorithms and data structures used in the code. Further information can be found at: http://www.fz-juelich.de/ias/jsc/jurassic

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Data Structure Documentation

4.1 atm_t Struct Reference

Atmospheric data.

```
#include <jurassic.h>
```

Data Fields

- int np
  Number of data points.
- double time [NP]
  Time (seconds since 2000-01-01T00:00Z).
- double z [NP]
  Altitude [km].
- double lon [NP]
  Longitude [deg].
- double lat [NP]

Latitude [deg].

- double p [NP]

  Pressure [hPa].

- double t [NP]

  Temperature [K].

- double q [NG][NP]

  Volume mixing ratio.

- double k [NW][NP]

  Extinction [1/km].

4.1.1 Detailed Description

Atmospheric data.

Definition at line 219 of file jurassic.h.

4.1.2 Field Documentation

4.1.2.1 int atm_t::np

Number of data points.

Definition at line 222 of file jurassic.h.

4.1.2.2 double atm_t::time[NP]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 225 of file jurassic.h.

4.1.2.3 double atm_t::z[NP]

Altitude [km].

Definition at line 228 of file jurassic.h.

4.1.2.4 double atm_t::lon[NP]

Longitude [deg].

Definition at line 231 of file jurassic.h.

4.1.2.5 double atm_t::lat[NP]

Latitude [deg].

Definition at line 234 of file jurassic.h.

4.1.2.6 double atm_t::p[NP]

Pressure [hPa].

Definition at line 237 of file jurassic.h.

4.1.2.7 double atm_t::t[NP]

Temperature [K].

Definition at line 240 of file jurassic.h.

4.1.2.8 double atm_t::q[NG][NP]

Volume mixing ratio.

Definition at line 243 of file jurassic.h.

4.1.2.9 double atm_t::k[NW][NP]

Extinction [1/km].

Definition at line 246 of file jurassic.h.

The documentation for this struct was generated from the following file:

- jurassic.h

4.2 ctl2_t Struct Reference

Control parameters.

Data Fields

- double dt_met
  Time step of meteorological data [s].
- char met_geopot [LEN]
  Surface geopotential data file.
- int met_dx
  Stride for longitudes.
- int met_dy
  Stride for latitudes.
- int met_dp
  Stride for pressure levels.
- int met_sx
  Smoothing for longitudes.
- int met_sy
  Smoothing for latitudes.
- int met_sp
  Smoothing for pressure levels.

4.2.1 Detailed Description

Control parameters.

Definition at line 32 of file extract.c.

4.2.2 Field Documentation

4.2.2.1 double ctl2_t::dt_met

Time step of meteorological data [s].

Definition at line 35 of file extract.c.

4.2.2.2 char ctl2_t::met_geopot[LEN]

Surface geopotential data file.

Definition at line 38 of file extract.c.

4.2.2.3 int ctl2_t::met_dx

Stride for longitudes.

Definition at line 41 of file extract.c.

4.2.2.4 int ctl2_t::met_dy

Stride for latitudes.

Definition at line 44 of file extract.c.

4.2.2.5 int ctl2_t::met_dp

Stride for pressure levels.

Definition at line 47 of file extract.c.

4.2.2.6 int ctl2_t::met_sx

Smoothing for longitudes.

Definition at line 50 of file extract.c.

4.2.2.7 int ctl2_t::met_sy

Smoothing for latitudes.

Definition at line 53 of file extract.c.

4.2.2.8 int ctl2_t::met_sp

Smoothing for pressure levels.

Definition at line 56 of file extract.c.

The documentation for this struct was generated from the following file:

- extract.c

4.3 ctl_t Struct Reference

Forward model control parameters.

`#include <jurassic.h>`

Data Fields

- int ng
  Number of emitters.
- char emitter [NG][LEN]
  Name of each emitter.
- int nd
  Number of radiance channels.
- int nw
  Number of spectral windows.
- double nu [ND]
  Centroid wavenumber of each channel [cm^-1].
- int window [ND]
  Window index of each channel.
- char tblbase [LEN]
  Basename for table files and filter function files.
- double hydz
  Reference height for hydrostatic pressure profile (-999 to skip) [km].
- int ctm_co2
  Compute CO2 continuum (0=no, 1=yes).
- int ctm_h2o
  Compute H2O continuum (0=no, 1=yes).
- int ctm_n2
  Compute N2 continuum (0=no, 1=yes).
- int ctm_o2
  Compute O2 continuum (0=no, 1=yes).
- int refrac
  Take into account refractivity (0=no, 1=yes).
- double rayds
  Maximum step length for raytracing [km].
- double raydz
  Vertical step length for raytracing [km].
- char fov [LEN]
  Field-of-view data file.

- double retp_zmin

  Minimum altitude for pressure retrieval [km].
- double retp_zmax

  Maximum altitude for pressure retrieval [km].
- double rett_zmin

  Minimum altitude for temperature retrieval [km].
- double rett_zmax

  Maximum altitude for temperature retrieval [km].
- double retq_zmin [NG]

  Minimum altitude for volume mixing ratio retrieval [km].
- double retq_zmax [NG]

  Maximum altitude for volume mixing ratio retrieval [km].
- double retk_zmin [NW]

  Minimum altitude for extinction retrieval [km].
- double retk_zmax [NW]

  Maximum altitude for extinction retrieval [km].
- int write_bbt

  Use brightness temperature instead of radiance (0=no, 1=yes).
- int write_matrix

  Write matrix file (0=no, 1=yes).

4.3.1 Detailed Description

Forward model control parameters.

Definition at line 251 of file jurassic.h.

4.3.2 Field Documentation

4.3.2.1 int ctl_t::ng

Number of emitters.

Definition at line 254 of file jurassic.h.

4.3.2.2 char ctl_t::emitter[NG][LEN]

Name of each emitter.

Definition at line 257 of file jurassic.h.

4.3.2.3 int ctl_t::nd

Number of radiance channels.

Definition at line 260 of file jurassic.h.

4.3.2.4 int ctl_t::nw

Number of spectral windows.

Definition at line 263 of file jurassic.h.

4.3.2.5 double ctl_t::nu[ND]

Centroid wavenumber of each channel [cm^-1].

Definition at line 266 of file jurassic.h.

4.3.2.6 int ctl_t::window[ND]

Window index of each channel.

Definition at line 269 of file jurassic.h.

4.3.2.7 char ctl_t::tblbase[LEN]

Basename for table files and filter function files.

Definition at line 272 of file jurassic.h.

4.3.2.8 double ctl_t::hydz

Reference height for hydrostatic pressure profile (-999 to skip) [km].

Definition at line 275 of file jurassic.h.

4.3.2.9 int ctl_t::ctm_co2

Compute CO2 continuum (0=no, 1=yes).

Definition at line 278 of file jurassic.h.

4.3.2.10 int ctl_t::ctm_h2o

Compute H2O continuum (0=no, 1=yes).

Definition at line 281 of file jurassic.h.

4.3.2.11 int ctl_t::ctm_n2

Compute N2 continuum (0=no, 1=yes).

Definition at line 284 of file jurassic.h.

4.3.2.12 int ctl_t::ctm_o2

Compute O2 continuum (0=no, 1=yes).

Definition at line 287 of file jurassic.h.

4.3.2.13 int ctl_t::refrac

Take into account refractivity (0=no, 1=yes).

Definition at line 290 of file jurassic.h.

4.3.2.14 double ctl_t::rayds

Maximum step length for raytracing [km].

Definition at line 293 of file jurassic.h.

4.3.2.15 double ctl_t::raydz

Vertical step length for raytracing [km].

Definition at line 296 of file jurassic.h.

4.3.2.16 char ctl_t::fov[LEN]

Field-of-view data file.

Definition at line 299 of file jurassic.h.

4.3.2.17 double ctl_t::retp_zmin

Minimum altitude for pressure retrieval [km].

Definition at line 302 of file jurassic.h.

4.3.2.18 double ctl_t::retp_zmax

Maximum altitude for pressure retrieval [km].

Definition at line 305 of file jurassic.h.

4.3.2.19 double ctl_t::rett_zmin

Minimum altitude for temperature retrieval [km].

Definition at line 308 of file jurassic.h.

4.3.2.20 double ctl_t::rett_zmax

Maximum altitude for temperature retrieval [km].

Definition at line 311 of file jurassic.h.

4.3.2.21 double ctl_t::retq_zmin[NG]

Minimum altitude for volume mixing ratio retrieval [km].

Definition at line 314 of file jurassic.h.

4.3.2.22 double ctl_t::retq_zmax[NG]

Maximum altitude for volume mixing ratio retrieval [km].

Definition at line 317 of file jurassic.h.

4.3.2.23 double ctl_t::retk_zmin[NW]

Minimum altitude for extinction retrieval [km].

Definition at line 320 of file jurassic.h.

4.3.2.24 double ctl_t::retk_zmax[NW]

Maximum altitude for extinction retrieval [km].

Definition at line 323 of file jurassic.h.

4.3.2.25 int ctl_t::write_bbt

Use brightness temperature instead of radiance (0=no, 1=yes).

Definition at line 326 of file jurassic.h.

4.3.2.26 int ctl_t::write_matrix

Write matrix file (0=no, 1=yes).

Definition at line 329 of file jurassic.h.

The documentation for this struct was generated from the following file:

- jurassic.h

4.4 iasi_l1_t Struct Reference

IASI Level-1 data.

```
#include <libiasi.h>
```

Data Fields

- size_t ntrack
  Number of along-track values.
- double time [L1_NTRACK][L1_NXTRACK]
  Time (seconds since 2000-01-01T00:00Z).
- double lon [L1_NTRACK][L1_NXTRACK]
  Footprint longitude [deg].
- double lat [L1_NTRACK][L1_NXTRACK]
  Footprint latitude [deg].
- double sat_z [L1_NTRACK]
  Satellite altitude [km].
- double sat_lon [L1_NTRACK]
  Satellite longitude [deg].
- double sat_lat [L1_NTRACK]
  Satellite latitude [deg].
- double nu [L1_NCHAN]
  Channel frequencies [cm^-1].
- float rad [L1_NTRACK][L1_NXTRACK][L1_NCHAN]
  Radiance [W/(m^2 sr cm^-1)].

4.4.1 Detailed Description

IASI Level-1 data.

Definition at line 84 of file libiasi.h.

4.4.2 Field Documentation

4.4.2.1 size_t iasi_l1_t::ntrack

Number of along-track values.

Definition at line 87 of file libiasi.h.

4.4.2.2 double iasi_l1_t::time[L1_NTRACK][L1_NXTRACK]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 90 of file libiasi.h.

4.4.2.3 double iasi_l1_t::lon[L1_NTRACK][L1_NXTRACK]

Footprint longitude [deg].

Definition at line 93 of file libiasi.h.

4.4.2.4 double iasi_l1_t::lat[L1_NTRACK][L1_NXTRACK]

Footprint latitude [deg].

Definition at line 96 of file libiasi.h.

4.4.2.5 double iasi_l1_t::sat_z[L1_NTRACK]

Satellite altitude [km].

Definition at line 99 of file libiasi.h.

4.4.2.6 double iasi_l1_t::sat_lon[L1_NTRACK]

Satellite longitude [deg].

Definition at line 102 of file libiasi.h.

4.4.2.7 double iasi_l1_t::sat_lat[L1_NTRACK]

Satellite latitude [deg].

Definition at line 105 of file libiasi.h.

4.4.2.8 double iasi_l1_t::nu[L1_NCHAN]

Channel frequencies [cm^-1].

Definition at line 108 of file libiasi.h.

4.4.2.9 float iasi_l1_t::rad[L1_NTRACK][L1_NXTRACK][L1_NCHAN]

Radiance [W/(m^2 sr cm^-1)].

Definition at line 111 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

4.5 iasi_l2_t Struct Reference

IASI Level-2 data.

```
#include <libiasi.h>
```

Data Fields

- size_t ntrack
  Number of along-track values.
- double time [L2_NTRACK][L2_NXTRACK]
  Time (seconds since 2000-01-01T00:00Z).
- double z [L2_NTRACK][L2_NXTRACK][L2_NLAY]
  Geopotential height [km].
- double lon [L2_NTRACK][L2_NXTRACK]
  Longitude [deg].
- double lat [L2_NTRACK][L2_NXTRACK]
  Latitude [deg].
- double p [L2_NLAY]
  Pressure [hPa].
- double t [L2_NTRACK][L2_NXTRACK][L2_NLAY]
  Temperature [K].

4.5.1 Detailed Description

IASI Level-2 data.

Definition at line 116 of file libiasi.h.

4.5.2 Field Documentation

4.5.2.1 size_t iasi_l2_t::ntrack

Number of along-track values.

Definition at line 119 of file libiasi.h.

4.5.2.2 double iasi_l2_t::time[L2_NTRACK][L2_NXTRACK]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 122 of file libiasi.h.

4.5.2.3 double iasi_l2_t::z[L2_NTRACK][L2_NXTRACK][L2_NLAY]

Geopotential height [km].

Definition at line 125 of file libiasi.h.

4.5.2.4 double iasi_l2_t::lon[L2_NTRACK][L2_NXTRACK]

Longitude [deg].

Definition at line 128 of file libiasi.h.

4.5.2.5 double iasi_l2_t::lat[L2_NTRACK][L2_NXTRACK]

Latitude [deg].

Definition at line 131 of file libiasi.h.

4.5.2.6 double iasi_l2_t::p[L2_NLAY]

Pressure [hPa].

Definition at line 134 of file libiasi.h.

4.5.2.7 double iasi_l2_t::t[L2_NTRACK][L2_NXTRACK][L2_NLAY]

Temperature [K].

Definition at line 137 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

4.6 iasi_rad_t Struct Reference

IASI converted Level-1 radiation data.

```
#include <libiasi.h>
```

Data Fields

- int ntrack

  Number of along-track samples.
- double freq [IASI_L1_NCHAN]

  channel wavenumber [cm^-1]
- double Time [L1_NTRACK][L1_NXTRACK]

  Seconds since 2000-01-01 for each sounder pixel.
- double Longitude [L1_NTRACK][L1_NXTRACK]

  Longitude of the sounder pixel.
- double Latitude [L1_NTRACK][L1_NXTRACK]

  Latitude of the sounder pixel.
- float Rad [L1_NTRACK][L1_NXTRACK][IASI_L1_NCHAN]

  Radiance [W/(m^2 sr cm^-1)].
- double Sat_z [L1_NTRACK]

  Altitude of the satellite.
- double Sat_lon [L1_NTRACK]

  Estimated longitude of the satellite.
- double Sat_lat [L1_NTRACK]

  Estimated latitude of the satellite.

4.6.1 Detailed Description

IASI converted Level-1 radiation data.

Definition at line 206 of file libiasi.h.

4.6.2 Field Documentation

4.6.2.1 int iasi_rad_t::ntrack

Number of along-track samples.

Definition at line 209 of file libiasi.h.

4.6.2.2 double iasi_rad_t::freq[IASI_L1_NCHAN]

channel wavenumber [cm^-1]

Definition at line 212 of file libiasi.h.

4.6.2.3 double iasi_rad_t::Time[L1_NTRACK][L1_NXTRACK]

Seconds since 2000-01-01 for each sounder pixel.

Definition at line 215 of file libiasi.h.

4.6.2.4 double iasi_rad_t::Longitude[L1_NTRACK][L1_NXTRACK]

Longitude of the sounder pixel.

Definition at line 218 of file libiasi.h.

4.6.2.5 double iasi_rad_t::Latitude[L1_NTRACK][L1_NXTRACK]

Latitude of the sounder pixel.

Definition at line 221 of file libiasi.h.

4.6.2.6 float iasi_rad_t::Rad[L1_NTRACK][L1_NXTRACK][IASI_L1_NCHAN]

Radiance [W/(m^2 sr cm^-1)].

Definition at line 224 of file libiasi.h.

4.6.2.7 double iasi_rad_t::Sat_z[L1_NTRACK]

Altitude of the satellite.

Definition at line 227 of file libiasi.h.

4.6.2.8 double iasi_rad_t::Sat_lon[L1_NTRACK]

Estimated longitude of the satellite.

Definition at line 230 of file libiasi.h.

4.6.2.9 double iasi_rad_t::Sat_lat[L1_NTRACK]

Estimated latitude of the satellite.

Definition at line 233 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

4.7 iasi_raw_t Struct Reference

IASI raw Level-1 data.

```
#include <libiasi.h>
```

Data Fields

- long ntrack

  Number of along-track samples.
- float IDefSpectDWn1b [L1_NTRACK]

  Constants for radiation spectrum (must be equal to the expected constant).
- int32_t IDefNsfirst1b [L1_NTRACK]

  Constants for radiation spectrum (must be equal to the expected constant).
- int32_t IDefNslast1b [L1_NTRACK]

  Constants for radiation spectrum (must be equal to the expected constant).
- double Time [L1_NTRACK][IASI_NXTRACK]

  Time (seconds since 2000-01-01T00:00Z).
- double Loc [L1_NTRACK][IASI_NXTRACK][IASI_PM][2]

  Location of the sounder pixel (long,lat).
- float Wavenumber [IASI_L1_NCHAN]

  Wavenumbers are computed with the expected values.
- short int Radiation [L1_NTRACK][IASI_NXTRACK][IASI_PM][IASI_L1_NCHAN]

  Radiance [W/(m^2 sr m^-1)].
- unsigned int Sat_z [L1_NTRACK]

  Satellite altitude [m].

4.7.1 Detailed Description

IASI raw Level-1 data.

Definition at line 174 of file libiasi.h.

4.7.2 Field Documentation

4.7.2.1 long iasi_raw_t::ntrack

Number of along-track samples.

Definition at line 177 of file libiasi.h.

4.7.2.2 float iasi_raw_t::IDefSpectDWn1b[L1_NTRACK]

Constants for radiation spectrum (must be equal to the expected constant).

Definition at line 180 of file libiasi.h.

4.7.2.3 int32_t iasi_raw_t::IDefNsfirst1b[L1_NTRACK]

Constants for radiation spectrum (must be equal to the expected constant).

Definition at line 183 of file libiasi.h.

4.7.2.4 int32_t iasi_raw_t::lDefNslast1b[L1_NTRACK]

Constants for radiation spectrum (must be equal to the expected constant).

Definition at line 186 of file libiasi.h.

4.7.2.5 double iasi_raw_t::Time[L1_NTRACK][IASI_NXTRACK]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 189 of file libiasi.h.

4.7.2.6 double iasi_raw_t::Loc[L1_NTRACK][IASI_NXTRACK][IASI_PM][2]

Location of the sounder pixel (long,lat).

Definition at line 192 of file libiasi.h.

4.7.2.7 float iasi_raw_t::Wavenumber[IASI_L1_NCHAN]

Wavenumbers are computed with the expected values.

Definition at line 195 of file libiasi.h.

4.7.2.8 short int iasi_raw_t::Radiation[L1_NTRACK][IASI_NXTRACK][IASI_PM][IASI_L1_NCHAN]

Radiance [W/(m^2 sr m^-1)].

Definition at line 198 of file libiasi.h.

4.7.2.9 unsigned int iasi_raw_t::Sat_z[L1_NTRACK]

Satellite altitude [m].

Definition at line 201 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

4.8 los_t Struct Reference

Line-of-sight data.

```
#include <jurassic.h>
```

Data Fields

- int np
  Number of LOS points.
- double z [NLOS]
  Altitude [km].
- double lon [NLOS]
  Longitude [deg].
- double lat [NLOS]
  Latitude [deg].
- double p [NLOS]
  Pressure [hPa].
- double t [NLOS]
  Temperature [K].
- double q [NG][NLOS]
  Volume mixing ratio.
- double k [NW][NLOS]
  Extinction [1/km].
- double tsurf
  Surface temperature [K].
- double ds [NLOS]
  Segment length [km].
- double u [NG][NLOS]
  Column density [molecules/cm^2].

4.8.1 Detailed Description

Line-of-sight data.

Definition at line 334 of file jurassic.h.

4.8.2 Field Documentation

4.8.2.1 int los_t::np

Number of LOS points.

Definition at line 337 of file jurassic.h.

4.8.2.2 double los_t::z[NLOS]

Altitude [km].

Definition at line 340 of file jurassic.h.

4.8.2.3 double los_t::lon[NLOS]

Longitude [deg].

Definition at line 343 of file jurassic.h.

4.8.2.4 double los_t::lat[NLOS]

Latitude [deg].

Definition at line 346 of file jurassic.h.

4.8.2.5 double los_t::p[NLOS]

Pressure [hPa].

Definition at line 349 of file jurassic.h.

4.8.2.6 double los_t::t[NLOS]

Temperature [K].

Definition at line 352 of file jurassic.h.

4.8.2.7 double los_t::q[NG][NLOS]

Volume mixing ratio.

Definition at line 355 of file jurassic.h.

4.8.2.8 double los_t::k[NW][NLOS]

Extinction [1/km].

Definition at line 358 of file jurassic.h.

4.8.2.9 double los_t::tsurf

Surface temperature [K].

Definition at line 361 of file jurassic.h.

4.8.2.10 double los_t::ds[NLOS]

Segment length [km].

Definition at line 364 of file jurassic.h.

4.8.2.11 double los_t::u[NG][NLOS]

Column density [molecules/cm^2].

Definition at line 367 of file jurassic.h.

The documentation for this struct was generated from the following file:

- jurassic.h

4.9 met_t Struct Reference

Meteorological data.

Data Fields

- double time
  Time [s].
- int nx
  Number of longitudes.
- int ny
  Number of latitudes.
- int np
  Number of pressure levels.
- double lon [EX]
  Longitude [deg].
- double lat [EY]
  Latitude [deg].
- double p [EP]
  Pressure [hPa].
- double ps [EX][EY]
  Surface pressure [hPa].
- double pt [EX][EY]
  Tropopause pressure [hPa].
- float z [EX][EY][EP]
  Geopotential height [km].
- float t [EX][EY][EP]
  Temperature [K].
- float u [EX][EY][EP]
  Zonal wind [m/s].
- float v [EX][EY][EP]
  Meridional wind [m/s].
- float w [EX][EY][EP]
  Vertical wind [hPa/s].
- float pv [EX][EY][EP]
  Potential vorticity [PVU].
- float h2o [EX][EY][EP]
  Water vapor volume mixing ratio [1].
- float o3 [EX][EY][EP]
  Ozone volume mixing ratio [1].
- float pl [EX][EY][EP]
  Pressure on model levels [hPa].

4.9.1 Detailed Description

Meteorological data.

Definition at line 61 of file extract.c.

4.9.2 Field Documentation

4.9.2.1 double met_t::time

Time [s].

Definition at line 64 of file extract.c.

4.9.2.2 int met_t::nx

Number of longitudes.

Definition at line 67 of file extract.c.

4.9.2.3 int met_t::ny

Number of latitudes.

Definition at line 70 of file extract.c.

4.9.2.4 int met_t::np

Number of pressure levels.

Definition at line 73 of file extract.c.

4.9.2.5 double met_t::lon[EX]

Longitude [deg].

Definition at line 76 of file extract.c.

4.9.2.6 double met_t::lat[EY]

Latitude [deg].

Definition at line 79 of file extract.c.

4.9.2.7 double met_t::p[EP]

Pressure [hPa].

Definition at line 82 of file extract.c.

4.9.2.8 double met_t::ps[EX][EY]

Surface pressure [hPa].

Definition at line 85 of file extract.c.

4.9.2.9 double met_t::pt[EX][EY]

Tropopause pressure [hPa].

Definition at line 88 of file extract.c.

4.9.2.10 float met_t::z[EX][EY][EP]

Geopotential height [km].

Definition at line 91 of file extract.c.

4.9.2.11 float met_t::t[EX][EY][EP]

Temperature [K].

Definition at line 94 of file extract.c.

4.9.2.12 float met_t::u[EX][EY][EP]

Zonal wind [m/s].

Definition at line 97 of file extract.c.

4.9.2.13 float met_t::v[EX][EY][EP]

Meridional wind [m/s].

Definition at line 100 of file extract.c.

4.9.2.14 float met_t::w[EX][EY][EP]

Vertical wind [hPa/s].

Definition at line 103 of file extract.c.

4.9.2.15 float met_t::pv[EX][EY][EP]

Potential vorticity [PVU].

Definition at line 106 of file extract.c.

4.9.2.16 float met_t::h2o[EX][EY][EP]

Water vapor volume mixing ratio [1].

Definition at line 109 of file extract.c.

4.9.2.17 float met_t::o3[EX][EY][EP]

Ozone volume mixing ratio [1].

Definition at line 112 of file extract.c.

4.9.2.18 float met_t::pl[EX][EY][EP]

Pressure on model levels [hPa].

Definition at line 115 of file extract.c.

The documentation for this struct was generated from the following file:

- extract.c

4.10 ncd_t Struct Reference

Buffer for netCDF data.

Data Fields

- int ncid
  NetCDF file ID.
- int np
  Number of retrieval altitudes.
- int ntrack
  Number of tacks.
- double l1_time [L1_NTRACK][L1_NXTRACK]
  Time (seconds since 2000-01-01T00:00Z).
- double l1_lon [L1_NTRACK][L1_NXTRACK]
  Footprint longitude [deg].
- double l1_lat [L1_NTRACK][L1_NXTRACK]
  Footprint latitude [deg].
- double l1_sat_z [L1_NTRACK]
  Satellite altitude [km].
- double l1_sat_lon [L1_NTRACK]
  Satellite longitude [deg].
- double l1_sat_lat [L1_NTRACK]
  Satellite latitude [deg].
- double l1_nu [L1_NCHAN]
  Channel frequencies [cm^-1].
- float l1_rad [L1_NTRACK][L1_NXTRACK][L1_NCHAN]
  Radiance [W/(m^2 sr cm^-1)].
- double l2_z [L2_NTRACK][L2_NXTRACK][L2_NLAY]
  Altitude [km].
- double l2_p [L2_NLAY]
  Pressure [hPa].
- double l2_t [L2_NTRACK][L2_NXTRACK][L2_NLAY]
  Temperature [K].
- float ret_z [NP]
  Altitude [km].
- float ret_p [L1_NTRACK ∗L1_NXTRACK]
  Pressure [hPa].
- float ret_t [L1_NTRACK ∗L1_NXTRACK ∗NP]
  Temperature [K].
- float ret_chisq [L1_NTRACK ∗L1_NXTRACK]
  chi^2 value of fit.

4.10.1 Detailed Description

Buffer for netCDF data.

Definition at line 43 of file retrieval.c.

4.10.2 Field Documentation

4.10.2.1 int ncd_t::ncid

NetCDF file ID.

Definition at line 46 of file retrieval.c.

4.10.2.2 int ncd_t::np

Number of retrieval altitudes.

Definition at line 49 of file retrieval.c.

4.10.2.3 int ncd_t::ntrack

Number of tacks.

Definition at line 52 of file retrieval.c.

4.10.2.4 double ncd_t::l1_time[L1_NTRACK][L1_NXTRACK]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 55 of file retrieval.c.

4.10.2.5 double ncd_t::l1_lon[L1_NTRACK][L1_NXTRACK]

Footprint longitude [deg].

Definition at line 58 of file retrieval.c.

4.10.2.6 double ncd_t::l1_lat[L1_NTRACK][L1_NXTRACK]

Footprint latitude [deg].

Definition at line 61 of file retrieval.c.

4.10.2.7 double ncd_t::l1_sat_z[L1_NTRACK]

Satellite altitude [km].

Definition at line 64 of file retrieval.c.

4.10.2.8 double ncd_t::l1_sat_lon[L1_NTRACK]

Satellite longitude [deg].

Definition at line 67 of file retrieval.c.

4.10.2.9 double ncd_t::l1_sat_lat[L1_NTRACK]

Satellite latitude [deg].

Definition at line 70 of file retrieval.c.

4.10.2.10 double ncd_t::l1_nu[L1_NCHAN]

Channel frequencies [cm^-1].

Definition at line 73 of file retrieval.c.

4.10.2.11 float ncd_t::l1_rad[L1_NTRACK][L1_NXTRACK][L1_NCHAN]

Radiance [W/(m^2 sr cm^-1)].

Definition at line 76 of file retrieval.c.

4.10.2.12 double ncd_t::l2_z[L2_NTRACK][L2_NXTRACK][L2_NLAY]

Altitude [km].

Definition at line 79 of file retrieval.c.

4.10.2.13 double ncd_t::l2_p[L2_NLAY]

Pressure [hPa].

Definition at line 82 of file retrieval.c.

4.10.2.14 double ncd_t::l2_t[L2_NTRACK][L2_NXTRACK][L2_NLAY]

Temperature [K].

Definition at line 85 of file retrieval.c.

4.10.2.15 float ncd_t::ret_z[NP]

Altitude [km].

Definition at line 88 of file retrieval.c.

4.10.2.16 float ncd_t::ret_p[L1_NTRACK ∗L1_NXTRACK]

Pressure [hPa].

Definition at line 91 of file retrieval.c.

4.10.2.17 float ncd_t::ret_t[L1_NTRACK ∗L1_NXTRACK ∗NP]

Temperature [K].

Definition at line 94 of file retrieval.c.

4.10.2.18 float ncd_t::ret_chisq[L1_NTRACK ∗L1_NXTRACK]

chi^2 value of fit.

Definition at line 97 of file retrieval.c.

The documentation for this struct was generated from the following file:

- retrieval.c

4.11 obs_t Struct Reference

Observation geometry and radiance data.

```
#include <jurassic.h>
```

Data Fields

- int nr
  Number of ray paths.
- double time [NR]
  Time (seconds since 2000-01-01T00:00Z).
- double obsz [NR]
  Observer altitude [km].
- double obslon [NR]
  Observer longitude [deg].
- double obslat [NR]
  Observer latitude [deg].
- double vpz [NR]
  View point altitude [km].
- double vplon [NR]
  View point longitude [deg].
- double vplat [NR]
  View point latitude [deg].
- double tpz [NR]
  Tangent point altitude [km].
- double tplon [NR]
  Tangent point longitude [deg].
- double tplat [NR]
  Tangent point latitude [deg].
- double tau [ND][NR]
  Transmittance of ray path.
- double rad [ND][NR]
  Radiance [W/(m^2 sr cm^-1)].

4.11.1 Detailed Description

Observation geometry and radiance data.

Definition at line 372 of file jurassic.h.

4.11.2 Field Documentation

4.11.2.1 int obs_t::nr

Number of ray paths.

Definition at line 375 of file jurassic.h.

4.11.2.2 double obs_t::time[NR]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 378 of file jurassic.h.

4.11.2.3 double obs_t::obsz[NR]

Observer altitude [km].

Definition at line 381 of file jurassic.h.

4.11.2.4 double obs_t::obslon[NR]

Observer longitude [deg].

Definition at line 384 of file jurassic.h.

4.11.2.5 double obs_t::obslat[NR]

Observer latitude [deg].

Definition at line 387 of file jurassic.h.

4.11.2.6 double obs_t::vpz[NR]

View point altitude [km].

Definition at line 390 of file jurassic.h.

4.11.2.7 double obs_t::vplon[NR]

View point longitude [deg].

Definition at line 393 of file jurassic.h.

4.11.2.8 double obs_t::vplat[NR]

View point latitude [deg].

Definition at line 396 of file jurassic.h.

4.11.2.9 double obs_t::tpz[NR]

Tangent point altitude [km].

Definition at line 399 of file jurassic.h.

4.11.2.10 double obs_t::tplon[NR]

Tangent point longitude [deg].

Definition at line 402 of file jurassic.h.

4.11.2.11 double obs_t::tplat[NR]

Tangent point latitude [deg].

Definition at line 405 of file jurassic.h.

4.11.2.12 double obs_t::tau[ND][NR]

Transmittance of ray path.

Definition at line 408 of file jurassic.h.

4.11.2.13 double obs_t::rad[ND][NR]

Radiance [W/(m^2 sr cm^-1)].

Definition at line 411 of file jurassic.h.

The documentation for this struct was generated from the following file:

- jurassic.h

4.12 pert_t Struct Reference

Perturbation data.

`#include <libiasi.h>`

Data Fields

- int ntrack

  Number of along-track values.
- int nxtrack

  Number of across-track values.
- double time [PERT_NTRACK][PERT_NXTRACK]

  Time (seconds since 2000-01-01T00:00Z).
- double lon [PERT_NTRACK][PERT_NXTRACK]

  Longitude [deg].
- double lat [PERT_NTRACK][PERT_NXTRACK]

  Latitude [deg].
- double dc [PERT_NTRACK][PERT_NXTRACK]

  Brightness temperature (8 micron) [K].
- double bt [PERT_NTRACK][PERT_NXTRACK]

  Brightness temperature (4 or 15 micron) [K].
- double pt [PERT_NTRACK][PERT_NXTRACK]

  Brightness temperature perturbation (4 or 15 micron) [K].
- double var [PERT_NTRACK][PERT_NXTRACK]

  Brightness temperature variance (4 or 15 micron) [K].

4.12.1 Detailed Description

Perturbation data.

Definition at line 142 of file libiasi.h.

4.12.2 Field Documentation

4.12.2.1 int pert_t::ntrack

Number of along-track values.

Definition at line 145 of file libiasi.h.

4.12.2.2 int pert_t::nxtrack

Number of across-track values.

Definition at line 148 of file libiasi.h.

4.12.2.3 double pert_t::time[PERT_NTRACK][PERT_NXTRACK]

Time (seconds since 2000-01-01T00:00Z).

Definition at line 151 of file libiasi.h.

4.12.2.4 double pert_t::lon[PERT_NTRACK][PERT_NXTRACK]

Longitude [deg].

Definition at line 154 of file libiasi.h.

4.12.2.5 double pert_t::lat[PERT_NTRACK][PERT_NXTRACK]

Latitude [deg].

Definition at line 157 of file libiasi.h.

4.12.2.6 double pert_t::dc[PERT_NTRACK][PERT_NXTRACK]

Brightness temperature (8 micron) [K].

Definition at line 160 of file libiasi.h.

4.12.2.7 double pert_t::bt[PERT_NTRACK][PERT_NXTRACK]

Brightness temperature (4 or 15 micron) [K].

Definition at line 163 of file libiasi.h.

4.12.2.8 double pert_t::pt[PERT_NTRACK][PERT_NXTRACK]

Brightness temperature perturbation (4 or 15 micron) [K].

Definition at line 166 of file libiasi.h.

4.12.2.9 double pert_t::var[PERT_NTRACK][PERT_NXTRACK]

Brightness temperature variance (4 or 15 micron) [K].

Definition at line 169 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

4.13 ret_t Struct Reference

Retrieval control parameters.

Data Fields

- int kernel_recomp

  Recomputation of kernel matrix (number of iterations).
- int conv_itmax

  Maximum number of iterations.
- double conv_dmin

  Minimum normalized step size in state space.
- double err_formod [ND]

  Forward model error [%].
- double err_noise [ND]

  Noise error [W/(m^2 sr cm^-1)].
- double err_press

  Pressure error [%].
- double err_press_cz

  Vertical correlation length for pressure error [km].
- double err_press_ch

  Horizontal correlation length for pressure error [km].
- double err_temp

  Temperature error [K].
- double err_temp_cz

  Vertical correlation length for temperature error [km].
- double err_temp_ch

  Horizontal correlation length for temperature error [km].
- double err_q [NG]

  Volume mixing ratio error [%].
- double err_q_cz [NG]

  Vertical correlation length for volume mixing ratio error [km].
- double err_q_ch [NG]

  Horizontal correlation length for volume mixing ratio error [km].
- double err_k [NW]

  Extinction error [1/km].
- double err_k_cz [NW]

  Vertical correlation length for extinction error [km].
- double err_k_ch [NW]

  Horizontal correlation length for extinction error [km].

4.13.1 Detailed Description

Retrieval control parameters.

Definition at line 102 of file retrieval.c.

4.13.2 Field Documentation

4.13.2.1 int ret_t::kernel_recomp

Recomputation of kernel matrix (number of iterations).

Definition at line 105 of file retrieval.c.

4.13.2.2 int ret_t::conv_itmax

Maximum number of iterations.

Definition at line 108 of file retrieval.c.

4.13.2.3 double ret_t::conv_dmin

Minimum normalized step size in state space.

Definition at line 111 of file retrieval.c.

4.13.2.4 double ret_t::err_formod[ND]

Forward model error [%].

Definition at line 114 of file retrieval.c.

4.13.2.5 double ret_t::err_noise[ND]

Noise error [W/(m^2 sr cm^-1)].

Definition at line 117 of file retrieval.c.

4.13.2.6 double ret_t::err_press

Pressure error [%].

Definition at line 120 of file retrieval.c.

4.13.2.7 double ret_t::err_press_cz

Vertical correlation length for pressure error [km].

Definition at line 123 of file retrieval.c.

4.13.2.8 double ret_t::err_press_ch

Horizontal correlation length for pressure error [km].

Definition at line 126 of file retrieval.c.

4.13.2.9 double ret_t::err_temp

Temperature error [K].

Definition at line 129 of file retrieval.c.

4.13.2.10 double ret_t::err_temp_cz

Vertical correlation length for temperature error [km].

Definition at line 132 of file retrieval.c.

4.13.2.11 double ret_t::err_temp_ch

Horizontal correlation length for temperature error [km].

Definition at line 135 of file retrieval.c.

4.13.2.12 double ret_t::err_q[NG]

Volume mixing ratio error [%].

Definition at line 138 of file retrieval.c.

4.13.2.13 double ret_t::err_q_cz[NG]

Vertical correlation length for volume mixing ratio error [km].

Definition at line 141 of file retrieval.c.

4.13.2.14 double ret_t::err_q_ch[NG]

Horizontal correlation length for volume mixing ratio error [km].

Definition at line 144 of file retrieval.c.

4.13.2.15 double ret_t::err_k[NW]

Extinction error [1/km].

Definition at line 147 of file retrieval.c.

4.13.2.16 double ret_t::err_k_cz[NW]

Vertical correlation length for extinction error [km].

Definition at line 150 of file retrieval.c.

4.13.2.17 double ret_t::err_k_ch[NW]

Horizontal correlation length for extinction error [km].

Definition at line 153 of file retrieval.c.

The documentation for this struct was generated from the following file:

- retrieval.c

4.14 tbl_t Struct Reference

Emissivity look-up tables.

`#include <jurassic.h>`

Data Fields

- int np [NG][ND]

  Number of pressure levels.
- int nt [NG][ND][TBLNP]

  Number of temperatures.
- int nu [NG][ND][TBLNP][TBLNT]

  Number of column densities.
- double p [NG][ND][TBLNP]

  Pressure [hPa].
- double t [NG][ND][TBLNP][TBLNT]

  Temperature [K].
- float u [NG][ND][TBLNP][TBLNT][TBLNU]

  Column density [molecules/cm^ 2].
- float eps [NG][ND][TBLNP][TBLNT][TBLNU]

  Emissivity.
- double st [TBLNS]

  Source function temperature [K].
- double sr [ND][TBLNS]

  Source function radiance [W/(m^ 2 sr cm^ -1)].

4.14.1 Detailed Description

Emissivity look-up tables.

Definition at line 416 of file jurassic.h.

4.14.2 Field Documentation

4.14.2.1 int tbl_t::np[NG][ND]

Number of pressure levels.

Definition at line 419 of file jurassic.h.

4.14.2.2 int tbl_t::nt[NG][ND][TBLNP]

Number of temperatures.

Definition at line 422 of file jurassic.h.

4.14.2.3 int tbl_t::nu[NG][ND][TBLNP][TBLNT]

Number of column densities.

Definition at line 425 of file jurassic.h.

4.14.2.4 double tbl_t::p[NG][ND][TBLNP]

Pressure [hPa].

Definition at line 428 of file jurassic.h.

4.14.2.5 double tbl_t::t[NG][ND][TBLNP][TBLNT]

Temperature [K].

Definition at line 431 of file jurassic.h.

4.14.2.6 float tbl_t::u[NG][ND][TBLNP][TBLNT][TBLNU]

Column density [molecules/cm^2].

Definition at line 434 of file jurassic.h.

4.14.2.7 float tbl_t::eps[NG][ND][TBLNP][TBLNT][TBLNU]

Emissivity.

Definition at line 437 of file jurassic.h.

4.14.2.8 double tbl_t::st[TBLNS]

Source function temperature [K].

Definition at line 440 of file jurassic.h.

4.14.2.9 double tbl_t::sr[ND][TBLNS]

Source function radiance [W/(m^2 sr cm^-1)].

Definition at line 443 of file jurassic.h.

The documentation for this struct was generated from the following file:

- jurassic.h

4.15 wave_t Struct Reference

Wave analysis data.

```
#include <libiasi.h>
```

Data Fields

- int nx

  Number of across-track values.
- int ny

  Number of along-track values.
- double time

  Time (seconds since 2000-01-01T00:00Z).
- double z

  Altitude [km].
- double lon [WX][WY]

  Longitude [deg].
- double lat [WX][WY]

  Latitude [deg].
- double x [WX]

  Across-track distance [km].
- double y [WY]

  Along-track distance [km].
- double temp [WX][WY]

  Temperature [K].
- double bg [WX][WY]

  Background [K].
- double pt [WX][WY]

  Perturbation [K].
- double var [WX][WY]

  Variance [K].

4.15.1 Detailed Description

Wave analysis data.

Definition at line 238 of file libiasi.h.

4.15.2 Field Documentation

4.15.2.1 int wave_t::nx

Number of across-track values.

Definition at line 241 of file libiasi.h.

4.15.2.2 int wave_t::ny

Number of along-track values.

Definition at line 244 of file libiasi.h.

4.15.2.3 double wave_t::time

Time (seconds since 2000-01-01T00:00Z).

Definition at line 247 of file libiasi.h.

4.15.2.4 double wave_t::z

Altitude [km].

Definition at line 250 of file libiasi.h.

4.15.2.5 double wave_t::lon[WX][WY]

Longitude [deg].

Definition at line 253 of file libiasi.h.

4.15.2.6 double wave_t::lat[WX][WY]

Latitude [deg].

Definition at line 256 of file libiasi.h.

4.15.2.7 double wave_t::x[WX]

Across-track distance [km].

Definition at line 259 of file libiasi.h.

4.15.2.8 double wave_t::y[WY]

Along-track distance [km].

Definition at line 262 of file libiasi.h.

4.15.2.9 double wave_t::temp[WX][WY]

Temperature [K].

Definition at line 265 of file libiasi.h.

4.15.2.10 double wave_t::bg[WX][WY]

Background [K].

Definition at line 268 of file libiasi.h.

4.15.2.11 double wave_t::pt[WX][WY]

Perturbation [K].

Definition at line 271 of file libiasi.h.

4.15.2.12 double wave_t::var[WX][WY]

Variance [K].

Definition at line 274 of file libiasi.h.

The documentation for this struct was generated from the following file:

- libiasi.h

5 File Documentation

5.1 bands.c File Reference

Functions

- int main (int argc, char *argv[])

5.1.1 Function Documentation

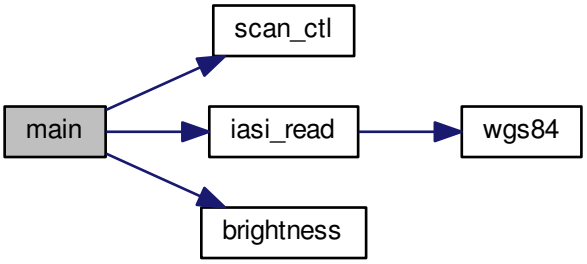
5.1.1.1 int main (int *argc*, char * *argv[]*)

Definition at line 14 of file bands.c.

```
00016                {
00017
00018   static iasi_rad_t *iasi_rad;
00019
00020   static FILE *out;
00021
00022   static double numin[NB], numax[NB], rad[NB];
00023
00024   static int iarg, ib, ichan, n, nb, track, xtrack;
00025
00026   /* Check arguments... */
00027   if (argc < 4)
00028     ERRMSG("Give parameters: <ctl> <out.tab> <l1b_file1> [<l1b_file2> ...]");
00029
00030   /* Allocate... */
00031   ALLOC(iasi_rad, iasi_rad_t, 1);
00032
00033   /* Get control parameters... */
00034   nb = (int) scan_ctl(argc, argv, "NB", -1, "1", NULL);
00035   if (nb > NB)
00036     ERRMSG("Too many bands!");
00037   for (ib = 0; ib < nb; ib++) {
00038     numin[ib] = scan_ctl(argc, argv, "NUMIN", ib, "", NULL);
00039     numax[ib] = scan_ctl(argc, argv, "NUMAX", ib, "", NULL);
00040   }
00041
00042   /* Create file... */
00043   printf("Write band data: %s\n", argv[2]);
00044   if (!(out = fopen(argv[2], "w")))
00045     ERRMSG("Cannot create file!");
00046
00047   /* Loop over IASI files... */
00048   for (iarg = 3; iarg < argc; iarg++) {
00049
00050     /* Read IASI data... */
00051     printf("Read IASI Level-1C data file: %s\n", argv[iarg]);
00052     iasi_read(argv[iarg], iasi_rad);
00053
00054     /* Write header... */
00055     if (iarg == 3) {
00056       fprintf(out,
```

```
00057               "# $1 = time [s]\n"
00058               "# $2 = footprint longitude [deg]\n"
00059               "# $3 = footprint latitude [deg]\n"
00060               "# $4 = satellite altitude [km]\n"
00061               "# $5 = satellite longitude [deg]\n"
00062               "# $6 = satellite latitude [deg]\n");
00063       for (ib = 0; ib < nb; ib++)
00064         fprintf(out,
00065                 "# $%d = BT(%.2f/cm...%.2f/cm) [K]\n",
00066                 7 + ib, numin[ib], numax[ib]);
00067     }
00068
00069     /* Loop over scans... */
00070     for (track = 0; track < iasi_rad->ntrack; track++) {
00071
00072       /* Write output... */
00073       fprintf(out, "\n");
00074
00075       /* Loop over footprints... */
00076       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00077
00078         /* Write output... */
00079         fprintf(out, "%.2f %.4f %.4f %.3f %.4f %.4f",
00080                 iasi_rad->Time[track][xtrack],
00081                 iasi_rad->Longitude[track][xtrack],
00082                 iasi_rad->Latitude[track][xtrack],
00083                 iasi_rad->Sat_z[track],
00084                 iasi_rad->Sat_lon[track], iasi_rad->Sat_lat[track]);
00085
00086         /* Loop over bands... */
00087         for (ib = 0; ib < nb; ib++) {
00088
00089           /* Get mean radiance... */
00090           n = 0;
00091           rad[ib] = 0;
00092           for (ichan = 0; ichan <= IASI_L1_NCHAN; ichan++)
00093             if (iasi_rad->freq[ichan] >= numin[ib]
00094                 && iasi_rad->freq[ichan] <= numax[ib]
00095                 && gsl_finite(iasi_rad->Rad[track][xtrack][ichan])) {
00096               rad[ib] += iasi_rad->Rad[track][xtrack][ichan];
00097               n++;
00098             }
00099           if (n > 0)
00100             rad[ib] /= n;
00101           else
00102             rad[ib] = GSL_NAN;
00103
00104           /* Convert to brightness temperature... */
00105           rad[ib] = brightness(rad[ib], 0.5 * (numin[ib] + numax[ib]));
00106
00107           /* Write output... */
00108           fprintf(out, " %.3f", rad[ib]);
00109         }
00110
00111         /* Write output... */
00112         fprintf(out, "\n");
00113       }
00114     }
00115   }
00116
00117   /* Close file... */
00118   fclose(out);
00119
00120   /* Free... */
00121   free(iasi_rad);
00122
00123   return EXIT_SUCCESS;
00124 }
```

Here is the call graph for this function:



5.2 bands.c

```
00001 #include "libiasi.h"
00002
00003 /* ------------------------------------------------------------
00004    Dimensions...
00005    ------------------------------------------------------------ */
00006
00007 /* Maximum number of bands... */
00008 #define NB 100
00009
00010 /* ------------------------------------------------------------
00011    Main...
00012    ------------------------------------------------------------ */
00013
00014 int main(
00015   int argc,
00016   char *argv[]) {
00017
00018   static iasi_rad_t *iasi_rad;
00019
00020   static FILE *out;
00021
00022   static double numin[NB], numax[NB], rad[NB];
00023
00024   static int iarg, ib, ichan, n, nb, track, xtrack;
00025
00026   /* Check arguments... */
00027   if (argc < 4)
00028     ERRMSG("Give parameters: <ctl> <out.tab> <l1b_file1> [<l1b_file2> ...]");
00029
00030   /* Allocate... */
00031   ALLOC(iasi_rad, iasi_rad_t, 1);
00032
00033   /* Get control parameters... */
00034   nb = (int) scan_ctl(argc, argv, "NB", -1, "1", NULL);
00035   if (nb > NB)
00036     ERRMSG("Too many bands!");
00037   for (ib = 0; ib < nb; ib++) {
00038     numin[ib] = scan_ctl(argc, argv, "NUMIN", ib, "", NULL);
00039     numax[ib] = scan_ctl(argc, argv, "NUMAX", ib, "", NULL);
00040   }
00041
00042   /* Create file... */
00043   printf("Write band data: %s\n", argv[2]);
00044   if (!(out = fopen(argv[2], "w")))
00045     ERRMSG("Cannot create file!");
00046
00047   /* Loop over IASI files... */
00048   for (iarg = 3; iarg < argc; iarg++) {
00049
00050     /* Read IASI data... */
00051     printf("Read IASI Level-1C data file: %s\n", argv[iarg]);
00052     iasi_read(argv[iarg], iasi_rad);
00053
00054     /* Write header... */
```

```
00055     if (iarg == 3) {
00056       fprintf(out,
00057               "# $1 = time [s]\n"
00058               "# $2 = footprint longitude [deg]\n"
00059               "# $3 = footprint latitude [deg]\n"
00060               "# $4 = satellite altitude [km]\n"
00061               "# $5 = satellite longitude [deg]\n"
00062               "# $6 = satellite latitude [deg]\n");
00063       for (ib = 0; ib < nb; ib++)
00064         fprintf(out,
00065                 "# $%d = BT(%.2f/cm...%.2f/cm) [K]\n",
00066                 7 + ib, numin[ib], numax[ib]);
00067     }
00068
00069     /* Loop over scans... */
00070     for (track = 0; track < iasi_rad->ntrack; track++) {
00071
00072       /* Write output... */
00073       fprintf(out, "\n");
00074
00075       /* Loop over footprints... */
00076       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00077
00078         /* Write output... */
00079         fprintf(out, "%.2f %.4f %.4f %.4f %.3f %.4f %.4f",
00080                 iasi_rad->Time[track][xtrack],
00081                 iasi_rad->Longitude[track][xtrack],
00082                 iasi_rad->Latitude[track][xtrack],
00083                 iasi_rad->Sat_z[track],
00084                 iasi_rad->Sat_lon[track], iasi_rad->Sat_lat[track]);
00085
00086         /* Loop over bands... */
00087         for (ib = 0; ib < nb; ib++) {
00088
00089           /* Get mean radiance... */
00090           n = 0;
00091           rad[ib] = 0;
00092           for (ichan = 0; ichan <= IASI_L1_NCHAN; ichan++)
00093             if (iasi_rad->freq[ichan] >= numin[ib]
00094                 && iasi_rad->freq[ichan] <= numax[ib]
00095                 && gsl_finite(iasi_rad->Rad[track][xtrack][ichan])) {
00096               rad[ib] += iasi_rad->Rad[track][xtrack][ichan];
00097               n++;
00098             }
00099           if (n > 0)
00100             rad[ib] /= n;
00101           else
00102             rad[ib] = GSL_NAN;
00103
00104           /* Convert to brightness temperature... */
00105           rad[ib] = brightness(rad[ib], 0.5 * (numin[ib] + numax[ib]));
00106
00107           /* Write output... */
00108           fprintf(out, " %.3f", rad[ib]);
00109         }
00110
00111         /* Write output... */
00112         fprintf(out, "\n");
00113       }
00114     }
00115   }
00116
00117   /* Close file... */
00118   fclose(out);
00119
00120   /* Free... */
00121   free(iasi_rad);
00122
00123   return EXIT_SUCCESS;
00124 }
```

5.3 extract.c File Reference

Data Structures

- struct ctl2_t

  Control parameters.

- struct met_t

  Meteorological data.

Functions

- void get_met (ctl2_t *ctl2, char *metbase, double t, met_t *met0, met_t *met1)

  Get meteorological data for given timestep.
- void get_met_help (double t, int direct, char *metbase, double dt_met, char *filename)

  Get meteorological data for timestep.
- void intpol_met_2d (double array[EX][EY], int ix, int iy, double wx, double wy, double *var)

  Linear interpolation of 2-D meteorological data.
- void intpol_met_3d (float array[EX][EY][EP], int ip, int ix, int iy, double wp, double wx, double wy, double *var)

  Linear interpolation of 3-D meteorological data.
- void intpol_met_space (met_t *met, double p, double lon, double lat, double *ps, double *pt, double *z, double *t, double *u, double *v, double *w, double *pv, double *h2o, double *o3)

  Spatial interpolation of meteorological data.
- void intpol_met_time (met_t *met0, met_t *met1, double ts, double p, double lon, double lat, double *ps, double *pt, double *z, double *t, double *u, double *v, double *w, double *pv, double *h2o, double *o3)

  Temporal interpolation of meteorological data.
- void read_ctl2 (int argc, char *argv[ ], ctl2_t *ctl2)

  Read control parameters.
- void read_met (ctl2_t *ctl2, char *filename, met_t *met)

  Read meteorological data file.
- void read_met_extrapolate (met_t *met)

  Extrapolate meteorological data at lower boundary.
- void read_met_geopot (ctl2_t *ctl2, met_t *met)

  Calculate geopotential heights.
- void read_met_help (int ncid, char *varname, char *varname2, met_t *met, float dest[EX][EY][EP], float scl)

  Read and convert variable from meteorological data file.
- void read_met_periodic (met_t *met)

  Create meteorological data with periodic boundary conditions.
- void read_met_sample (ctl2_t *ctl2, met_t *met)

  Downsampling of meteorological data.
- int main (int argc, char *argv[ ])

Variables

- int iasi_chan [L1_NCHAN]

  List of IASI channels (don't change).

5.3.1 Function Documentation

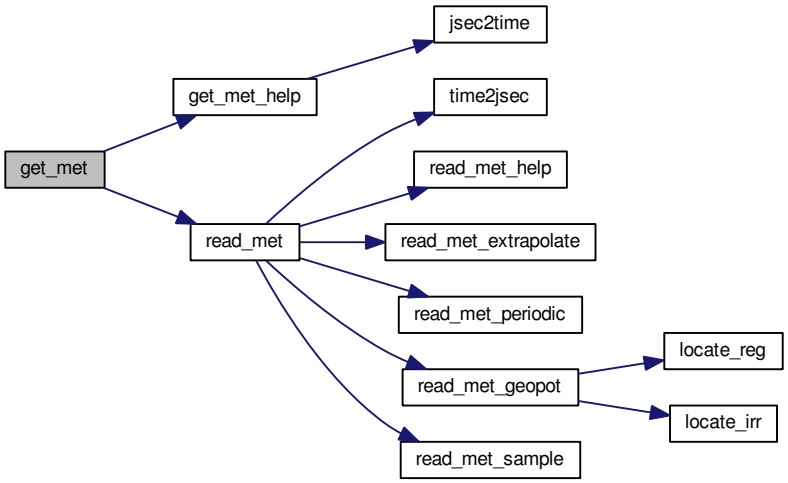
5.3.1.1 void get_met (ctl2_t * *ctl2*, char * *metbase*, double *t*, met_t * *met0*, met_t * *met1*)

Get meteorological data for given timestep.

Definition at line 330 of file extract.c.

```
00335                {
00336
00337   char filename[LEN];
00338
00339   static int init;
00340
00341   /* Init... */
00342   if (!init) {
00343     init = 1;
00344
00345     get_met_help(t, -1, metbase, ctl2->dt_met, filename);
00346     read_met(ctl2, filename, met0);
00347
00348     get_met_help(t + 1.0, 1, metbase, ctl2->dt_met, filename);
00349     read_met(ctl2, filename, met1);
00350   }
00351
00352   /* Read new data for forward trajectories... */
00353   if (t > met1->time) {
00354     memcpy(met0, met1, sizeof(met_t));
00355     get_met_help(t, 1, metbase, ctl2->dt_met, filename);
00356     read_met(ctl2, filename, met1);
00357   }
00358 }
```

Here is the call graph for this function:



5.3.1.2 void get_met_help (double *t*, int *direct*, char * *metbase*, double *dt_met*, char * *filename*)

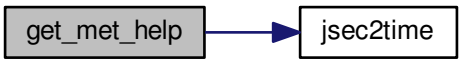
Get meteorological data for timestep.

Definition at line 362 of file extract.c.

```
00367                  {
00368
00369   double t6, r;
00370
00371   int year, mon, day, hour, min, sec;
00372
00373   /* Round time to fixed intervals... */
00374   if (direct == -1)
00375     t6 = floor(t / dt_met) * dt_met;
00376   else
00377     t6 = ceil(t / dt_met) * dt_met;
00378
```

```
00379   /* Decode time... */
00380   jsec2time(t6, &year, &mon, &day, &hour, &min, &sec, &r);
00381
00382   /* Set filename... */
00383   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", metbase, year, mon, day, hour);
00384 }
```

Here is the call graph for this function:



5.3.1.3 void intpol_met_2d (double *array[EX][EY]*, int *ix*, int *iy*, double *wx*, double *wy*, double * *var*)

Linear interpolation of 2-D meteorological data.

Definition at line 388 of file extract.c.

```
00394               {
00395
00396   double aux00, aux01, aux10, aux11;
00397
00398   /* Set variables... */
00399   aux00 = array[ix][iy];
00400   aux01 = array[ix][iy + 1];
00401   aux10 = array[ix + 1][iy];
00402   aux11 = array[ix + 1][iy + 1];
00403
00404   /* Interpolate horizontally... */
00405   aux00 = wy * (aux00 - aux01) + aux01;
00406   aux11 = wy * (aux10 - aux11) + aux11;
00407   *var = wx * (aux00 - aux11) + aux11;
00408 }
```

5.3.1.4 void intpol_met_3d (float *array[EX][EY][EP]*, int *ip*, int *ix*, int *iy*, double *wp*, double *wx*, double *wy*, double * *var*)

Linear interpolation of 3-D meteorological data.

Definition at line 412 of file extract.c.

```
00420               {
00421
00422   double aux00, aux01, aux10, aux11;
00423
00424   /* Interpolate vertically... */
00425   aux00 = wp * (array[ix][iy][ip] - array[ix][iy][ip + 1])
00426     + array[ix][iy][ip + 1];
00427   aux01 = wp * (array[ix][iy + 1][ip] - array[ix][iy + 1][ip + 1])
00428     + array[ix][iy + 1][ip + 1];
00429   aux10 = wp * (array[ix + 1][iy][ip] - array[ix + 1][iy][ip + 1])
00430     + array[ix + 1][iy][ip + 1];
00431   aux11 = wp * (array[ix + 1][iy + 1][ip] - array[ix + 1][iy + 1][ip + 1])
00432     + array[ix + 1][iy + 1][ip + 1];
00433
00434   /* Interpolate horizontally... */
00435   aux00 = wy * (aux00 - aux01) + aux01;
00436   aux11 = wy * (aux10 - aux11) + aux11;
00437   *var = wx * (aux00 - aux11) + aux11;
00438 }
```

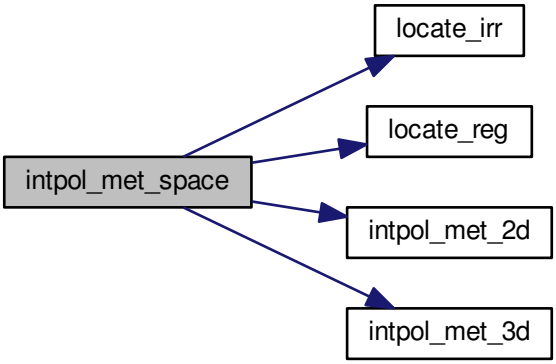
5.3.1.5 void intpol_met_space (met_t * *met*, double *p*, double *lon*, double *lat*, double * *ps*, double * *pt*, double * *z*, double * *t*, double * *u*, double * *v*, double * *w*, double * *pv*, double * *h2o*, double * *o3*)

Spatial interpolation of meteorological data.

Definition at line 442 of file extract.c.

```
00456              {
00457
00458   double wp, wx, wy;
00459
00460   int ip, ix, iy;
00461
00462   /* Check longitude... */
00463   if (met->lon[met->nx - 1] > 180 && lon < 0)
00464     lon += 360;
00465
00466   /* Get indices... */
00467   ip = locate_irr(met->p, met->np, p);
00468   ix = locate_reg(met->lon, met->nx, lon);
00469   iy = locate_reg(met->lat, met->ny, lat);
00470
00471   /* Get weights... */
00472   wp = (met->p[ip + 1] - p) / (met->p[ip + 1] - met->p[ip]);
00473   wx = (met->lon[ix + 1] - lon) / (met->lon[ix + 1] - met->lon[ix]);
00474   wy = (met->lat[iy + 1] - lat) / (met->lat[iy + 1] - met->lat[iy]);
00475
00476   /* Interpolate... */
00477   if (ps != NULL)
00478     intpol_met_2d(met->ps, ix, iy, wx, wy, ps);
00479   if (pt != NULL)
00480     intpol_met_2d(met->pt, ix, iy, wx, wy, pt);
00481   if (z != NULL)
00482     intpol_met_3d(met->z, ip, ix, iy, wp, wx, wy, z);
00483   if (t != NULL)
00484     intpol_met_3d(met->t, ip, ix, iy, wp, wx, wy, t);
00485   if (u != NULL)
00486     intpol_met_3d(met->u, ip, ix, iy, wp, wx, wy, u);
00487   if (v != NULL)
00488     intpol_met_3d(met->v, ip, ix, iy, wp, wx, wy, v);
00489   if (w != NULL)
00490     intpol_met_3d(met->w, ip, ix, iy, wp, wx, wy, w);
00491   if (pv != NULL)
00492     intpol_met_3d(met->pv, ip, ix, iy, wp, wx, wy, pv);
00493   if (h2o != NULL)
00494     intpol_met_3d(met->h2o, ip, ix, iy, wp, wx, wy, h2o);
00495   if (o3 != NULL)
00496     intpol_met_3d(met->o3, ip, ix, iy, wp, wx, wy, o3);
00497 }
```

Here is the call graph for this function:



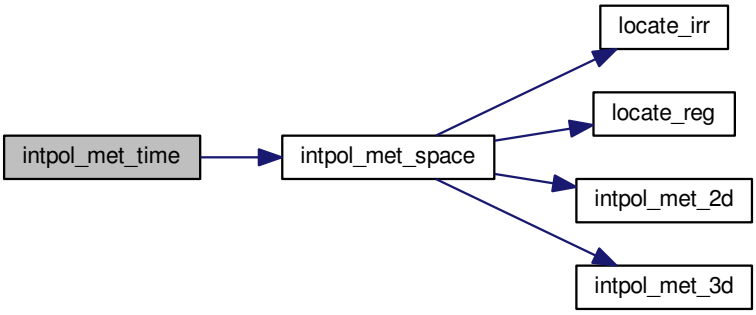
5.3.1.6 void intpol_met_time (met_t * *met0*, met_t * *met1*, double *ts*, double *p*, double *lon*, double *lat*, double * *ps*, double * *pt*, double * *z*, double * *t*, double * *u*, double * *v*, double * *w*, double * *pv*, double * *h2o*, double * *o3*)

Temporal interpolation of meteorological data.

Definition at line 501 of file extract.c.

```
00517             {
00518
00519   double h2o0, h2o1, o30, o31, ps0, ps1, pt0, pt1, pv0, pv1, t0, t1, u0, u1,
00520     v0, v1, w0, w1, wt, z0, z1;
00521
00522   /* Spatial interpolation... */
00523   intpol_met_space(met0, p, lon, lat,
00524                    ps == NULL ? NULL : &ps0,
00525                    pt == NULL ? NULL : &pt0,
00526                    z == NULL ? NULL : &z0,
00527                    t == NULL ? NULL : &t0,
00528                    u == NULL ? NULL : &u0,
00529                    v == NULL ? NULL : &v0,
00530                    w == NULL ? NULL : &w0,
00531                    pv == NULL ? NULL : &pv0,
00532                    h2o == NULL ? NULL : &h2o0, o3 == NULL ? NULL : &o30);
00533   intpol_met_space(met1, p, lon, lat,
00534                    ps == NULL ? NULL : &ps1,
00535                    pt == NULL ? NULL : &pt1,
00536                    z == NULL ? NULL : &z1,
00537                    t == NULL ? NULL : &t1,
00538                    u == NULL ? NULL : &u1,
00539                    v == NULL ? NULL : &v1,
00540                    w == NULL ? NULL : &w1,
00541                    pv == NULL ? NULL : &pv1,
00542                    h2o == NULL ? NULL : &h2o1, o3 == NULL ? NULL : &o31);
00543
00544   /* Get weighting factor... */
00545   wt = (met1->time - ts) / (met1->time - met0->time);
00546
00547   /* Interpolate... */
00548   if (ps != NULL)
00549     *ps = wt * (ps0 - ps1) + ps1;
00550   if (pt != NULL)
00551     *pt = wt * (pt0 - pt1) + pt1;
00552   if (z != NULL)
00553     *z = wt * (z0 - z1) + z1;
00554   if (t != NULL)
00555     *t = wt * (t0 - t1) + t1;
00556   if (u != NULL)
00557     *u = wt * (u0 - u1) + u1;
00558   if (v != NULL)
00559     *v = wt * (v0 - v1) + v1;
00560   if (w != NULL)
00561     *w = wt * (w0 - w1) + w1;
00562   if (pv != NULL)
00563     *pv = wt * (pv0 - pv1) + pv1;
00564   if (h2o != NULL)
00565     *h2o = wt * (h2o0 - h2o1) + h2o1;
00566   if (o3 != NULL)
00567     *o3 = wt * (o30 - o31) + o31;
00568 }
```

Here is the call graph for this function:



5.3.1.7 void read_ctl2 (int *argc*, char * *argv[ ]*, ctl2_t * *ctl2*)

Read control parameters.

Definition at line 572 of file extract.c.

```
00575                 {
00576
00577   /* Meteorological data... */
00578   ctl2->dt_met = scan_ctl(argc, argv, "DT_MET", -1, "21600", NULL);
00579   scan_ctl(argc, argv, "MET_GEOPOT", -1, "", ctl2->met_geopot);
00580   ctl2->met_dx = (int) scan_ctl(argc, argv, "MET_DX", -1, "1", NULL);
00581   ctl2->met_dy = (int) scan_ctl(argc, argv, "MET_DY", -1, "1", NULL);
00582   ctl2->met_dp = (int) scan_ctl(argc, argv, "MET_DP", -1, "1", NULL);
00583   ctl2->met_sx = (int) scan_ctl(argc, argv, "MET_SX", -1, "20", NULL);
00584   ctl2->met_sy = (int) scan_ctl(argc, argv, "MET_SY", -1, "10", NULL);
00585   ctl2->met_sp = (int) scan_ctl(argc, argv, "MET_SP", -1, "1", NULL);
00586 }
```

Here is the call graph for this function:

5.3.1.8 void read_met (ctl2_t * *ctl2*, char * *filename*, met_t * *met*)

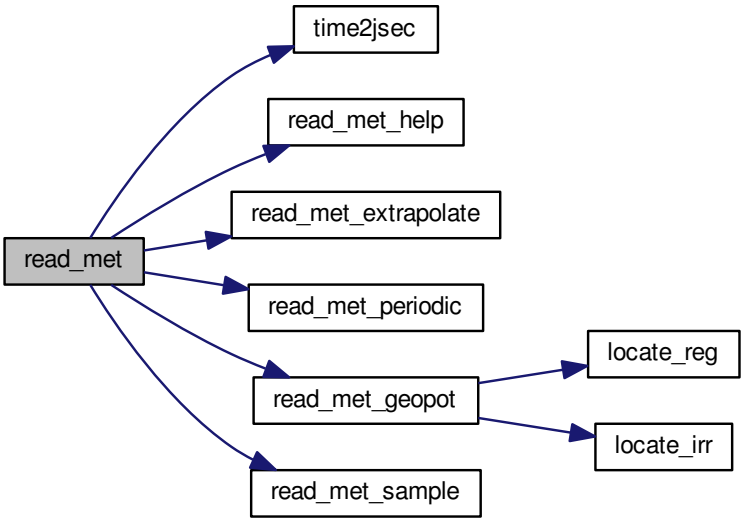
Read meteorological data file.

Definition at line 590 of file extract.c.

```
00593               {
00594
00595   char levname[LEN], tstr[10];
00596
00597   static float help[EX * EY];
00598
00599   int ix, iy, ip, dimid, ncid, varid, year, mon, day, hour;
00600
00601   size_t np, nx, ny;
00602
00603   /* Write info... */
00604   printf("Read meteorological data: %s\n", filename);
00605
00606   /* Get time from filename... */
00607   sprintf(tstr, "%.4s", &filename[strlen(filename) - 16]);
00608   year = atoi(tstr);
00609   sprintf(tstr, "%.2s", &filename[strlen(filename) - 11]);
00610   mon = atoi(tstr);
00611   sprintf(tstr, "%.2s", &filename[strlen(filename) - 8]);
00612   day = atoi(tstr);
00613   sprintf(tstr, "%.2s", &filename[strlen(filename) - 5]);
00614   hour = atoi(tstr);
00615   time2jsec(year, mon, day, hour, 0, 0, 0, &met->time);
00616
00617   /* Open netCDF file... */
00618   NC(nc_open(filename, NC_NOWRITE, &ncid));
00619
00620   /* Get dimensions... */
00621   NC(nc_inq_dimid(ncid, "lon", &dimid));
00622   NC(nc_inq_dimlen(ncid, dimid, &nx));
00623   if (nx < 2 || nx > EX)
00624     ERRMSG("Number of longitudes out of range!");
00625
00626   NC(nc_inq_dimid(ncid, "lat", &dimid));
00627   NC(nc_inq_dimlen(ncid, dimid, &ny));
00628   if (ny < 2 || ny > EY)
00629     ERRMSG("Number of latitudes out of range!");
00630
00631   sprintf(levname, "lev");
00632   NC(nc_inq_dimid(ncid, levname, &dimid));
00633   NC(nc_inq_dimlen(ncid, dimid, &np));
00634   if (np == 1) {
00635     sprintf(levname, "lev_2");
00636     NC(nc_inq_dimid(ncid, levname, &dimid));
00637     NC(nc_inq_dimlen(ncid, dimid, &np));
00638   }
00639   if (np < 2 || np > EP)
00640     ERRMSG("Number of levels out of range!");
00641
00642   /* Store dimensions... */
00643   met->np = (int) np;
00644   met->nx = (int) nx;
00645   met->ny = (int) ny;
00646
00647   /* Get horizontal grid... */
00648   NC(nc_inq_varid(ncid, "lon", &varid));
00649   NC(nc_get_var_double(ncid, varid, met->lon));
00650   NC(nc_inq_varid(ncid, "lat", &varid));
00651   NC(nc_get_var_double(ncid, varid, met->lat));
00652
00653   /* Read meteorological data... */
00654   read_met_help(ncid, "t", "T", met, met->t, 1.0);
00655   read_met_help(ncid, "u", "U", met, met->u, 1.0);
00656   read_met_help(ncid, "v", "V", met, met->v, 1.0);
00657   read_met_help(ncid, "w", "W", met, met->w, 0.01f);
00658   read_met_help(ncid, "q", "Q", met, met->h2o, 1.608f);
00659   read_met_help(ncid, "o3", "O3", met, met->o3, 0.602f);
00660
00661   /* Read pressure levels from file... */
00662   NC(nc_inq_varid(ncid, levname, &varid));
00663   NC(nc_get_var_double(ncid, varid, met->p));
00664   for (ip = 0; ip < met->np; ip++)
00665     met->p[ip] /= 100.;
00666
00667   /* Extrapolate data for lower boundary... */
00668   read_met_extrapolate(met);
00669
00670   /* Check ordering of pressure levels... */
00671   for (ip = 1; ip < met->np; ip++)
00672     if (met->p[ip - 1] < met->p[ip])
00673       ERRMSG("Pressure levels must be descending!");
00674
00675   /* Read surface pressure... */
00676   if (nc_inq_varid(ncid, "ps", &varid) == NC_NOERR
00677       || nc_inq_varid(ncid, "PS", &varid) == NC_NOERR) {
00678     NC(nc_get_var_float(ncid, varid, help));
00679     for (iy = 0; iy < met->ny; iy++)
```

```
00680       for (ix = 0; ix < met->nx; ix++)
00681         met->ps[ix][iy] = help[iy * met->nx + ix] / 100.;
00682     } else if (nc_inq_varid(ncid, "lnsp", &varid) == NC_NOERR
00683                || nc_inq_varid(ncid, "LNSP", &varid) == NC_NOERR) {
00684       NC(nc_get_var_float(ncid, varid, help));
00685       for (iy = 0; iy < met->ny; iy++)
00686         for (ix = 0; ix < met->nx; ix++)
00687           met->ps[ix][iy] = exp(help[iy * met->nx + ix]) / 100.;
00688     } else
00689       for (ix = 0; ix < met->nx; ix++)
00690         for (iy = 0; iy < met->ny; iy++)
00691           met->ps[ix][iy] = met->p[0];
00692
00693     /* Create periodic boundary conditions... */
00694     read_met_periodic(met);
00695
00696     /* Calculate geopotential heights... */
00697     read_met_geopot(ctl2, met);
00698
00699     /* Downsampling... */
00700     read_met_sample(ctl2, met);
00701
00702     /* Close file... */
00703     NC(nc_close(ncid));
00704   }
```

Here is the call graph for this function:



5.3.1.9 void read_met_extrapolate (met_t * *met*)

Extrapolate meteorological data at lower boundary.

Definition at line 708 of file extract.c.

```
00709                 {
00710
00711     int ip, ip0, ix, iy;
00712
00713     /* Loop over columns... */
00714     for (ix = 0; ix < met->nx; ix++)
00715       for (iy = 0; iy < met->ny; iy++) {
00716
```

```
00717       /* Find lowest valid data point... */
00718       for (ip0 = met->np - 1; ip0 >= 0; ip0--)
00719         if (!gsl_finite(met->t[ix][iy][ip0])
00720             || !gsl_finite(met->u[ix][iy][ip0])
00721             || !gsl_finite(met->v[ix][iy][ip0])
00722             || !gsl_finite(met->w[ix][iy][ip0]))
00723           break;
00724
00725       /* Extrapolate... */
00726       for (ip = ip0; ip >= 0; ip--) {
00727         met->t[ix][iy][ip] = met->t[ix][iy][ip + 1];
00728         met->u[ix][iy][ip] = met->u[ix][iy][ip + 1];
00729         met->v[ix][iy][ip] = met->v[ix][iy][ip + 1];
00730         met->w[ix][iy][ip] = met->w[ix][iy][ip + 1];
00731         met->h2o[ix][iy][ip] = met->h2o[ix][iy][ip + 1];
00732         met->o3[ix][iy][ip] = met->o3[ix][iy][ip + 1];
00733       }
00734     }
00735 }
```

5.3.1.10 void read_met_geopot (ctl2_t * *ctl2*, met_t * *met*)

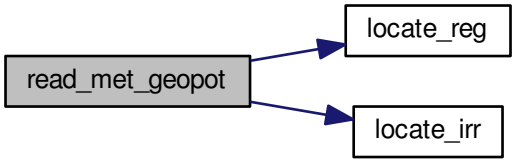
Calculate geopotential heights.

Definition at line 739 of file extract.c.

```
00741               {
00742
00743   static double topo_lat[EY], topo_lon[EX], topo_z[EX][EY];
00744
00745   static int init, topo_nx = -1, topo_ny;
00746
00747   FILE *in;
00748
00749   char line[LEN];
00750
00751   double data[30], lat, lon, rlat, rlon, rlon_old = -999, rz, ts, z0, z1;
00752
00753   float help[EX][EY];
00754
00755   int ip, ip0, ix, ix2, ix3, iy, iy2, n, tx, ty;
00756
00757   /* Initialize geopotential heights... */
00758   for (ix = 0; ix < met->nx; ix++)
00759     for (iy = 0; iy < met->ny; iy++)
00760       for (ip = 0; ip < met->np; ip++)
00761         met->z[ix][iy][ip] = GSL_NAN;
00762
00763   /* Check filename... */
00764   if (ctl2->met_geopot[0] == '-')
00765     return;
00766
00767   /* Read surface geopotential... */
00768   if (!init) {
00769
00770     /* Write info... */
00771     printf("Read surface geopotential: %s\n", ctl2->met_geopot);
00772
00773     /* Open file... */
00774     if (!(in = fopen(ctl2->met_geopot, "r")))
00775       ERRMSG("Cannot open file!");
00776
00777     /* Read data... */
00778     while (fgets(line, LEN, in))
00779       if (sscanf(line, "%lg %lg %lg", &rlon, &rlat, &rz) == 3) {
00780         if (rlon != rlon_old) {
00781           if ((++topo_nx) >= EX)
00782             ERRMSG("Too many longitudes!");
00783           topo_ny = 0;
00784         }
00785         rlon_old = rlon;
00786         topo_lon[topo_nx] = rlon;
00787         topo_lat[topo_ny] = rlat;
00788         topo_z[topo_nx][topo_ny] = rz;
00789         if ((++topo_ny) >= EY)
00790           ERRMSG("Too many latitudes!");
00791       }
00792     if ((++topo_nx) >= EX)
00793       ERRMSG("Too many longitudes!");
00794
```

```
00795     /* Close file... */
00796     fclose(in);
00797
00798     /* Check grid spacing... */
00799     if (fabs(met->lon[0] - met->lon[1]) != fabs(topo_lon[0] - topo_lon[1])
00800         || fabs(met->lat[0] - met->lat[1]) != fabs(topo_lat[0] - topo_lat[1]))
00801       printf("Warning: Grid spacing does not match!\n");
00802
00803     /* Set init flag... */
00804     init = 1;
00805   }
00806
00807   /* Apply hydrostatic equation to calculate geopotential heights... */
00808   for (ix = 0; ix < met->nx; ix++)
00809     for (iy = 0; iy < met->ny; iy++) {
00810
00811       /* Get surface height... */
00812       lon = met->lon[ix];
00813       if (lon < topo_lon[0])
00814         lon += 360;
00815       else if (lon > topo_lon[topo_nx - 1])
00816         lon -= 360;
00817       lat = met->lat[iy];
00818       tx = locate_reg(topo_lon, topo_nx, lon);
00819       ty = locate_reg(topo_lat, topo_ny, lat);
00820       z0 = LIN(topo_lon[tx], topo_z[tx][ty],
00821                topo_lon[tx + 1], topo_z[tx + 1][ty], lon);
00822       z1 = LIN(topo_lon[tx], topo_z[tx][ty + 1],
00823                topo_lon[tx + 1], topo_z[tx + 1][ty + 1], lon);
00824       z0 = LIN(topo_lat[ty], z0, topo_lat[ty + 1], z1, lat);
00825
00826       /* Find surface pressure level... */
00827       ip0 = locate_irr(met->p, met->np, met->ps[ix][iy]);
00828
00829       /* Get surface temperature... */
00830       ts = LIN(met->p[ip0], met->t[ix][iy][ip0],
00831                met->p[ip0 + 1], met->t[ix][iy][ip0 + 1], met->ps[ix][iy]);
00832
00833       /* Upper part of profile... */
00834       met->z[ix][iy][ip0 + 1]
00835         = (float) (z0 + 8.31441 / 28.9647 / G0
00836                    * 0.5 * (ts + met->t[ix][iy][ip0 + 1])
00837                    * log(met->ps[ix][iy] / met->p[ip0 + 1]));
00838       for (ip = ip0 + 2; ip < met->np; ip++)
00839         met->z[ix][iy][ip]
00840           = (float) (met->z[ix][iy][ip - 1] + 8.31441 / 28.9647 / G0
00841                      * 0.5 * (met->t[ix][iy][ip - 1] + met->t[ix][iy][ip])
00842                      * log(met->p[ip - 1] / met->p[ip]));
00843     }
00844
00845   /* Smooth fields... */
00846   for (ip = 0; ip < met->np; ip++) {
00847
00848     /* Median filter... */
00849     for (ix = 0; ix < met->nx; ix++)
00850       for (iy = 0; iy < met->nx; iy++) {
00851         n = 0;
00852         for (ix2 = ix - 2; ix2 <= ix + 2; ix2++) {
00853           ix3 = ix2;
00854           if (ix3 < 0)
00855             ix3 += met->nx;
00856           if (ix3 >= met->nx)
00857             ix3 -= met->nx;
00858           for (iy2 = GSL_MAX(iy - 2, 0); iy2 <= GSL_MIN(iy + 2, met->ny - 1);
00859                iy2++)
00860             if (gsl_finite(met->z[ix3][iy2][ip])) {
00861               data[n] = met->z[ix3][iy2][ip];
00862               n++;
00863             }
00864         }
00865         if (n > 0) {
00866           gsl_sort(data, 1, (size_t) n);
00867           help[ix][iy] = (float)
00868             gsl_stats_median_from_sorted_data(data, 1, (size_t) n);
00869         } else
00870           help[ix][iy] = GSL_NAN;
00871       }
00872
00873     /* Copy data... */
00874     for (ix = 0; ix < met->nx; ix++)
00875       for (iy = 0; iy < met->nx; iy++)
00876         met->z[ix][iy][ip] = help[ix][iy];
00877   }
00878 }
```

Here is the call graph for this function:



5.3.1.11 void read_met_help (int *ncid*, char * *varname*, char * *varname2*, met_t * *met*, float *dest[EX][EY][EP]*, float *scl*)

Read and convert variable from meteorological data file.

Definition at line 882 of file extract.c.

```
00888             {
00889
00890   static float help[EX * EY * EP];
00891
00892   int ip, ix, iy, varid;
00893
00894   /* Check if variable exists... */
00895   if (nc_inq_varid(ncid, varname, &varid) != NC_NOERR)
00896     if (nc_inq_varid(ncid, varname2, &varid) != NC_NOERR)
00897       return;
00898
00899   /* Read data... */
00900   NC(nc_get_var_float(ncid, varid, help));
00901
00902   /* Copy and check data... */
00903   for (ix = 0; ix < met->nx; ix++)
00904     for (iy = 0; iy < met->ny; iy++)
00905       for (ip = 0; ip < met->np; ip++) {
00906         dest[ix][iy][ip] = help[(ip * met->ny + iy) * met->nx + ix];
00907         if (fabsf(dest[ix][iy][ip]) < 1e14f)
00908           dest[ix][iy][ip] *= scl;
00909         else
00910           dest[ix][iy][ip] = GSL_NAN;
00911       }
00912 }
```

5.3.1.12 void read_met_periodic (met_t * *met*)

Create meteorological data with periodic boundary conditions.

Definition at line 916 of file extract.c.

```
00917               {
00918
00919   int ip, iy;
00920
00921   /* Check longitudes... */
00922   if (!(fabs(met->lon[met->nx - 1] - met->lon[0]
00923              + met->lon[1] - met->lon[0] - 360) < 0.01))
00924     return;
00925
00926   /* Increase longitude counter... */
00927   if ((++met->nx) > EX)
00928     ERRMSG("Cannot create periodic boundary conditions!");
00929
```

```
00930   /* Set longitude... */
00931   met->lon[met->nx - 1] = met->lon[met->nx - 2] + met->lon[1] - met->
      lon[0];
00932
00933   /* Loop over latitudes and pressure levels... */
00934   for (iy = 0; iy < met->ny; iy++)
00935     for (ip = 0; ip < met->np; ip++) {
00936       met->ps[met->nx - 1][iy] = met->ps[0][iy];
00937       met->pt[met->nx - 1][iy] = met->pt[0][iy];
00938       met->z[met->nx - 1][iy][ip] = met->z[0][iy][ip];
00939       met->t[met->nx - 1][iy][ip] = met->t[0][iy][ip];
00940       met->u[met->nx - 1][iy][ip] = met->u[0][iy][ip];
00941       met->v[met->nx - 1][iy][ip] = met->v[0][iy][ip];
00942       met->w[met->nx - 1][iy][ip] = met->w[0][iy][ip];
00943       met->pv[met->nx - 1][iy][ip] = met->pv[0][iy][ip];
00944       met->h2o[met->nx - 1][iy][ip] = met->h2o[0][iy][ip];
00945       met->o3[met->nx - 1][iy][ip] = met->o3[0][iy][ip];
00946     }
00947 }
```

5.3.1.13 void read_met_sample (ctl2_t * *ctl2*, met_t * *met*)

Downsampling of meteorological data.

Definition at line 951 of file extract.c.

```
00953                 {
00954
00955   met_t *help;
00956
00957   float w, wsum;
00958
00959   int ip, ip2, ix, ix2, ix3, iy, iy2;
00960
00961   /* Check parameters... */
00962   if (ctl2->met_dp <= 1 && ctl2->met_dx <= 1 && ctl2->met_dy <= 1
00963       && ctl2->met_sp <= 1 && ctl2->met_sx <= 1 && ctl2->met_sy <= 1)
00964     return;
00965
00966   /* Allocate... */
00967   ALLOC(help, met_t, 1);
00968
00969   /* Copy data... */
00970   help->nx = met->nx;
00971   help->ny = met->ny;
00972   help->np = met->np;
00973   memcpy(help->lon, met->lon, sizeof(met->lon));
00974   memcpy(help->lat, met->lat, sizeof(met->lat));
00975   memcpy(help->p, met->p, sizeof(met->p));
00976
00977   /* Smoothing... */
00978   for (ix = 0; ix < met->nx; ix += ctl2->met_dx) {
00979     for (iy = 0; iy < met->ny; iy += ctl2->met_dy) {
00980       for (ip = 0; ip < met->np; ip += ctl2->met_dp) {
00981         help->ps[ix][iy] = 0;
00982         help->pt[ix][iy] = 0;
00983         help->z[ix][iy][ip] = 0;
00984         help->t[ix][iy][ip] = 0;
00985         help->u[ix][iy][ip] = 0;
00986         help->v[ix][iy][ip] = 0;
00987         help->w[ix][iy][ip] = 0;
00988         help->pv[ix][iy][ip] = 0;
00989         help->h2o[ix][iy][ip] = 0;
00990         help->o3[ix][iy][ip] = 0;
00991         wsum = 0;
00992         for (ix2 = ix - ctl2->met_sx + 1; ix2 <= ix + ctl2->met_sx - 1; ix2++) {
00993           ix3 = ix2;
00994           if (ix3 < 0)
00995             ix3 += met->nx;
00996           else if (ix3 >= met->nx)
00997             ix3 -= met->nx;
00998
00999           for (iy2 = GSL_MAX(iy - ctl2->met_sy + 1, 0);
01000                iy2 <= GSL_MIN(iy + ctl2->met_sy - 1, met->ny - 1); iy2++)
01001             for (ip2 = GSL_MAX(ip - ctl2->met_sp + 1, 0);
01002                  ip2 <= GSL_MIN(ip + ctl2->met_sp - 1, met->np - 1); ip2++) {
01003               w = (float) (1.0 - fabs(ix - ix2) / ctl2->met_sx)
01004                 * (float) (1.0 - fabs(iy - iy2) / ctl2->met_sy)
01005                 * (float) (1.0 - fabs(ip - ip2) / ctl2->met_sp);
01006               help->ps[ix][iy] += w * met->ps[ix3][iy2];
```

```
01007               help->pt[ix][iy] += w * met->pt[ix3][iy2];
01008               help->z[ix][iy][ip] += w * met->z[ix3][iy2][ip2];
01009               help->t[ix][iy][ip] += w * met->t[ix3][iy2][ip2];
01010               help->u[ix][iy][ip] += w * met->u[ix3][iy2][ip2];
01011               help->v[ix][iy][ip] += w * met->v[ix3][iy2][ip2];
01012               help->w[ix][iy][ip] += w * met->w[ix3][iy2][ip2];
01013               help->pv[ix][iy][ip] += w * met->pv[ix3][iy2][ip2];
01014               help->h2o[ix][iy][ip] += w * met->h2o[ix3][iy2][ip2];
01015               help->o3[ix][iy][ip] += w * met->o3[ix3][iy2][ip2];
01016               wsum += w;
01017             }
01018         }
01019         help->ps[ix][iy] /= wsum;
01020         help->pt[ix][iy] /= wsum;
01021         help->t[ix][iy][ip] /= wsum;
01022         help->z[ix][iy][ip] /= wsum;
01023         help->u[ix][iy][ip] /= wsum;
01024         help->v[ix][iy][ip] /= wsum;
01025         help->w[ix][iy][ip] /= wsum;
01026         help->pv[ix][iy][ip] /= wsum;
01027         help->h2o[ix][iy][ip] /= wsum;
01028         help->o3[ix][iy][ip] /= wsum;
01029       }
01030     }
01031   }
01032
01033   /* Downsampling... */
01034   met->nx = 0;
01035   for (ix = 0; ix < help->nx; ix += ctl2->met_dx) {
01036     met->lon[met->nx] = help->lon[ix];
01037     met->ny = 0;
01038     for (iy = 0; iy < help->ny; iy += ctl2->met_dy) {
01039       met->lat[met->ny] = help->lat[iy];
01040       met->ps[met->nx][met->ny] = help->ps[ix][iy];
01041       met->pt[met->nx][met->ny] = help->pt[ix][iy];
01042       met->np = 0;
01043       for (ip = 0; ip < help->np; ip += ctl2->met_dp) {
01044         met->p[met->np] = help->p[ip];
01045         met->z[met->nx][met->ny][met->np] = help->z[ix][iy][ip];
01046         met->t[met->nx][met->ny][met->np] = help->t[ix][iy][ip];
01047         met->u[met->nx][met->ny][met->np] = help->u[ix][iy][ip];
01048         met->v[met->nx][met->ny][met->np] = help->v[ix][iy][ip];
01049         met->w[met->nx][met->ny][met->np] = help->w[ix][iy][ip];
01050         met->pv[met->nx][met->ny][met->np] = help->pv[ix][iy][ip];
01051         met->h2o[met->nx][met->ny][met->np] = help->h2o[ix][iy][ip];
01052         met->o3[met->nx][met->ny][met->np] = help->o3[ix][iy][ip];
01053         met->np++;
01054       }
01055       met->ny++;
01056     }
01057     met->nx++;
01058   }
01059
01060   /* Free... */
01061   free(help);
01062 }
```

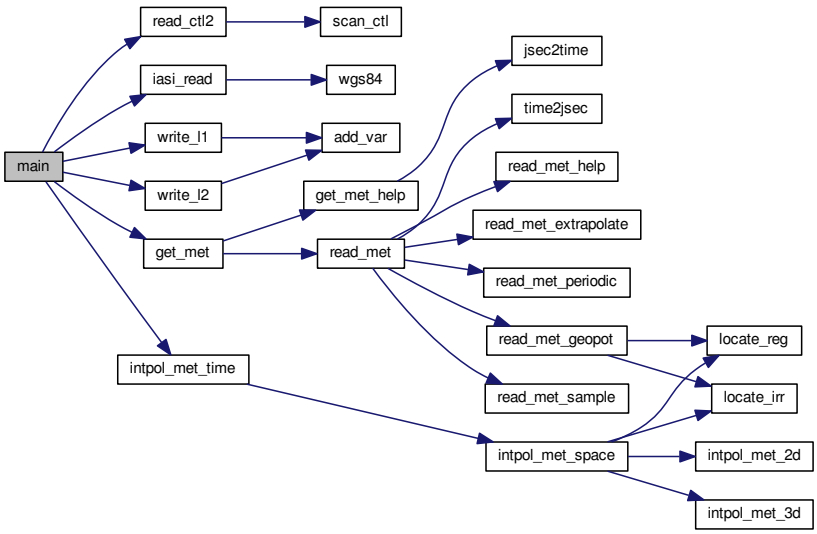
5.3.1.14 int main (int *argc*, char * *argv[]*)

Definition at line 238 of file extract.c.

```
00240               {
00241
00242   static iasi_rad_t *iasi_rad;
00243
00244   static iasi_l1_t l1;
00245   static iasi_l2_t l2;
00246
00247   static ctl2_t ctl2;
00248
00249   met_t *met0, *met1;
00250
00251   double ts;
00252
00253   int ichan, lay, track = 0, xtrack;
00254
00255   /* Check arguments... */
00256   if (argc < 4)
00257     ERRMSG("Give parameters: <ctl> <iasi_l1_file> <metbase> <out.nc>");
00258
00259   /* Allocate... */
```

```
00260   ALLOC(iasi_rad, iasi_rad_t, 1);
00261   ALLOC(met0, met_t, 1);
00262   ALLOC(met1, met_t, 1);
00263
00264   /* Read control parameters... */
00265   read_ctl2(argc, argv, &ctl2);
00266
00267   /* Read IASI data... */
00268   iasi_read(argv[2], iasi_rad);
00269
00270   /* Copy data to struct... */
00271   l1.ntrack = (size_t) iasi_rad->ntrack;
00272   for (track = 0; track < iasi_rad->ntrack; track++)
00273     for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00274       l1.time[track][xtrack]
00275         = iasi_rad->Time[track][xtrack];
00276       l1.lon[track][xtrack]
00277         = iasi_rad->Longitude[track][xtrack];
00278       l1.lat[track][xtrack]
00279         = iasi_rad->Latitude[track][xtrack];
00280       l1.sat_z[track]
00281         = iasi_rad->Sat_z[track];
00282       l1.sat_lon[track]
00283         = iasi_rad->Sat_lon[track];
00284       l1.sat_lat[track]
00285         = iasi_rad->Sat_lat[track];
00286       for (ichan = 0; ichan < L1_NCHAN; ichan++) {
00287         l1.nu[ichan]
00288           = iasi_rad->freq[iasi_chan[ichan]];
00289         l1.rad[track][xtrack][ichan]
00290           = iasi_rad->Rad[track][xtrack][iasi_chan[ichan]];
00291       }
00292     }
00293
00294   /* Write netCDF file... */
00295   write_l1(argv[4], &l1);
00296
00297   /* Read meteo data... */
00298   l2.ntrack = l1.ntrack;
00299   ts = l1.time[l2.ntrack / 2][L1_NXTRACK / 2];
00300   get_met(&ctl2, argv[3], ts, met0, met1);
00301
00302   /* Interpolate meteo data... */
00303   for (track = 0; track < (int) l2.ntrack; track++)
00304     for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++)
00305       for (lay = 0; lay < L2_NLAY; lay++) {
00306         l2.time[track][xtrack] = l1.time[track][xtrack];
00307         l2.lon[track][xtrack] = l1.lon[track][xtrack];
00308         l2.lat[track][xtrack] = l1.lat[track][xtrack];
00309         l2.p[lay] = 1013.25 * exp(-2.5 * lay / 7.0);
00310         intpol_met_time(met0, met1, ts, l2.p[lay],
00311                         l2.lon[track][xtrack], l2.lat[track][xtrack],
00312                         NULL, NULL, &l2.z[track][xtrack][lay],
00313                         &l2.t[track][xtrack][lay],
00314                         NULL, NULL, NULL, NULL, NULL, NULL);
00315       }
00316
00317   /* Write netCDF file... */
00318   write_l2(argv[4], &l2);
00319
00320   /* Free... */
00321   free(iasi_rad);
00322   free(met0);
00323   free(met1);
00324
00325   return EXIT_SUCCESS;
00326 }
```

Here is the call graph for this function:



5.3.2 Variable Documentation

5.3.2.1 int iasi_chan[L1_NCHAN]

Initial value:

```
= { 71, 88, 89, 90, 91, 92, 93, 94, 95, 96,
  97, 98, 99, 6712, 6720, 6735, 6742, 6749, 6750,
  6756, 6757, 6763, 6764, 6770, 6771, 6777, 6778,
  6784, 6791, 6797, 6838, 6855, 6866
}
```

List of IASI channels (don't change).

Definition at line 21 of file extract.c.

5.4 extract.c

```
00001 #include "libiasi.h"
00002
00003 /* ------------------------------------------------------------
00004    Dimensions...
00005    ------------------------------------------------------------ */
00006
00008 #define EP 72
00009
00011 #define EX 362
00012
00014 #define EY 182
00015
00016 /* ------------------------------------------------------------
00017    Global variables...
00018    ------------------------------------------------------------ */
00019
00021 int iasi_chan[L1_NCHAN] = { 71, 88, 89, 90, 91, 92, 93, 94, 95, 96,
00022   97, 98, 99, 6712, 6720, 6735, 6742, 6749, 6750,
00023   6756, 6757, 6763, 6764, 6770, 6771, 6777, 6778,
```

```
00024   6784, 6791, 6797, 6838, 6855, 6866
00025 };
00026
00027 /* ------------------------------------------------------------
00028    Structs...
00029    ------------------------------------------------------------ */
00030
00032 typedef struct {
00033
00035   double dt_met;
00036
00038   char met_geopot[LEN];
00039
00041   int met_dx;
00042
00044   int met_dy;
00045
00047   int met_dp;
00048
00050   int met_sx;
00051
00053   int met_sy;
00054
00056   int met_sp;
00057
00058 } ctl2_t;
00059
00061 typedef struct {
00062
00064   double time;
00065
00067   int nx;
00068
00070   int ny;
00071
00073   int np;
00074
00076   double lon[EX];
00077
00079   double lat[EY];
00080
00082   double p[EP];
00083
00085   double ps[EX][EY];
00086
00088   double pt[EX][EY];
00089
00091   float z[EX][EY][EP];
00092
00094   float t[EX][EY][EP];
00095
00097   float u[EX][EY][EP];
00098
00100   float v[EX][EY][EP];
00101
00103   float w[EX][EY][EP];
00104
00106   float pv[EX][EY][EP];
00107
00109   float h2o[EX][EY][EP];
00110
00112   float o3[EX][EY][EP];
00113
00115   float pl[EX][EY][EP];
00116
00117 } met_t;
00118
00119 /* ------------------------------------------------------------
00120    Functions...
00121    ------------------------------------------------------------ */
00122
00124 void get_met(
00125   ctl2_t * ctl2,
00126   char *metbase,
00127   double t,
00128   met_t * met0,
00129   met_t * met1);
00130
00132 void get_met_help(
00133   double t,
00134   int direct,
00135   char *metbase,
00136   double dt_met,
00137   char *filename);
00138
00140 void intpol_met_2d(
00141   double array[EX][EY],
```

```
00142   int ix,
00143   int iy,
00144   double wx,
00145   double wy,
00146   double *var);
00147
00149 void intpol_met_3d(
00150   float array[EX][EY][EP],
00151   int ip,
00152   int ix,
00153   int iy,
00154   double wp,
00155   double wx,
00156   double wy,
00157   double *var);
00158
00160 void intpol_met_space(
00161   met_t * met,
00162   double p,
00163   double lon,
00164   double lat,
00165   double *ps,
00166   double *pt,
00167   double *z,
00168   double *t,
00169   double *u,
00170   double *v,
00171   double *w,
00172   double *pv,
00173   double *h2o,
00174   double *o3);
00175
00177 void intpol_met_time(
00178   met_t * met0,
00179   met_t * met1,
00180   double ts,
00181   double p,
00182   double lon,
00183   double lat,
00184   double *ps,
00185   double *pt,
00186   double *z,
00187   double *t,
00188   double *u,
00189   double *v,
00190   double *w,
00191   double *pv,
00192   double *h2o,
00193   double *o3);
00194
00196 void read_ctl2(
00197   int argc,
00198   char *argv[],
00199   ctl2_t * ctl2);
00200
00202 void read_met(
00203   ctl2_t * ctl2,
00204   char *filename,
00205   met_t * met);
00206
00208 void read_met_extrapolate(
00209   met_t * met);
00210
00212 void read_met_geopot(
00213   ctl2_t * ctl2,
00214   met_t * met);
00215
00217 void read_met_help(
00218   int ncid,
00219   char *varname,
00220   char *varname2,
00221   met_t * met,
00222   float dest[EX][EY][EP],
00223   float scl);
00224
00226 void read_met_periodic(
00227   met_t * met);
00228
00230 void read_met_sample(
00231   ctl2_t * ctl2,
00232   met_t * met);
00233
00234 /* ------------------------------------------------------------
00235    Main...
00236    ------------------------------------------------------------ */
00237
00238 int main(
```

```
00239   int argc,
00240   char *argv[]) {
00241
00242   static iasi_rad_t *iasi_rad;
00243
00244   static iasi_l1_t l1;
00245   static iasi_l2_t l2;
00246
00247   static ctl2_t ctl2;
00248
00249   met_t *met0, *met1;
00250
00251   double ts;
00252
00253   int ichan, lay, track = 0, xtrack;
00254
00255   /* Check arguments... */
00256   if (argc < 4)
00257     ERRMSG("Give parameters: <ctl> <iasi_l1_file> <metbase> <out.nc>");
00258
00259   /* Allocate... */
00260   ALLOC(iasi_rad, iasi_rad_t, 1);
00261   ALLOC(met0, met_t, 1);
00262   ALLOC(met1, met_t, 1);
00263
00264   /* Read control parameters... */
00265   read_ctl2(argc, argv, &ctl2);
00266
00267   /* Read IASI data... */
00268   iasi_read(argv[2], iasi_rad);
00269
00270   /* Copy data to struct... */
00271   l1.ntrack = (size_t) iasi_rad->ntrack;
00272   for (track = 0; track < iasi_rad->ntrack; track++)
00273     for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00274       l1.time[track][xtrack]
00275         = iasi_rad->Time[track][xtrack];
00276       l1.lon[track][xtrack]
00277         = iasi_rad->Longitude[track][xtrack];
00278       l1.lat[track][xtrack]
00279         = iasi_rad->Latitude[track][xtrack];
00280       l1.sat_z[track]
00281         = iasi_rad->Sat_z[track];
00282       l1.sat_lon[track]
00283         = iasi_rad->Sat_lon[track];
00284       l1.sat_lat[track]
00285         = iasi_rad->Sat_lat[track];
00286       for (ichan = 0; ichan < L1_NCHAN; ichan++) {
00287         l1.nu[ichan]
00288           = iasi_rad->freq[iasi_chan[ichan]];
00289         l1.rad[track][xtrack][ichan]
00290           = iasi_rad->Rad[track][xtrack][iasi_chan[ichan]];
00291       }
00292     }
00293
00294   /* Write netCDF file... */
00295   write_l1(argv[4], &l1);
00296
00297   /* Read meteo data... */
00298   l2.ntrack = l1.ntrack;
00299   ts = l1.time[l2.ntrack / 2][L1_NXTRACK / 2];
00300   get_met(&ctl2, argv[3], ts, met0, met1);
00301
00302   /* Interpolate meteo data... */
00303   for (track = 0; track < (int) l2.ntrack; track++)
00304     for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++)
00305       for (lay = 0; lay < L2_NLAY; lay++) {
00306         l2.time[track][xtrack] = l1.time[track][xtrack];
00307         l2.lon[track][xtrack] = l1.lon[track][xtrack];
00308         l2.lat[track][xtrack] = l1.lat[track][xtrack];
00309         l2.p[lay] = 1013.25 * exp(-2.5 * lay / 7.0);
00310         intpol_met_time(met0, met1, ts, l2.p[lay],
00311                         l2.lon[track][xtrack], l2.lat[track][xtrack],
00312                         NULL, NULL, &l2.z[track][xtrack][lay],
00313                         &l2.t[track][xtrack][lay],
00314                         NULL, NULL, NULL, NULL, NULL, NULL);
00315       }
00316
00317   /* Write netCDF file... */
00318   write_l2(argv[4], &l2);
00319
00320   /* Free... */
00321   free(iasi_rad);
00322   free(met0);
00323   free(met1);
00324
00325   return EXIT_SUCCESS;
```

```
00326 }
00327
00328 /*****************************************************************************/
00329
00330 void get_met(
00331   ctl2_t * ctl2,
00332   char *metbase,
00333   double t,
00334   met_t * met0,
00335   met_t * met1) {
00336
00337   char filename[LEN];
00338
00339   static int init;
00340
00341   /* Init... */
00342   if (!init) {
00343     init = 1;
00344
00345     get_met_help(t, -1, metbase, ctl2->dt_met, filename);
00346     read_met(ctl2, filename, met0);
00347
00348     get_met_help(t + 1.0, 1, metbase, ctl2->dt_met, filename);
00349     read_met(ctl2, filename, met1);
00350   }
00351
00352   /* Read new data for forward trajectories... */
00353   if (t > met1->time) {
00354     memcpy(met0, met1, sizeof(met_t));
00355     get_met_help(t, 1, metbase, ctl2->dt_met, filename);
00356     read_met(ctl2, filename, met1);
00357   }
00358 }
00359
00360 /*****************************************************************************/
00361
00362 void get_met_help(
00363   double t,
00364   int direct,
00365   char *metbase,
00366   double dt_met,
00367   char *filename) {
00368
00369   double t6, r;
00370
00371   int year, mon, day, hour, min, sec;
00372
00373   /* Round time to fixed intervals... */
00374   if (direct == -1)
00375     t6 = floor(t / dt_met) * dt_met;
00376   else
00377     t6 = ceil(t / dt_met) * dt_met;
00378
00379   /* Decode time... */
00380   jsec2time(t6, &year, &mon, &day, &hour, &min, &sec, &r);
00381
00382   /* Set filename... */
00383   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", metbase, year, mon, day, hour);
00384 }
00385
00386 /*****************************************************************************/
00387
00388 void intpol_met_2d(
00389   double array[EX][EY],
00390   int ix,
00391   int iy,
00392   double wx,
00393   double wy,
00394   double *var) {
00395
00396   double aux00, aux01, aux10, aux11;
00397
00398   /* Set variables... */
00399   aux00 = array[ix][iy];
00400   aux01 = array[ix][iy + 1];
00401   aux10 = array[ix + 1][iy];
00402   aux11 = array[ix + 1][iy + 1];
00403
00404   /* Interpolate horizontally... */
00405   aux00 = wy * (aux00 - aux01) + aux01;
00406   aux11 = wy * (aux10 - aux11) + aux11;
00407   *var = wx * (aux00 - aux11) + aux11;
00408 }
00409
00410 /*****************************************************************************/
00411
00412 void intpol_met_3d(
```

```
00413   float array[EX][EY][EP],
00414   int ip,
00415   int ix,
00416   int iy,
00417   double wp,
00418   double wx,
00419   double wy,
00420   double *var) {
00421
00422   double aux00, aux01, aux10, aux11;
00423
00424   /* Interpolate vertically... */
00425   aux00 = wp * (array[ix][iy][ip] - array[ix][iy][ip + 1])
00426     + array[ix][iy][ip + 1];
00427   aux01 = wp * (array[ix][iy + 1][ip] - array[ix][iy + 1][ip + 1])
00428     + array[ix][iy + 1][ip + 1];
00429   aux10 = wp * (array[ix + 1][iy][ip] - array[ix + 1][iy][ip + 1])
00430     + array[ix + 1][iy][ip + 1];
00431   aux11 = wp * (array[ix + 1][iy + 1][ip] - array[ix + 1][iy + 1][ip + 1])
00432     + array[ix + 1][iy + 1][ip + 1];
00433
00434   /* Interpolate horizontally... */
00435   aux00 = wy * (aux00 - aux01) + aux01;
00436   aux11 = wy * (aux10 - aux11) + aux11;
00437   *var = wx * (aux00 - aux11) + aux11;
00438 }
00439
00440 /*****************************************************************************/
00441
00442 void intpol_met_space(
00443   met_t * met,
00444   double p,
00445   double lon,
00446   double lat,
00447   double *ps,
00448   double *pt,
00449   double *z,
00450   double *t,
00451   double *u,
00452   double *v,
00453   double *w,
00454   double *pv,
00455   double *h2o,
00456   double *o3) {
00457
00458   double wp, wx, wy;
00459
00460   int ip, ix, iy;
00461
00462   /* Check longitude... */
00463   if (met->lon[met->nx - 1] > 180 && lon < 0)
00464     lon += 360;
00465
00466   /* Get indices... */
00467   ip = locate_irr(met->p, met->np, p);
00468   ix = locate_reg(met->lon, met->nx, lon);
00469   iy = locate_reg(met->lat, met->ny, lat);
00470
00471   /* Get weights... */
00472   wp = (met->p[ip + 1] - p) / (met->p[ip + 1] - met->p[ip]);
00473   wx = (met->lon[ix + 1] - lon) / (met->lon[ix + 1] - met->lon[ix]);
00474   wy = (met->lat[iy + 1] - lat) / (met->lat[iy + 1] - met->lat[iy]);
00475
00476   /* Interpolate... */
00477   if (ps != NULL)
00478     intpol_met_2d(met->ps, ix, iy, wx, wy, ps);
00479   if (pt != NULL)
00480     intpol_met_2d(met->pt, ix, iy, wx, wy, pt);
00481   if (z != NULL)
00482     intpol_met_3d(met->z, ip, ix, iy, wp, wx, wy, z);
00483   if (t != NULL)
00484     intpol_met_3d(met->t, ip, ix, iy, wp, wx, wy, t);
00485   if (u != NULL)
00486     intpol_met_3d(met->u, ip, ix, iy, wp, wx, wy, u);
00487   if (v != NULL)
00488     intpol_met_3d(met->v, ip, ix, iy, wp, wx, wy, v);
00489   if (w != NULL)
00490     intpol_met_3d(met->w, ip, ix, iy, wp, wx, wy, w);
00491   if (pv != NULL)
00492     intpol_met_3d(met->pv, ip, ix, iy, wp, wx, wy, pv);
00493   if (h2o != NULL)
00494     intpol_met_3d(met->h2o, ip, ix, iy, wp, wx, wy, h2o);
00495   if (o3 != NULL)
00496     intpol_met_3d(met->o3, ip, ix, iy, wp, wx, wy, o3);
00497 }
00498
00499 /*****************************************************************************/
```

```
00500
00501 void intpol_met_time(
00502   met_t * met0,
00503   met_t * met1,
00504   double ts,
00505   double p,
00506   double lon,
00507   double lat,
00508   double *ps,
00509   double *pt,
00510   double *z,
00511   double *t,
00512   double *u,
00513   double *v,
00514   double *w,
00515   double *pv,
00516   double *h2o,
00517   double *o3) {
00518
00519   double h2o0, h2o1, o30, o31, ps0, ps1, pt0, pt1, pv0, pv1, t0, t1, u0, u1,
00520     v0, v1, w0, w1, wt, z0, z1;
00521
00522   /* Spatial interpolation... */
00523   intpol_met_space(met0, p, lon, lat,
00524                    ps == NULL ? NULL : &ps0,
00525                    pt == NULL ? NULL : &pt0,
00526                    z == NULL ? NULL : &z0,
00527                    t == NULL ? NULL : &t0,
00528                    u == NULL ? NULL : &u0,
00529                    v == NULL ? NULL : &v0,
00530                    w == NULL ? NULL : &w0,
00531                    pv == NULL ? NULL : &pv0,
00532                    h2o == NULL ? NULL : &h2o0, o3 == NULL ? NULL : &o30);
00533   intpol_met_space(met1, p, lon, lat,
00534                    ps == NULL ? NULL : &ps1,
00535                    pt == NULL ? NULL : &pt1,
00536                    z == NULL ? NULL : &z1,
00537                    t == NULL ? NULL : &t1,
00538                    u == NULL ? NULL : &u1,
00539                    v == NULL ? NULL : &v1,
00540                    w == NULL ? NULL : &w1,
00541                    pv == NULL ? NULL : &pv1,
00542                    h2o == NULL ? NULL : &h2o1, o3 == NULL ? NULL : &o31);
00543
00544   /* Get weighting factor... */
00545   wt = (met1->time - ts) / (met1->time - met0->time);
00546
00547   /* Interpolate... */
00548   if (ps != NULL)
00549     *ps = wt * (ps0 - ps1) + ps1;
00550   if (pt != NULL)
00551     *pt = wt * (pt0 - pt1) + pt1;
00552   if (z != NULL)
00553     *z = wt * (z0 - z1) + z1;
00554   if (t != NULL)
00555     *t = wt * (t0 - t1) + t1;
00556   if (u != NULL)
00557     *u = wt * (u0 - u1) + u1;
00558   if (v != NULL)
00559     *v = wt * (v0 - v1) + v1;
00560   if (w != NULL)
00561     *w = wt * (w0 - w1) + w1;
00562   if (pv != NULL)
00563     *pv = wt * (pv0 - pv1) + pv1;
00564   if (h2o != NULL)
00565     *h2o = wt * (h2o0 - h2o1) + h2o1;
00566   if (o3 != NULL)
00567     *o3 = wt * (o30 - o31) + o31;
00568 }
00569
00570 /*****************************************************************************/
00571
00572 void read_ctl2(
00573   int argc,
00574   char *argv[],
00575   ctl2_t * ctl2) {
00576
00577   /* Meteorological data... */
00578   ctl2->dt_met = scan_ctl(argc, argv, "DT_MET", -1, "21600", NULL);
00579   scan_ctl(argc, argv, "MET_GEOPOT", -1, "", ctl2->met_geopot);
00580   ctl2->met_dx = (int) scan_ctl(argc, argv, "MET_DX", -1, "1", NULL);
00581   ctl2->met_dy = (int) scan_ctl(argc, argv, "MET_DY", -1, "1", NULL);
00582   ctl2->met_dp = (int) scan_ctl(argc, argv, "MET_DP", -1, "1", NULL);
00583   ctl2->met_sx = (int) scan_ctl(argc, argv, "MET_SX", -1, "20", NULL);
00584   ctl2->met_sy = (int) scan_ctl(argc, argv, "MET_SY", -1, "10", NULL);
00585   ctl2->met_sp = (int) scan_ctl(argc, argv, "MET_SP", -1, "1", NULL);
00586 }
```

```
00587
00588 /*****************************************************************************/
00589
00590 void read_met(
00591   ctl2_t * ctl2,
00592   char *filename,
00593   met_t * met) {
00594
00595   char levname[LEN], tstr[10];
00596
00597   static float help[EX * EY];
00598
00599   int ix, iy, ip, dimid, ncid, varid, year, mon, day, hour;
00600
00601   size_t np, nx, ny;
00602
00603   /* Write info... */
00604   printf("Read meteorological data: %s\n", filename);
00605
00606   /* Get time from filename... */
00607   sprintf(tstr, "%.4s", &filename[strlen(filename) - 16]);
00608   year = atoi(tstr);
00609   sprintf(tstr, "%.2s", &filename[strlen(filename) - 11]);
00610   mon = atoi(tstr);
00611   sprintf(tstr, "%.2s", &filename[strlen(filename) - 8]);
00612   day = atoi(tstr);
00613   sprintf(tstr, "%.2s", &filename[strlen(filename) - 5]);
00614   hour = atoi(tstr);
00615   time2jsec(year, mon, day, hour, 0, 0, 0, &met->time);
00616
00617   /* Open netCDF file... */
00618   NC(nc_open(filename, NC_NOWRITE, &ncid));
00619
00620   /* Get dimensions... */
00621   NC(nc_inq_dimid(ncid, "lon", &dimid));
00622   NC(nc_inq_dimlen(ncid, dimid, &nx));
00623   if (nx < 2 || nx > EX)
00624     ERRMSG("Number of longitudes out of range!");
00625
00626   NC(nc_inq_dimid(ncid, "lat", &dimid));
00627   NC(nc_inq_dimlen(ncid, dimid, &ny));
00628   if (ny < 2 || ny > EY)
00629     ERRMSG("Number of latitudes out of range!");
00630
00631   sprintf(levname, "lev");
00632   NC(nc_inq_dimid(ncid, levname, &dimid));
00633   NC(nc_inq_dimlen(ncid, dimid, &np));
00634   if (np == 1) {
00635     sprintf(levname, "lev_2");
00636     NC(nc_inq_dimid(ncid, levname, &dimid));
00637     NC(nc_inq_dimlen(ncid, dimid, &np));
00638   }
00639   if (np < 2 || np > EP)
00640     ERRMSG("Number of levels out of range!");
00641
00642   /* Store dimensions... */
00643   met->np = (int) np;
00644   met->nx = (int) nx;
00645   met->ny = (int) ny;
00646
00647   /* Get horizontal grid... */
00648   NC(nc_inq_varid(ncid, "lon", &varid));
00649   NC(nc_get_var_double(ncid, varid, met->lon));
00650   NC(nc_inq_varid(ncid, "lat", &varid));
00651   NC(nc_get_var_double(ncid, varid, met->lat));
00652
00653   /* Read meteorological data... */
00654   read_met_help(ncid, "t", "T", met, met->t, 1.0);
00655   read_met_help(ncid, "u", "U", met, met->u, 1.0);
00656   read_met_help(ncid, "v", "V", met, met->v, 1.0);
00657   read_met_help(ncid, "w", "W", met, met->w, 0.01f);
00658   read_met_help(ncid, "q", "Q", met, met->h2o, 1.608f);
00659   read_met_help(ncid, "o3", "O3", met, met->o3, 0.602f);
00660
00661   /* Read pressure levels from file... */
00662   NC(nc_inq_varid(ncid, levname, &varid));
00663   NC(nc_get_var_double(ncid, varid, met->p));
00664   for (ip = 0; ip < met->np; ip++)
00665     met->p[ip] /= 100.;
00666
00667   /* Extrapolate data for lower boundary... */
00668   read_met_extrapolate(met);
00669
00670   /* Check ordering of pressure levels... */
00671   for (ip = 1; ip < met->np; ip++)
00672     if (met->p[ip - 1] < met->p[ip])
00673       ERRMSG("Pressure levels must be descending!");
```

```
00674
00675   /* Read surface pressure... */
00676   if (nc_inq_varid(ncid, "ps", &varid) == NC_NOERR
00677       || nc_inq_varid(ncid, "PS", &varid) == NC_NOERR) {
00678     NC(nc_get_var_float(ncid, varid, help));
00679     for (iy = 0; iy < met->ny; iy++)
00680       for (ix = 0; ix < met->nx; ix++)
00681         met->ps[ix][iy] = help[iy * met->nx + ix] / 100.;
00682   } else if (nc_inq_varid(ncid, "lnsp", &varid) == NC_NOERR
00683              || nc_inq_varid(ncid, "LNSP", &varid) == NC_NOERR) {
00684     NC(nc_get_var_float(ncid, varid, help));
00685     for (iy = 0; iy < met->ny; iy++)
00686       for (ix = 0; ix < met->nx; ix++)
00687         met->ps[ix][iy] = exp(help[iy * met->nx + ix]) / 100.;
00688   } else
00689     for (ix = 0; ix < met->nx; ix++)
00690       for (iy = 0; iy < met->ny; iy++)
00691         met->ps[ix][iy] = met->p[0];
00692
00693   /* Create periodic boundary conditions... */
00694   read_met_periodic(met);
00695
00696   /* Calculate geopotential heights... */
00697   read_met_geopot(ctl2, met);
00698
00699   /* Downsampling... */
00700   read_met_sample(ctl2, met);
00701
00702   /* Close file... */
00703   NC(nc_close(ncid));
00704 }
00705
00706 /*****************************************************************************/
00707
00708 void read_met_extrapolate(
00709   met_t * met) {
00710
00711   int ip, ip0, ix, iy;
00712
00713   /* Loop over columns... */
00714   for (ix = 0; ix < met->nx; ix++)
00715     for (iy = 0; iy < met->ny; iy++) {
00716
00717       /* Find lowest valid data point... */
00718       for (ip0 = met->np - 1; ip0 >= 0; ip0--)
00719         if (!gsl_finite(met->t[ix][iy][ip0])
00720             || !gsl_finite(met->u[ix][iy][ip0])
00721             || !gsl_finite(met->v[ix][iy][ip0])
00722             || !gsl_finite(met->w[ix][iy][ip0]))
00723           break;
00724
00725       /* Extrapolate... */
00726       for (ip = ip0; ip >= 0; ip--) {
00727         met->t[ix][iy][ip] = met->t[ix][iy][ip + 1];
00728         met->u[ix][iy][ip] = met->u[ix][iy][ip + 1];
00729         met->v[ix][iy][ip] = met->v[ix][iy][ip + 1];
00730         met->w[ix][iy][ip] = met->w[ix][iy][ip + 1];
00731         met->h2o[ix][iy][ip] = met->h2o[ix][iy][ip + 1];
00732         met->o3[ix][iy][ip] = met->o3[ix][iy][ip + 1];
00733       }
00734     }
00735 }
00736
00737 /*****************************************************************************/
00738
00739 void read_met_geopot(
00740   ctl2_t * ctl2,
00741   met_t * met) {
00742
00743   static double topo_lat[EY], topo_lon[EX], topo_z[EX][EY];
00744
00745   static int init, topo_nx = -1, topo_ny;
00746
00747   FILE *in;
00748
00749   char line[LEN];
00750
00751   double data[30], lat, lon, rlat, rlon, rlon_old = -999, rz, ts, z0, z1;
00752
00753   float help[EX][EY];
00754
00755   int ip, ip0, ix, ix2, ix3, iy, iy2, n, tx, ty;
00756
00757   /* Initialize geopotential heights... */
00758   for (ix = 0; ix < met->nx; ix++)
00759     for (iy = 0; iy < met->ny; iy++)
00760       for (ip = 0; ip < met->np; ip++)
```

```
00761         met->z[ix][iy][ip] = GSL_NAN;
00762
00763     /* Check filename... */
00764     if (ctl2->met_geopot[0] == '-')
00765       return;
00766
00767     /* Read surface geopotential... */
00768     if (!init) {
00769
00770       /* Write info... */
00771       printf("Read surface geopotential: %s\n", ctl2->met_geopot);
00772
00773       /* Open file... */
00774       if (!(in = fopen(ctl2->met_geopot, "r")))
00775         ERRMSG("Cannot open file!");
00776
00777       /* Read data... */
00778       while (fgets(line, LEN, in))
00779         if (sscanf(line, "%lg %lg %lg", &rlon, &rlat, &rz) == 3) {
00780           if (rlon != rlon_old) {
00781             if ((++topo_nx) >= EX)
00782               ERRMSG("Too many longitudes!");
00783             topo_ny = 0;
00784           }
00785           rlon_old = rlon;
00786           topo_lon[topo_nx] = rlon;
00787           topo_lat[topo_ny] = rlat;
00788           topo_z[topo_nx][topo_ny] = rz;
00789           if ((++topo_ny) >= EY)
00790             ERRMSG("Too many latitudes!");
00791         }
00792       if ((++topo_nx) >= EX)
00793         ERRMSG("Too many longitudes!");
00794
00795       /* Close file... */
00796       fclose(in);
00797
00798       /* Check grid spacing... */
00799       if (fabs(met->lon[0] - met->lon[1]) != fabs(topo_lon[0] - topo_lon[1])
00800           || fabs(met->lat[0] - met->lat[1]) != fabs(topo_lat[0] - topo_lat[1]))
00801         printf("Warning: Grid spacing does not match!\n");
00802
00803       /* Set init flag... */
00804       init = 1;
00805     }
00806
00807     /* Apply hydrostatic equation to calculate geopotential heights... */
00808     for (ix = 0; ix < met->nx; ix++)
00809       for (iy = 0; iy < met->ny; iy++) {
00810
00811         /* Get surface height... */
00812         lon = met->lon[ix];
00813         if (lon < topo_lon[0])
00814           lon += 360;
00815         else if (lon > topo_lon[topo_nx - 1])
00816           lon -= 360;
00817         lat = met->lat[iy];
00818         tx = locate_reg(topo_lon, topo_nx, lon);
00819         ty = locate_reg(topo_lat, topo_ny, lat);
00820         z0 = LIN(topo_lon[tx], topo_z[tx][ty],
00821                  topo_lon[tx + 1], topo_z[tx + 1][ty], lon);
00822         z1 = LIN(topo_lon[tx], topo_z[tx][ty + 1],
00823                  topo_lon[tx + 1], topo_z[tx + 1][ty + 1], lon);
00824         z0 = LIN(topo_lat[ty], z0, topo_lat[ty + 1], z1, lat);
00825
00826         /* Find surface pressure level... */
00827         ip0 = locate_irr(met->p, met->np, met->ps[ix][iy]);
00828
00829         /* Get surface temperature... */
00830         ts = LIN(met->p[ip0], met->t[ix][iy][ip0],
00831                  met->p[ip0 + 1], met->t[ix][iy][ip0 + 1], met->ps[ix][iy]);
00832
00833         /* Upper part of profile... */
00834         met->z[ix][iy][ip0 + 1]
00835           = (float) (z0 + 8.31441 / 28.9647 / G0
00836                      * 0.5 * (ts + met->t[ix][iy][ip0 + 1])
00837                      * log(met->ps[ix][iy] / met->p[ip0 + 1]));
00838         for (ip = ip0 + 2; ip < met->np; ip++)
00839           met->z[ix][iy][ip]
00840             = (float) (met->z[ix][iy][ip - 1] + 8.31441 / 28.9647 / G0
00841                        * 0.5 * (met->t[ix][iy][ip - 1] + met->t[ix][iy][ip])
00842                        * log(met->p[ip - 1] / met->p[ip]));
00843       }
00844
00845     /* Smooth fields... */
00846     for (ip = 0; ip < met->np; ip++) {
00847
```

```
00848     /* Median filter... */
00849     for (ix = 0; ix < met->nx; ix++)
00850       for (iy = 0; iy < met->nx; iy++) {
00851         n = 0;
00852         for (ix2 = ix - 2; ix2 <= ix + 2; ix2++) {
00853           ix3 = ix2;
00854           if (ix3 < 0)
00855             ix3 += met->nx;
00856           if (ix3 >= met->nx)
00857             ix3 -= met->nx;
00858           for (iy2 = GSL_MAX(iy - 2, 0); iy2 <= GSL_MIN(iy + 2, met->ny - 1);
00859                iy2++)
00860             if (gsl_finite(met->z[ix3][iy2][ip])) {
00861               data[n] = met->z[ix3][iy2][ip];
00862               n++;
00863             }
00864         }
00865         if (n > 0) {
00866           gsl_sort(data, 1, (size_t) n);
00867           help[ix][iy] = (float)
00868             gsl_stats_median_from_sorted_data(data, 1, (size_t) n);
00869         } else
00870           help[ix][iy] = GSL_NAN;
00871       }
00872
00873     /* Copy data... */
00874     for (ix = 0; ix < met->nx; ix++)
00875       for (iy = 0; iy < met->nx; iy++)
00876         met->z[ix][iy][ip] = help[ix][iy];
00877   }
00878 }
00879
00880 /*****************************************************************************/
00881
00882 void read_met_help(
00883   int ncid,
00884   char *varname,
00885   char *varname2,
00886   met_t * met,
00887   float dest[EX][EY][EP],
00888   float scl) {
00889
00890   static float help[EX * EY * EP];
00891
00892   int ip, ix, iy, varid;
00893
00894   /* Check if variable exists... */
00895   if (nc_inq_varid(ncid, varname, &varid) != NC_NOERR)
00896     if (nc_inq_varid(ncid, varname2, &varid) != NC_NOERR)
00897       return;
00898
00899   /* Read data... */
00900   NC(nc_get_var_float(ncid, varid, help));
00901
00902   /* Copy and check data... */
00903   for (ix = 0; ix < met->nx; ix++)
00904     for (iy = 0; iy < met->ny; iy++)
00905       for (ip = 0; ip < met->np; ip++) {
00906         dest[ix][iy][ip] = help[(ip * met->ny + iy) * met->nx + ix];
00907         if (fabsf(dest[ix][iy][ip]) < 1e14f)
00908           dest[ix][iy][ip] *= scl;
00909         else
00910           dest[ix][iy][ip] = GSL_NAN;
00911       }
00912 }
00913
00914 /*****************************************************************************/
00915
00916 void read_met_periodic(
00917   met_t * met) {
00918
00919   int ip, iy;
00920
00921   /* Check longitudes... */
00922   if (!(fabs(met->lon[met->nx - 1] - met->lon[0]
00923              + met->lon[1] - met->lon[0] - 360) < 0.01))
00924     return;
00925
00926   /* Increase longitude counter... */
00927   if ((++met->nx) > EX)
00928     ERRMSG("Cannot create periodic boundary conditions!");
00929
00930   /* Set longitude... */
00931   met->lon[met->nx - 1] = met->lon[met->nx - 2] + met->lon[1] - met->
      lon[0];
00932
00933   /* Loop over latitudes and pressure levels... */
```

```
00934   for (iy = 0; iy < met->ny; iy++)
00935     for (ip = 0; ip < met->np; ip++) {
00936       met->ps[met->nx - 1][iy] = met->ps[0][iy];
00937       met->pt[met->nx - 1][iy] = met->pt[0][iy];
00938       met->z[met->nx - 1][iy][ip] = met->z[0][iy][ip];
00939       met->t[met->nx - 1][iy][ip] = met->t[0][iy][ip];
00940       met->u[met->nx - 1][iy][ip] = met->u[0][iy][ip];
00941       met->v[met->nx - 1][iy][ip] = met->v[0][iy][ip];
00942       met->w[met->nx - 1][iy][ip] = met->w[0][iy][ip];
00943       met->pv[met->nx - 1][iy][ip] = met->pv[0][iy][ip];
00944       met->h2o[met->nx - 1][iy][ip] = met->h2o[0][iy][ip];
00945       met->o3[met->nx - 1][iy][ip] = met->o3[0][iy][ip];
00946     }
00947 }
00948
00949 /*****************************************************************************/
00950
00951 void read_met_sample(
00952   ctl2_t * ctl2,
00953   met_t * met) {
00954
00955   met_t *help;
00956
00957   float w, wsum;
00958
00959   int ip, ip2, ix, ix2, ix3, iy, iy2;
00960
00961   /* Check parameters... */
00962   if (ctl2->met_dp <= 1 && ctl2->met_dx <= 1 && ctl2->met_dy <= 1
00963       && ctl2->met_sp <= 1 && ctl2->met_sx <= 1 && ctl2->met_sy <= 1)
00964     return;
00965
00966   /* Allocate... */
00967   ALLOC(help, met_t, 1);
00968
00969   /* Copy data... */
00970   help->nx = met->nx;
00971   help->ny = met->ny;
00972   help->np = met->np;
00973   memcpy(help->lon, met->lon, sizeof(met->lon));
00974   memcpy(help->lat, met->lat, sizeof(met->lat));
00975   memcpy(help->p, met->p, sizeof(met->p));
00976
00977   /* Smoothing... */
00978   for (ix = 0; ix < met->nx; ix += ctl2->met_dx) {
00979     for (iy = 0; iy < met->ny; iy += ctl2->met_dy) {
00980       for (ip = 0; ip < met->np; ip += ctl2->met_dp) {
00981         help->ps[ix][iy] = 0;
00982         help->pt[ix][iy] = 0;
00983         help->z[ix][iy][ip] = 0;
00984         help->t[ix][iy][ip] = 0;
00985         help->u[ix][iy][ip] = 0;
00986         help->v[ix][iy][ip] = 0;
00987         help->w[ix][iy][ip] = 0;
00988         help->pv[ix][iy][ip] = 0;
00989         help->h2o[ix][iy][ip] = 0;
00990         help->o3[ix][iy][ip] = 0;
00991         wsum = 0;
00992         for (ix2 = ix - ctl2->met_sx + 1; ix2 <= ix + ctl2->met_sx - 1; ix2++) {
00993           ix3 = ix2;
00994           if (ix3 < 0)
00995             ix3 += met->nx;
00996           else if (ix3 >= met->nx)
00997             ix3 -= met->nx;
00998
00999           for (iy2 = GSL_MAX(iy - ctl2->met_sy + 1, 0);
01000                iy2 <= GSL_MIN(iy + ctl2->met_sy - 1, met->ny - 1); iy2++)
01001             for (ip2 = GSL_MAX(ip - ctl2->met_sp + 1, 0);
01002                  ip2 <= GSL_MIN(ip + ctl2->met_sp - 1, met->np - 1); ip2++) {
01003               w = (float) (1.0 - fabs(ix - ix2) / ctl2->met_sx)
01004                 * (float) (1.0 - fabs(iy - iy2) / ctl2->met_sy)
01005                 * (float) (1.0 - fabs(ip - ip2) / ctl2->met_sp);
01006               help->ps[ix][iy] += w * met->ps[ix3][iy2];
01007               help->pt[ix][iy] += w * met->pt[ix3][iy2];
01008               help->z[ix][iy][ip] += w * met->z[ix3][iy2][ip2];
01009               help->t[ix][iy][ip] += w * met->t[ix3][iy2][ip2];
01010               help->u[ix][iy][ip] += w * met->u[ix3][iy2][ip2];
01011               help->v[ix][iy][ip] += w * met->v[ix3][iy2][ip2];
01012               help->w[ix][iy][ip] += w * met->w[ix3][iy2][ip2];
01013               help->pv[ix][iy][ip] += w * met->pv[ix3][iy2][ip2];
01014               help->h2o[ix][iy][ip] += w * met->h2o[ix3][iy2][ip2];
01015               help->o3[ix][iy][ip] += w * met->o3[ix3][iy2][ip2];
01016               wsum += w;
01017             }
01018       }
01019       help->ps[ix][iy] /= wsum;
01020       help->pt[ix][iy] /= wsum;
```

```
01021         help->t[ix][iy][ip] /= wsum;
01022         help->z[ix][iy][ip] /= wsum;
01023         help->u[ix][iy][ip] /= wsum;
01024         help->v[ix][iy][ip] /= wsum;
01025         help->w[ix][iy][ip] /= wsum;
01026         help->pv[ix][iy][ip] /= wsum;
01027         help->h2o[ix][iy][ip] /= wsum;
01028         help->o3[ix][iy][ip] /= wsum;
01029       }
01030     }
01031   }
01032
01033   /* Downsampling... */
01034   met->nx = 0;
01035   for (ix = 0; ix < help->nx; ix += ctl2->met_dx) {
01036     met->lon[met->nx] = help->lon[ix];
01037     met->ny = 0;
01038     for (iy = 0; iy < help->ny; iy += ctl2->met_dy) {
01039       met->lat[met->ny] = help->lat[iy];
01040       met->ps[met->nx][met->ny] = help->ps[ix][iy];
01041       met->pt[met->nx][met->ny] = help->pt[ix][iy];
01042       met->np = 0;
01043       for (ip = 0; ip < help->np; ip += ctl2->met_dp) {
01044         met->p[met->np] = help->p[ip];
01045         met->z[met->nx][met->ny][met->np] = help->z[ix][iy][ip];
01046         met->t[met->nx][met->ny][met->np] = help->t[ix][iy][ip];
01047         met->u[met->nx][met->ny][met->np] = help->u[ix][iy][ip];
01048         met->v[met->nx][met->ny][met->np] = help->v[ix][iy][ip];
01049         met->w[met->nx][met->ny][met->np] = help->w[ix][iy][ip];
01050         met->pv[met->nx][met->ny][met->np] = help->pv[ix][iy][ip];
01051         met->h2o[met->nx][met->ny][met->np] = help->h2o[ix][iy][ip];
01052         met->o3[met->nx][met->ny][met->np] = help->o3[ix][iy][ip];
01053         met->np++;
01054       }
01055       met->ny++;
01056     }
01057     met->nx++;
01058   }
01059
01060   /* Free... */
01061   free(help);
01062 }
```

5.5 jurassic.c File Reference

JURASSIC library definitions.

Functions

- size_t atm2x (ctl_t *ctl, atm_t *atm, gsl_vector *x, int *iqa, int *ipa)

  Compose state vector or parameter vector.
- void atm2x_help (atm_t *atm, double zmin, double zmax, double *value, int val_iqa, gsl_vector *x, int *iqa, int *ipa, size_t *n)

  Add elements to state vector.
- double brightness (double rad, double nu)

  Compute brightness temperature.
- void cart2geo (double *x, double *z, double *lon, double *lat)

  Convert Cartesian coordinates to geolocation.
- void climatology (ctl_t *ctl, atm_t *atm)

  Interpolate climatological data.
- double ctmco2 (double nu, double p, double t, double u)

  Compute carbon dioxide continuum (optical depth).
- double ctmh2o (double nu, double p, double t, double q, double u)

  Compute water vapor continuum (optical depth).
- double ctmn2 (double nu, double p, double t)

  Compute nitrogen continuum (absorption coefficient).

Read atmospheric data.

- void read_ctl (int argc, char *argv[], ctl_t *ctl)

  Read forward model control parameters.
- void read_matrix (const char *dirname, const char *filename, gsl_matrix *matrix)

  Read matrix.
- void read_obs (const char *dirname, const char *filename, ctl_t *ctl, obs_t *obs)

  Read observation data.
- void read_shape (const char *filename, double *x, double *y, int *n)

  Read shape function.
- double refractivity (double p, double t)

  Compute refractivity (return value is n - 1).
- double scan_ctl (int argc, char *argv[], const char *varname, int arridx, const char *defvalue, char *value)

  Search control parameter file for variable entry.
- void tangent_point (los_t *los, double *tpz, double *tplon, double *tplat)

  Find tangent point of a given LOS.
- void time2jsec (int year, int mon, int day, int hour, int min, int sec, double remain, double *jsec)

  Convert date to seconds.
- void timer (const char *name, const char *file, const char *func, int line, int mode)

  Measure wall-clock time.
- void write_atm (const char *dirname, const char *filename, ctl_t *ctl, atm_t *atm)

  Write atmospheric data.
- void write_matrix (const char *dirname, const char *filename, ctl_t *ctl, gsl_matrix *matrix, atm_t *atm, obs_t *obs, const char *rowspace, const char *colspace, const char *sort)

  Write matrix.
- void write_obs (const char *dirname, const char *filename, ctl_t *ctl, obs_t *obs)

  Write observation data.
- void x2atm (ctl_t *ctl, gsl_vector *x, atm_t *atm)

  Decompose parameter vector or state vector.
- void x2atm_help (atm_t *atm, double zmin, double zmax, double *value, gsl_vector *x, size_t *n)

  Extract elements from state vector.
- void y2obs (ctl_t *ctl, gsl_vector *y, obs_t *obs)

  Decompose measurement vector.

5.5.1 Detailed Description

JURASSIC library definitions.

Definition in file jurassic.c.

5.5.2 Function Documentation

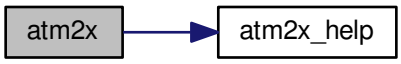
5.5.2.1 size_t atm2x (ctl_t * *ctl*, atm_t * *atm*, gsl_vector * *x*, int * *iqa*, int * *ipa*)

Compose state vector or parameter vector.

Definition at line 29 of file jurassic.c.

```
00034            {
00035
00036   int ig, iw;
00037
00038   size_t n = 0;
00039
00040   /* Add pressure... */
00041   atm2x_help(atm, ctl->retp_zmin, ctl->retp_zmax,
00042              atm->p, IDXP, x, iqa, ipa, &n);
00043
00044   /* Add temperature... */
00045   atm2x_help(atm, ctl->rett_zmin, ctl->rett_zmax,
00046              atm->t, IDXT, x, iqa, ipa, &n);
00047
00048   /* Add volume mixing ratios... */
00049   for (ig = 0; ig < ctl->ng; ig++)
00050     atm2x_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
00051                atm->q[ig], IDXQ(ig), x, iqa, ipa, &n);
00052
00053   /* Add extinction... */
00054   for (iw = 0; iw < ctl->nw; iw++)
00055     atm2x_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
00056                atm->k[iw], IDXK(iw), x, iqa, ipa, &n);
00057
00058   return n;
00059 }
```

Here is the call graph for this function:



5.5.2.2 void atm2x_help (atm_t * *atm*, double *zmin*, double *zmax*, double * *value*, int *val_iqa*, gsl_vector * *x*, int * *iqa*, int * *ipa*, size_t * *n*)

Add elements to state vector.

Definition at line 63 of file jurassic.c.

```
00072              {
00073
00074   int ip;
00075
00076   /* Add elements to state vector... */
00077   for (ip = 0; ip < atm->np; ip++)
00078     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
00079       if (x != NULL)
00080         gsl_vector_set(x, *n, value[ip]);
00081       if (iqa != NULL)
00082         iqa[*n] = val_iqa;
00083       if (ipa != NULL)
00084         ipa[*n] = ip;
00085       (*n)++;
00086     }
00087 }
```

5.5.2.3 double brightness (double *rad*, double *nu*)

Compute brightness temperature.

Definition at line 91 of file jurassic.c.

```
00093              {
00094
00095   return C2 * nu / gsl_log1p(C1 * POW3(nu) / rad);
00096 }
```

5.5.2.4 void cart2geo (double * x, double * z, double * lon, double * lat)

Convert Cartesian coordinates to geolocation.

Definition at line 101 of file jurassic.c.

```
00105               {
00106
00107   double radius;
00108
00109   radius = NORM(x);
00110   *lat = asin(x[2] / radius) * 180 / M_PI;
00111   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00112   *z = radius - RE;
00113 }
```

5.5.2.5 void climatology (ctl_t * ctl, atm_t * atm_mean)

Interpolate climatological data.

Definition at line 117 of file jurassic.c.

```
00119               {
00120
00121   static double z[121] = {
00122     0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00123     20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
00124     38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00125     56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00126     74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00127     92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00128     108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00129   };
00130
00131   static double pre[121] = {
00132     1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00133     357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00134     104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00135     29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00136     10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00137     3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00138     1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00139     0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00140     0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00141     0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00142     0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00143     0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00144     0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00145     0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00146     0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00147     0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00148     9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00149     4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00150     2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00151   };
00152
00153   static double tem[121] = {
00154     285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00155     229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00156     215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00157     222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00158     241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00159     262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00160     258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,
00161     237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00162     220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00163     207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00164     190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00165     178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00166     201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00167     272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00168   };
00169
00170   static double c2h2[121] = {
00171     1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00172     2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
```

```
00173     5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00174     2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00175     9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00176     1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00177     1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00178     1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00179     2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00180     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00181     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00182     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00183   };
00184
00185   static double c2h6[121] = {
00186     2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00187     1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00188     5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00189     2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00190     2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00191     1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00192     5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00193     2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00194     1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00195     7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,
00196     3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00197     1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00198     4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00199     1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00200     3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00201     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00202     0, 0, 0, 0, 0, 0, 0, 0, 0
00203   };
00204
00205   static double ccl4[121] = {
00206     1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00207     1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00208     8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00209     3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00210     3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
00211     4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00212     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00213     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00214     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00215     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00216     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00217     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00218     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00219     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00220     1e-14, 1e-14, 1e-14
00221   };
00222
00223   static double ch4[121] = {
00224     1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00225     1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00226     1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00227     1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00228     1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00229     8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00230     6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00231     4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00232     3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00233     2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00234     1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00235     1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00236     1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00237     9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00238     7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00239     5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00240     4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00241     3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00242     2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00243     2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00244     1.782e-08
00245   };
00246
00247   static double clo[121] = {
00248     7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00249     6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00250     8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00251     2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00252     1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00253     2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00254     4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00255     5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,
00256     3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00257     1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00258     6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00259     2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
```

```
00260     8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00261     3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00262     1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00263     3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00264     1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00265     3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00266     1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00267     5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00268     3.148e-15
00269   };
00270
00271   static double clono2[121] = {
00272     1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00273     1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00274     2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00275     2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00276     8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00277     6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00278     1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00279     1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00280     1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00281     1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00282     9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
00283     6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00284     3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00285     1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00286     8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00287     3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00288     9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00289     3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00290     2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00291     2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00292     4.041e-27
00293   };
00294
00295   static double co[121] = {
00296     1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00297     9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
00298     5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00299     2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00300     1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00301     2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00302     3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00303     3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00304     6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00305     2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00306     8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00307     2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00308     3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00309     6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00310     1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00311     1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00312     3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00313     5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00314     6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00315     7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05
00316   };
00317
00318   static double cof2[121] = {
00319     7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00320     6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00321     7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00322     4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00323     1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00324     1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00325     1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00326     8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00327     5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00328     2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00329     7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00330     1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00331     4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00332     1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00333     2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00334     4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
00335     7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00336     1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00337     3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00338     1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00339     4.662e-18
00340   };
00341
00342   static double f11[121] = {
00343     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00344     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00345     2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00346     1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
```

```
00347     7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00348     5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00349     1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00350     3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00351     6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00352     1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00353     1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00354     2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00355     2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00356     2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00357     2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00358     1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00359     1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00360     1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00361     2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00362     4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00363   };
00364
00365   static double f12[121] = {
00366     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00367     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00368     5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00369     4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
00370     2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00371     5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00372     2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00373     8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00374     3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00375     1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00376     8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00377     4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00378     2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00379     9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00380     4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00381     1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00382     7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00383     3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00384     1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
00385     1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00386   };
00387
00388   static double f14[121] = {
00389     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00390     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00391     8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00392     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00393     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00394     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00395     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00396     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00397     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00398     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00399     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00400     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00401     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00402     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00403     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00404     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00405     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00406   };
00407
00408   static double f22[121] = {
00409     1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00410     1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00411     1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00412     7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00413     4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00414     3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00415     1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00416     1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00417     8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00418     5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00419     4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00420     3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00421     3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
00422     2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00423     2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00424     2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00425     1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00426     1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00427     1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00428     1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00429   };
00430
00431   static double h2o[121] = {
00432     0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00433     0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
```

```
00434     6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00435     4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00436     4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00437     5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00438     5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00439     6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00440     6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00441     6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00442     5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00443     4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00444     3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00445     2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00446     1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00447     5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00448     1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00449     7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00450     3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00451     1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00452   };
00453
00454   static double h2o2[121] = {
00455     1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00456     4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
00457     3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00458     1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00459     8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00460     1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00461     1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00462     6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00463     5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00464     4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00465     3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00466     2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00467     1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00468     1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00469     9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00470     7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00471     4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
00472     3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00473     2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00474     2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00475     1.775e-12
00476   };
00477
00478   static double hcn[121] = {
00479     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00480     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00481     5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00482     1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00483     1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00484     1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00485     1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00486     1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00487     1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00488     9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00489     8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00490     7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00491     6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00492     6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00493     6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00494     6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00495     5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00496     5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00497     5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00498     5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00499   };
00500
00501   static double hno3[121] = {
00502     1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00503     2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00504     5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00505     3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00506     8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00507     3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00508     8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
00509     1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00510     6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00511     3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00512     1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00513     9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00514     5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00515     2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00516     1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
00517     5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00518     2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00519     1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00520     5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
```

```
00521     3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00522     2.332e-14
00523   };
00524
00525   static double hno4[121] = {
00526     6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00527     1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00528     3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00529     1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00530     2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00531     1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00532     3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00533     2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00534     1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00535     2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00536     1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00537     5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00538     2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00539     8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00540     3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00541     1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00542     3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00543     1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
00544     5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00545     2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00546     1.64e-18
00547   };
00548
00549   static double hocl[121] = {
00550     1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00551     2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00552     5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00553     1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00554     4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00555     7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00556     1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00557     6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00558     1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
00559     3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00560     6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00561     1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00562     1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00563     2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00564     3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00565     4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00566     5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00567     6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00568     9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00569     2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00570     7.881e-21
00571   };
00572
00573   static double n2o[121] = {
00574     3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00575     3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00576     2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00577     2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00578     1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00579     7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00580     2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00581     1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00582     5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00583     2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00584     1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00585     1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00586     9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00587     7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00588     5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00589     4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00590     3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00591     2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00592     2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00593     2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00594   };
00595
00596   static double n2o5[121] = {
00597     1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00598     1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00599     4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00600     7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00601     3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00602     2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00603     2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
00604     6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00605     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00606     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00607     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
```

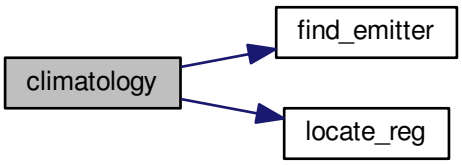
```
00608     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00609     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00610     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00611     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00612     1e-16, 1e-16
00613   };
00614
00615   static double nh3[121] = {
00616     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00617     1e-10, 1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00618     4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00619     5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00620     6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00621     1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00622     1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00623     1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00624     2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00625     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00626     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00627     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00628     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00629     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00630     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00631     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00632     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00633     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00634     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00635     1.914e-17
00636   };
00637
00638   static double no[121] = {
00639     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00640     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00641     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00642     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00643     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00644     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00645     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
00646     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00647     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00648     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00649     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00650     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00651     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00652     1.077e-08, 1.367e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00653     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00654     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00655     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00656     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00657     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00658     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00659     0.0001133
00660   };
00661
00662   static double no2[121] = {
00663     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00664     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00665     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00666     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00667     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00668     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00669     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00670     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00671     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00672     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00673     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00674     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00675     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00676     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00677     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00678     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00679     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00680     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00681   };
00682
00683   static double o3[121] = {
00684     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00685     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00686     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00687     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00688     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00689     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00690     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
00691     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00692     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00693     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00694     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
```

```
00695     4.509e-07, 3.922e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00696     2.476e-07, 2.284e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00697     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00698     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00699     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00700     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00701     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00702     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00703     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00704     3.665e-10
00705   };
00706
00707   static double ocs[121] = {
00708     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00709     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00710     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00711     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00712     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00713     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00714     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00715     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00716     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00717     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00718     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00719     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00720     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00721     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00722     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00723     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00724     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00725     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00726     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00727     1.091e-14, 1.091e-14, 1.091e-14
00728   };
00729
00730   static double sf6[121] = {
00731     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00732     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
00733     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00734     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00735     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00736     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00737     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00738     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00739     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00740     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00741     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00742     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00743     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00744     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00745     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00746     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00747     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00748     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00749     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00750   };
00751
00752   static double so2[121] = {
00753     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00754     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00755     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00756     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00757     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00758     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00759     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00760     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00761     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00762     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00763     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00764     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00765     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00766     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00767     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00768   };
00769
00770   static int ig_co2 = -999;
00771
00772   double co2, *q[NG] = { NULL };
00773
00774   int ig, ip, iw, iz;
00775
00776   /* Find emitter index of CO2... */
00777   if (ig_co2 == -999)
00778     ig_co2 = find_emitter(ctl, "CO2");
00779
00780   /* Identify variable... */
00781   for (ig = 0; ig < ctl->ng; ig++) {
```

```
00782     q[ig] = NULL;
00783     if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00784       q[ig] = c2h2;
00785     if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00786       q[ig] = c2h6;
00787     if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00788       q[ig] = ccl4;
00789     if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00790       q[ig] = ch4;
00791     if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00792       q[ig] = clo;
00793     if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00794       q[ig] = clono2;
00795     if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00796       q[ig] = co;
00797     if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00798       q[ig] = cof2;
00799     if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00800       q[ig] = f11;
00801     if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00802       q[ig] = f12;
00803     if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00804       q[ig] = f14;
00805     if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00806       q[ig] = f22;
00807     if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00808       q[ig] = h2o;
00809     if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00810       q[ig] = h2o2;
00811     if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00812       q[ig] = hcn;
00813     if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00814       q[ig] = hno3;
00815     if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00816       q[ig] = hno4;
00817     if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00818       q[ig] = hocl;
00819     if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
00820       q[ig] = n2o;
00821     if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00822       q[ig] = n2o5;
00823     if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00824       q[ig] = nh3;
00825     if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00826       q[ig] = no;
00827     if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00828       q[ig] = no2;
00829     if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00830       q[ig] = o3;
00831     if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00832       q[ig] = ocs;
00833     if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00834       q[ig] = sf6;
00835     if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00836       q[ig] = so2;
00837   }
00838
00839   /* Loop over atmospheric data points... */
00840   for (ip = 0; ip < atm->np; ip++) {
00841
00842     /* Get altitude index... */
00843     iz = locate_reg(z, 121, atm->z[ip]);
00844
00845     /* Interpolate pressure... */
00846     atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00847
00848     /* Interpolate temperature... */
00849     atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00850
00851     /* Interpolate trace gases... */
00852     for (ig = 0; ig < ctl->ng; ig++)
00853       if (q[ig] != NULL)
00854         atm->q[ig][ip] =
00855           LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00856       else
00857         atm->q[ig][ip] = 0;
00858
00859     /* Set CO2... */
00860     if (ig_co2 >= 0) {
00861       co2 =
00862         371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00863       atm->q[ig_co2][ip] = co2;
00864     }
00865
00866     /* Set extinction to zero... */
00867     for (iw = 0; iw < ctl->nw; iw++)
00868       atm->k[iw][ip] = 0;
```

```
00869   }
00870 }
```

Here is the call graph for this function:



5.5.2.6 double ctmco2 (double *nu*, double *p*, double *t*, double *u*)

Compute carbon dioxide continuum (optical depth).

Definition at line 874 of file jurassic.c.

```
00878             {
00879
00880   static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00881     1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00882     1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00883     1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00884     2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00885     3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00886     4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00887     5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00888     7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00889     .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00890     .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00891     .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00892     .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00893     .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00894     .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00895     .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00896     .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00897     .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00898     .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00899     .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00900     .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00901     .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00902     .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00903     .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00904     .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00905     .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00906     .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
00907     .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00908     .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,
00909     .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,
00910     1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,
00911     2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,
00912     3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,
00913     4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,
00914     7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,
00915     12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,
00916     21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,
00917     35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,
00918     60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,
00919     107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,
00920     199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,
00921     386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,
00922     756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,
00923     1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,
00924     3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,
```

```
00925     5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,
00926     2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,
00927     820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,
00928     1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,
00929     6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,
00930     4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,
00931     1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,
00932     999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,
00933     476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,
00934     251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,
00935     133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,
00936     73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,
00937     42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,
00938     25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,
00939     15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,
00940     9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,
00941     5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,
00942     3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,
00943     2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,
00944     1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,
00945     1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,
00946     .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,
00947     .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,
00948     .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,
00949     .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,
00950     .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,
00951     .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,
00952     .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,
00953     .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,
00954     .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,
00955     .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,
00956     .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,
00957     .099745, .091118, .083404, .076494, .070292, .064716, .059697,
00958     .055173, .051093, .047411, .044089, .041092, .038392, .035965,
00959     .033789, .031846, .030122, .028607, .02729, .026169, .025209,
00960     .024405, .023766, .023288, .022925, .022716, .022681, .022685,
00961     .022768, .023133, .023325, .023486, .024004, .024126, .024083,
00962     .023785, .024023, .023029, .021649, .021108, .019454, .017809,
00963     .017292, .016635, .017037, .018068, .018977, .018756, .017847,
00964     .016557, .016142, .014459, .012869, .012381, .010875, .0098701,
00965     .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,
00966     .014362, .015017, .016507, .017741, .01768, .017784, .0171,
00967     .016357, .016172, .017257, .018978, .020935, .021741, .023567,
00968     .025183, .025589, .026732, .027648, .028278, .028215, .02856,
00969     .029015, .029062, .028851, .028497, .027825, .027801, .026523,
00970     .02487, .022967, .022168, .020194, .018605, .017903, .018439,
00971     .019697, .020311, .020855, .020057, .018608, .016738, .015963,
00972     .013844, .011801, .011134, .0097573, .0086007, .0086226,
00973     .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,
00974     .014657, .015771, .016351, .016079, .014829, .013431, .013185,
00975     .013207, .01448, .016176, .017971, .018265, .019526, .020455,
00976     .019797, .019802, .0194, .018176, .017505, .016197, .015339,
00977     .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,
00978     .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,
00979     .0041501, .003752, .0033996, .0030865, .0028077, .0025586,
00980     .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,
00981     .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,
00982     8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,
00983     5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,
00984     3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,
00985     2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,
00986     2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,
00987     1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,
00988     1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,
00989     1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,
00990     1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,
00991     2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,
00992     2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,
00993     3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,
00994     4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,
00995     6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,
00996     8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
00997     .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
00998     .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
00999     .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01000     .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01001     .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01002     .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01003     .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01004     .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01005     .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01006     .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01007     .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01008     .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01009     .056326, .0602, .064093, .068521, .073278, .077734, .083064,
01010     .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,
01011     .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,
```

```
01012     .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,
01013     .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,
01014     .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,
01015     .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,
01016     .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,
01017     1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,
01018     1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,
01019     1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,
01020     2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,
01021     4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,
01022     7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,
01023     9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,
01024     10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,
01025     16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,
01026     27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,
01027     43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,
01028     63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,
01029     98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,
01030     159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,
01031     264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,
01032     449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,
01033     789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,
01034     1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,
01035     2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,
01036     4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,
01037     9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,
01038     20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,
01039     40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,
01040     43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,
01041     44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,
01042     21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,
01043     28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,
01044     31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,
01045     54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,
01046     29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,
01047     11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,
01048     5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,
01049     2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,
01050     1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,
01051     845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,
01052     489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,
01053     291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,
01054     176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,
01055     108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,
01056     67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,
01057     42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,
01058     27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,
01059     17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,
01060     11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,
01061     7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,
01062     4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,
01063     3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,
01064     2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,
01065     1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,
01066     .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,
01067     .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,
01068     .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,
01069     .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,
01070     .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,
01071     .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,
01072     .093203, .088815, .084641, .080671, .076892, .073296, .069873,
01073     .066613, .06351, .060555, .05774, .055058, .052504, .050071,
01074     .047752, .045543, .043438, .041432, .039521, .037699, .035962,
01075     .034307, .032729, .031225, .029791, .028423, .02712, .025877,
01076     .024692, .023563, .022485, .021458, .020478, .019543, .018652,
01077     .017802, .016992, .016219, .015481, .014778, .014107, .013467,
01078     .012856, .012274, .011718, .011188, .010682, .0102, .0097393,
01079     .0093001, .008881, .0084812, .0080997, .0077358, .0073885,
01080     .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,
01081     .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,
01082     .0040884, .0039088, .0037373, .0035739, .003418, .0032693,
01083     .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,
01084     .0024074, .0023066, .0022108, .0021196, .002033, .0019507,
01085     .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,
01086     .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,
01087     .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,
01088     .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,
01089     9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,
01090     .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,
01091     .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,
01092     .0016144, .0016312, .001705, .0017892, .0018499, .0019715,
01093     .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,
01094     .0031193, .003346, .0034552, .0036906, .0037584, .0040084,
01095     .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,
01096     .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,
01097     .0060972, .0055539, .0055653, .0055772, .005331, .0054953,
01098     .0055919, .0058684, .006183, .0066675, .0069808, .0075142,
```

```
01099     .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,
01100     .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,
01101     .0105, .010617, .010706, .01078, .011177, .011212, .011304,
01102     .011446, .011603, .011816, .012165, .012545, .013069, .013539,
01103     .01411, .014776, .016103, .017016, .017994, .018978, .01998,
01104     .021799, .022745, .023681, .024627, .025562, .026992, .027958,
01105     .029013, .030154, .031402, .03228, .033651, .035272, .037088,
01106     .039021, .041213, .043597, .045977, .04877, .051809, .054943,
01107     .058064, .061528, .06537, .069309, .071928, .075752, .079589,
01108     .083352, .084096, .087497, .090817, .091198, .094966, .099045,
01109     .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,
01110     .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,
01111     .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,
01112     .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01113     .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01114     .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01115     1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01116     2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01117     3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01118     5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01119     10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01120     18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01121     30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01122     57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01123     107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01124     205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01125     394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01126     501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01127     455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01128     376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01129     367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01130     588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01131     962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01132     1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01133     751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01134     777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01135     322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01136     654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01137     679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01138     818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01139     364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01140     155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01141     74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01142     38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01143     21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01144     12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01145     7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01146     4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01147     2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01148     1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01149     1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01150     .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01151     .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01152     .26732, .25337, .24017, .22774, .21601, .20479, .19426
01153   };
01154
01155   static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01156     6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01157     9.2421e-5, 9.6846e-5, 1.0149e-4, 1.0635e-4, 1.1145e-4, 1.1679e-4,
01158     1.224e-4, 1.2828e-4, 1.3444e-4, 1.409e-4, 1.4768e-4, 1.5479e-4,
01159     1.6224e-4, 1.7006e-4, 1.7826e-4, 1.8685e-4, 1.9587e-4, 2.0532e-4,
01160     2.1524e-4, 2.2565e-4, 2.3656e-4, 2.48e-4, 2.6001e-4, 2.7261e-4,
01161     2.8582e-4, 2.9968e-4, 3.1422e-4, 3.2948e-4, 3.4548e-4, 3.6228e-4,
01162     3.799e-4, 3.9838e-4, 4.1778e-4, 4.3814e-4, 4.595e-4, 4.8191e-4,
01163     5.0543e-4, 5.3012e-4, 5.5603e-4, 5.8321e-4, 6.1175e-4, 6.417e-4,
01164     6.7314e-4, 7.0614e-4, 7.4078e-4, 7.7714e-4, 8.1531e-4, 8.5538e-4,
01165     8.9745e-4, 9.4162e-4, 9.8798e-4, .0010367, .0010878, .0011415,
01166     .0011978, .001257, .0013191, .0013844, .001453, .0015249,
01167     .0016006, .00168, .0017634, .001851, .001943, .0020397, .0021412,
01168     .0022479, .00236, .0024778, .0026015, .0027316, .0028682,
01169     .0030117, .0031626, .0033211, .0034877, .0036628, .0038469,
01170     .0040403, .0042436, .0044574, .004682, .0049182, .0051665,
01171     .0054276, .0057021, .0059907, .0062942, .0066133, .0069489,
01172     .0073018, .0076729, .0080632, .0084738, .0089056, .0093599,
01173     .0098377, .01034, .010869, .011426, .012011, .012627, .013276,
01174     .013958, .014676, .015431, .016226, .017063, .017944, .018872,
01175     .019848, .020876, .021958, .023098, .024298, .025561, .026892,
01176     .028293, .029769, .031323, .032961, .034686, .036503, .038418,
01177     .040435, .042561, .044801, .047161, .049649, .052271, .055035,
01178     .057948, .061019, .064256, .06767, .07127, .075066, .079069,
01179     .083291, .087744, .092441, .097396, .10262, .10814, .11396,
01180     .1201, .12658, .13342, .14064, .14826, .1563, .1648, .17376,
01181     .18323, .19324, .2038, .21496, .22674, .23919, .25234, .26624,
01182     .28093, .29646, .31287, .33021, .34855, .36794, .38844, .41012,
01183     .43305, .45731, .48297, .51011, .53884, .56924, .60141, .63547,
01184     .67152, .70969, .75012, .79292, .83826, .8863, .93718, .99111,
01185     1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,
```

```
01186     1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,
01187     2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,
01188     4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,
01189     6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,
01190     11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,
01191     18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,
01192     31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,
01193     54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,
01194     97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,
01195     182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,
01196     356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,
01197     716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,
01198     1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,
01199     3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,
01200     6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,
01201     2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,
01202     843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,
01203     1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381.,
01204     9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,
01205     4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,
01206     2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,
01207     1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,
01208     525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,
01209     268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,
01210     139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,
01211     76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,
01212     43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,
01213     25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,
01214     14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,
01215     9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,
01216     5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,
01217     3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,
01218     2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,
01219     1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,
01220     .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,
01221     .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,
01222     .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,
01223     .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,
01224     .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,
01225     .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,
01226     .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,
01227     .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,
01228     .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,
01229     .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,
01230     .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,
01231     .1152, .10304, .092437, .083163, .075031, .067878, .061564,
01232     .055976, .051018, .046609, .042679, .03917, .036032, .033223,
01233     .030706, .02845, .026428, .024617, .022998, .021554, .02027,
01234     .019136, .018141, .017278, .016541, .015926, .015432, .015058,
01235     .014807, .014666, .014635, .014728, .014947, .01527, .015728,
01236     .016345, .017026, .017798, .018839, .019752, .020636, .021886,
01237     .022695, .02327, .023478, .024292, .023544, .022222, .021932,
01238     .020052, .018143, .017722, .017031, .017782, .01938, .020734,
01239     .020476, .019255, .017477, .016878, .014617, .012489, .011765,
01240     .0099077, .0086446, .0079446, .0078644, .0079763, .008671,
01241     .01001, .0108, .012933, .015349, .016341, .018484, .020254,
01242     .020254, .020478, .019591, .018595, .018385, .019913, .022254,
01243     .024847, .025809, .028053, .029924, .030212, .031367, .03222,
01244     .032739, .032537, .03286, .033344, .033507, .033499, .033339,
01245     .032809, .033041, .031723, .029837, .027511, .026603, .024032,
01246     .021914, .020948, .021701, .023425, .024259, .024987, .023818,
01247     .021768, .019223, .018144, .015282, .012604, .01163, .0097907,
01248     .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,
01249     .014378, .016761, .01726, .018997, .019998, .019809, .01819,
01250     .016358, .016099, .01617, .017939, .020223, .022521, .02277,
01251     .024279, .025247, .024222, .023989, .023224, .021493, .020362,
01252     .018596, .017309, .015975, .014466, .013171, .011921, .01078,
01253     .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,
01254     .0051459, .0046273, .0041712, .0037686, .0034119, .003095,
01255     .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,
01256     .001636, .0015017, .00138, .0012697, .0011694, .0010782,
01257     9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,
01258     6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,
01259     4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,
01260     2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,
01261     1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,
01262     1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,
01263     1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,
01264     1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,
01265     1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,
01266     1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,
01267     1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,
01268     2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,
01269     2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,
01270     3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,
01271     4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,
01272     6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,
```

```
01273     8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,
01274     .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,
01275     .0014953, .001572, .0016529, .0017381, .0018279, .0019226,
01276     .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,
01277     .002747, .0028921, .0030453, .0032071, .003378, .0035586,
01278     .0037494, .003951, .0041642, .0043897, .0046282, .0048805,
01279     .0051476, .0054304, .00573, .0060473, .0063837, .0067404,
01280     .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,
01281     .0099366, .010516, .011134, .011792, .012494, .013244, .014046,
01282     .014898, .015808, .016781, .017822, .018929, .020108, .02138,
01283     .022729, .02419, .02576, .027412, .029233, .031198, .033301,
01284     .035594, .038092, .040767, .04372, .046918, .050246, .053974,
01285     .058009, .061976, .066586, .071537, .076209, .081856, .087998,
01286     .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,
01287     .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,
01288     .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,
01289     .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,
01290     .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,
01291     .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,
01292     .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,
01293     .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,
01294     1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,
01295     1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,
01296     2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,
01297     5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,
01298     7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,
01299     10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,
01300     11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,
01301     18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,
01302     29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,
01303     46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,
01304     70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,
01305     114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,
01306     190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,
01307     324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,
01308     568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,
01309     1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543., 1672.8, 1813.4,
01310     1966.1, 2131.4, 2309.5, 2499.3, 2705., 2925.7, 3161.6, 3411.3,
01311     3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,
01312     6719.6, 7385., 8145., 8977.7, 9831.9, 10827., 11934., 13063.,
01313     14434., 15878., 17591., 19435., 21510., 23835., 26835., 29740.,
01314     32878., 36305., 39830., 43273., 46931., 50499., 49586., 51598.,
01315     53429., 54619., 55081., 55102., 54485., 53487., 52042., 42689.,
01316     42607., 44020., 47994., 54169., 53916., 55808., 56642., 46049.,
01317     44243., 32929., 30658., 21963., 20835., 15962., 13679., 17652.,
01318     19680., 22388., 25625., 29184., 32520., 35720., 38414., 40523.,
01319     49228., 48173., 45678., 41768., 37600., 41313., 42654., 44465.,
01320     55736., 56630., 65409., 63308., 66572., 61845., 60379., 56777.,
01321     51920., 46601., 41367., 36529., 32219., 28470., 25192., 22362.,
01322     19907., 17772., 15907., 14273., 12835., 11567., 10445., 9450.2,
01323     8565.1, 7776., 7070.8, 6439.2, 5872.3, 5362.4, 4903., 4488.3,
01324     4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296.,
01325     2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236.,
01326     1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,
01327     643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,
01328     371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,
01329     219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,
01330     131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,
01331     80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,
01332     49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,
01333     30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,
01334     19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,
01335     12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,
01336     7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,
01337     5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,
01338     3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,
01339     2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,
01340     1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,
01341     .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,
01342     .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,
01343     .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,
01344     .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01345     .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01346     .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01347     .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01348     .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01349     .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01350     .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01351     .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01352     .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01353     .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01354     .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01355     .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01356     .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01357     .003388, .0032329, .0030852, .0029445, .0028105, .0026829,
01358     .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01359     .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
```

```
01360     .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01361     .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01362     9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01363     7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01364     6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
01365     6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01366     7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01367     9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01368     .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01369     .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01370     .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01371     .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01372     .0053809, .0056699, .0059325, .0055488, .005634, .0056392,
01373     .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01374     .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01375     .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01376     .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01377     .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01378     .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01379     .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01380     .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01381     .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01382     .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01383     .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01384     .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01385     .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01386     .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01387     .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01388     .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01389     .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01390     .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01391     1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01392     1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01393     2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01394     4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01395     8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01396     15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01397     27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01398     46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01399     87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01400     176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01401     366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01402     600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01403     478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01404     545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01405     314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01406     696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01407     954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01408     1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01409     1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01410     1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01411     361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01412     578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01413     659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01414     1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01415     489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01416     199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01417     92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01418     46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01419     24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01420     13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01421     7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01422     4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01423     2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01424     1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01425     1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01426     .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01427     .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01428     .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
01429     .16469
01430   };
01431
01432   static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01433     3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01434     4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01435     6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01436     8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01437     1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01438     1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01439     1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01440     2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01441     3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01442     4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01443     6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01444     9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
01445     .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01446     .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
```

```
01447     .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01448     .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01449     .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01450     .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01451     .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
01452     .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01453     .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01454     .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01455     .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01456     .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01457     .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01458     .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01459     .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,
01460     .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01461     .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01462     .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01463     1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01464     1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01465     2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01466     4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01467     7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01468     13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01469     22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01470     40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01471     74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01472     137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01473     263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01474     525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01475     1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01476     2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01477     5457.9, 5946.2, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01478     11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01479     1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01480     722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01481     2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01482     7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01483     4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01484     1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01485     783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01486     387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01487     197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01488     105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01489     57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01490     31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01491     17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01492     10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01493     6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01494     3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01495     2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01496     1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01497     .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01498     .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01499     .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01500     .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01501     .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01502     .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01503     .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01504     .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01505     .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01506     .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01507     .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01508     .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01509     .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01510     .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01511     .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01512     .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01513     .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01514     .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01515     .022311, .023742, .0249, .025599, .026981, .026462, .025143,
01516     .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01517     .024163, .023728, .02199, .019506, .018591, .015576, .012784,
01518     .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01519     .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01520     .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01521     .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01522     .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01523     .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01524     .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01525     .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01526     .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01527     .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01528     .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01529     .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01530     .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01531     .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
01532     .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,
01533     .0031747, .0028864, .0026284, .002397, .002189, .0020017,
```

```
01534     .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,
01535     .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,
01536     6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,
01537     4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,
01538     2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,
01539     1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,
01540     1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,
01541     9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,
01542     7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,
01543     6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,
01544     7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,
01545     8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,
01546     9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,
01547     1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,
01548     1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,
01549     2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,
01550     3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,
01551     4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,
01552     5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,
01553     7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,
01554     .0010922, .001154, .0012195, .0012889, .0013626, .0014407,
01555     .0015235, .0016114, .0017048, .0018038, .001909, .0020207,
01556     .0021395, .0022657, .0023998, .0025426, .0026944, .002856,
01557     .0030281, .0032114, .0034068, .003615, .0038371, .004074,
01558     .004327, .0045971, .0048857, .0051942, .0055239, .0058766,
01559     .0062538, .0066573, .0070891, .007551, .0080455, .0085747,
01560     .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,
01561     .014415, .01541, .016475, .017621, .018857, .020175, .02162,
01562     .023185, .024876, .02672, .028732, .030916, .033319, .035939,
01563     .038736, .041847, .04524, .048715, .052678, .056977, .061203,
01564     .066184, .07164, .076952, .083477, .090674, .098049, .10697,
01565     .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,
01566     .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,
01567     .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,
01568     .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,
01569     .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,
01570     .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,
01571     .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,
01572     1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,
01573     1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,
01574     1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,
01575     3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,
01576     6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,
01577     8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,
01578     7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,
01579     11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,
01580     21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,
01581     35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,
01582     51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,
01583     83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,
01584     142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,
01585     249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,
01586     449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,
01587     833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350., 1466.3,
01588     1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,
01589     3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,
01590     5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203.,
01591     11223., 12358., 13493., 14916., 16416., 18236., 20222., 22501.,
01592     25102., 28358., 31707., 35404., 39538., 43911., 48391., 53193.,
01593     58028., 58082., 61276., 64193., 66294., 67480., 67921., 67423.,
01594     66254., 64341., 51737., 51420., 53072., 58145., 66195., 65358.,
01595     67377., 67869., 53509., 50553., 35737., 32425., 21704., 19974.,
01596     14457., 12142., 16798., 19489., 23049., 27270., 31910., 36457.,
01597     40877., 44748., 47876., 59793., 58626., 55454., 50337., 44893.,
01598     50228., 52216., 54747., 69541., 70455., 81014., 77694., 80533.,
01599     73953., 70927., 65539., 59002., 52281., 45953., 40292., 35360.,
01600     31124., 27478., 24346., 21647., 19308., 17271., 15491., 13927.,
01601     12550., 11331., 10250., 9288.8, 8431.4, 7664.9, 6978.3, 6361.8,
01602     5807.4, 5307.7, 4856.8, 4449., 4079.8, 3744.9, 3440.8, 3164.2,
01603     2912.3, 2682.7, 2473., 2281.4, 2106., 1945.3, 1797.9, 1662.5,
01604     1538.1, 1423.6, 1318.1, 1221., 1131.5, 1049., 972.99, 902.87,
01605     838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,
01606     462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,
01607     258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,
01608     147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,
01609     86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,
01610     51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,
01611     30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,
01612     18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,
01613     11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,
01614     7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,
01615     4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,
01616     2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,
01617     1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,
01618     1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,
01619     .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,
01620     .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,
```

```
01621     .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,
01622     .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,
01623     .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,
01624     .092236, .087844, .083443, .07938, .075452, .071564, .067931,
01625     .064389, .061078, .057901, .054921, .052061, .049364, .046789,
01626     .04435, .042044, .039866, .037808, .035863, .034023, .032282,
01627     .030634, .029073, .027595, .026194, .024866, .023608, .022415,
01628     .021283, .02021, .019193, .018228, .017312, .016443, .015619,
01629     .014837, .014094, .01339, .012721, .012086, .011483, .010911,
01630     .010368, .009852, .0093623, .0088972, .0084556, .0080362,
01631     .0076379, .0072596, .0069003, .006559, .0062349, .0059269,
01632     .0056344, .0053565, .0050925, .0048417, .0046034, .004377,
01633     .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,
01634     .0030785, .002928, .0027851, .0026492, .0025201, .0023975,
01635     .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,
01636     .0016938, .0016122, .0015348, .0014612, .0013913, .001325,
01637     .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,
01638     9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,
01639     7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,
01640     5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,
01641     4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,
01642     3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,
01643     3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,
01644     3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,
01645     4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,
01646     6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,
01647     8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,
01648     .001282, .0014016, .0015355, .0016764, .0018272, .0020055,
01649     .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,
01650     .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,
01651     .0052989, .0056148, .0052452, .0053357, .005333, .0045069,
01652     .0043851, .004253, .003738, .0038084, .0039013, .0041505,
01653     .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,
01654     .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,
01655     .0076269, .0063758, .006254, .0067749, .0067909, .0068231,
01656     .0072143, .0072762, .0072954, .007679, .0075107, .0073658,
01657     .0072441, .0071074, .0070378, .007176, .0072472, .0075844,
01658     .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,
01659     .013895, .015567, .016011, .016392, .016737, .017043, .017731,
01660     .018031, .018419, .018877, .019474, .019868, .020604, .021538,
01661     .022653, .023869, .025288, .026879, .028547, .030524, .03274,
01662     .035132, .03769, .040567, .043793, .047188, .049962, .053542,
01663     .057205, .060776, .061489, .064419, .067124, .065945, .068487,
01664     .071209, .074783, .077039, .082444, .08902, .09692, .10617,
01665     .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,
01666     .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,
01667     .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,
01668     .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,
01669     .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,
01670     .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,
01671     1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,
01672     2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,
01673     4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,
01674     8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,
01675     15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,
01676     28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,
01677     46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,
01678     86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,
01679     179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,
01680     397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,
01681     719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,
01682     580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,
01683     615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,
01684     316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,
01685     813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,
01686     1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,
01687     1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,
01688     1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,
01689     1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,
01690     338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.85,
01691     662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,
01692     803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01693     1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01694     523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01695     211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01696     97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01697     49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01698     26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01699     14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01700     7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01701     4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01702     2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01703     1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01704     .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01705     .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
01706     .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01707     .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
```

```
01708     .12584
01709   };
01710
01711   double xw, dw, ew, cw296, cw260, cw230, dt230, dt260, dt296, ctw, ctmpth;
01712
01713   int iw;
01714
01715   /* Get CO2 continuum absorption... */
01716   xw = nu / 2 + 1;
01717   if (xw >= 1 && xw < 2001) {
01718     iw = (int) xw;
01719     dw = xw - iw;
01720     ew = 1 - dw;
01721     cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01722     cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01723     cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01724     dt230 = t - 230;
01725     dt260 = t - 260;
01726     dt296 = t - 296;
01727     ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01728       * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01729     ctmpth = u / NA / 1000 * p / P0 * ctw;
01730   } else
01731     ctmpth = 0;
01732   return ctmpth;
01733 }
```

5.5.2.7 double ctmh2o (double *nu*, double *p*, double *t*, double *q*, double *u*)

Compute water vapor continuum (optical depth).

Definition at line 1737 of file jurassic.c.

```
01742             {
01743
01744   static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01745     .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01746     .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01747     .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01748     .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01749     .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01750     .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01751     6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01752     3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01753     1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01754     1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01755     6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01756     4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01757     3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01758     2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01759     1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01760     1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01761     1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01762     1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01763     1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01764     2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01765     4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01766     1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01767     2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01768     3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01769     3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01770     3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,
01771     2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,
01772     1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,
01773     4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,
01774     2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,
01775     1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,
01776     5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,
01777     2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,
01778     1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,
01779     7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,
01780     4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,
01781     2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,
01782     1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,
01783     1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,
01784     9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,
01785     7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,
01786     6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,
01787     5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,
01788     5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,
```

```
01789     5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,
01790     7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,
01791     1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,
01792     3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,
01793     6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,
01794     1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,
01795     1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,
01796     1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,
01797     1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,
01798     1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,
01799     3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,
01800     7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,
01801     1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,
01802     3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,
01803     7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,
01804     1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,
01805     1.387e-4, 1.363e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,
01806     1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,
01807     5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,
01808     2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,
01809     1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,
01810     5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,
01811     2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,
01812     1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,
01813     6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,
01814     3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,
01815     1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,
01816     1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,
01817     6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,
01818     4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,
01819     3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,
01820     2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,
01821     2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,
01822     2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,
01823     4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,
01824     8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,
01825     1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,
01826     3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,
01827     8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,
01828     2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,
01829     4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,
01830     8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,
01831     1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,
01832     9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,
01833     4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,
01834     1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,
01835     9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,
01836     4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,
01837     2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,
01838     1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,
01839     5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,
01840     2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,
01841     1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,
01842     8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,
01843     5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,
01844     3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,
01845     2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,
01846     2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,
01847     1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,
01848     2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,
01849     3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,
01850     5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,
01851     1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,
01852     2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,
01853     5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,
01854     1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,
01855     2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,
01856     3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,
01857     4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,
01858     5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01859     7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01860     1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01861     2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01862     4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01863     5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
01864     4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01865     2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01866     9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01867     4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01868     2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01869     1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01870     5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01871     2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,
01872     1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,
01873     7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,
01874     4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,
01875     3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,
```

```
01876     2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,
01877     2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,
01878     2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,
01879     4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,
01880     6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,
01881     9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,
01882     1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,
01883     1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,
01884     1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,
01885     2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,
01886     6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,
01887     1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,
01888     2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,
01889     2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,
01890     2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,
01891     1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,
01892     4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,
01893     2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,
01894     1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,
01895     5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,
01896     2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,
01897     1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,
01898     7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,
01899     4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,
01900     2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,
01901     1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,
01902     1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,
01903     1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,
01904     1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,
01905     2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,
01906     2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,
01907     3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,
01908     3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,
01909     6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,
01910     1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,
01911     3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,
01912     6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,
01913     1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,
01914     2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,
01915     2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,
01916     3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,
01917     7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,
01918     1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,
01919     1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,
01920     1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,
01921     5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,
01922     2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,
01923     1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,
01924     1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,
01925     1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,
01926     1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,
01927     1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,
01928     5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,
01929     2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,
01930     1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,
01931     5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,
01932     3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,
01933     1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,
01934     9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,
01935     7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,
01936     8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,
01937     1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,
01938     2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,
01939     3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,
01940     6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,
01941     1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,
01942     2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,
01943     4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,
01944     7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,
01945     7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,
01946     5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,
01947     2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,
01948     1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,
01949     8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,
01950     7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,
01951     7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,
01952     6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,
01953     3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,
01954     1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,
01955     7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,
01956     3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,
01957     1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,
01958     1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,
01959     6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,
01960     4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,
01961     4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,
01962     7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,
```

```
01963     1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,
01964     3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,
01965     7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,
01966     1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,
01967     3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,
01968     7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,
01969     1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,
01970     2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,
01971     4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,
01972     5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,
01973     5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,
01974     3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,
01975     1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,
01976     8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,
01977     5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,
01978     5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,
01979     8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,
01980     1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,
01981     1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,
01982     7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,
01983     3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,
01984     1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,
01985     7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,
01986     4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,
01987     2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,
01988     1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,
01989     6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,
01990     6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,
01991     8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,
01992     1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,
01993     3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,
01994     6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,
01995     1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,
01996     2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,
01997     3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,
01998     4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,
01999     3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,
02000     1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,
02001     7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,
02002     4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,
02003     3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,
02004     4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,
02005     7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,
02006     8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,
02007     6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,
02008     3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,
02009     1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,
02010     6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,
02011     3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,
02012     1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,
02013     1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,
02014     1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,
02015     2.534e-12, 2.858e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,
02016     5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,
02017     1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,
02018     2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,
02019     5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,
02020     1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,
02021     2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,
02022     4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,
02023     6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,
02024     5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,
02025     3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,
02026     1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,
02027     7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,
02028     4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,
02029     2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,
02030     2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,
02031     4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,
02032     6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,
02033     8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,
02034     8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,
02035     5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,
02036     2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,
02037     1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,
02038     5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,
02039     2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,
02040     1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,
02041     1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,
02042     1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,
02043     2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,
02044     5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,
02045     1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,
02046     2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,
02047     3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,
02048     4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,
02049     3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,
```

```
02050     2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,
02051     1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,
02052     5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,
02053     2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,
02054     1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,
02055     1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,
02056     1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,
02057     2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,
02058     3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,
02059     3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,
02060     2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,
02061     1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,
02062     5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,
02063     4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,
02064     5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,
02065     9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,
02066     2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,
02067     4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,
02068     1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,
02069     2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,
02070     4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,
02071     7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,
02072     7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,
02073     5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,
02074     2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02075     1.093e-11, 9.558e-12
02076   };
02077
02078   static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02079     .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02080     .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02081     .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02082     .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02083     .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02084     .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02085     .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02086     5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02087     3.366e-4, 3.085e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02088     2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02089     1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02090     9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02091     6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02092     4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02093     3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02094     2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02095     2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02096     2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02097     2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02098     3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02099     7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02100     1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02101     3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02102     5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02103     6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02104     5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02105     2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02106     1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02107     6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02108     2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02109     1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02110     7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02111     4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02112     2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02113     1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02114     9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02115     6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02116     4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02117     3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02118     2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
02119     1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,
02120     1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,
02121     1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,
02122     1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,
02123     1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,
02124     1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,
02125     2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,
02126     5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,
02127     1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,
02128     1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,
02129     2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,
02130     2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,
02131     2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,
02132     2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,
02133     5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,
02134     1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,
02135     2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,
02136     5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,
```

```
02137     1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,
02138     1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,
02139     2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,
02140     1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,
02141     7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,
02142     2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,
02143     1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,
02144     7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,
02145     3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,
02146     1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,
02147     1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,
02148     5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,
02149     3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,
02150     2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,
02151     1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,
02152     9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,
02153     7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,
02154     5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,
02155     5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,
02156     6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,
02157     9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,
02158     1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,
02159     3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,
02160     6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,
02161     1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,
02162     2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,
02163     6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,
02164     1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,
02165     1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,
02166     1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,
02167     7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,
02168     3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,
02169     1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,
02170     7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,
02171     3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,
02172     1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,
02173     9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,
02174     5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,
02175     3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,
02176     1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,
02177     1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,
02178     9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,
02179     6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,
02180     4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,
02181     4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,
02182     4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,
02183     5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,
02184     9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,
02185     1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,
02186     3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,
02187     7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,
02188     1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,
02189     3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,
02190     5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,
02191     7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,
02192     7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,
02193     8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,
02194     1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,
02195     3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,
02196     7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,
02197     8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,
02198     6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,
02199     2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,
02200     1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,
02201     6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,
02202     3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,
02203     1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,
02204     8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,
02205     4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,
02206     2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,
02207     1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,
02208     9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,
02209     6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,
02210     5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,
02211     4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,
02212     4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,
02213     6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,
02214     1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,
02215     1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,
02216     1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,
02217     1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,
02218     2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,
02219     3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,
02220     8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,
02221     2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,
02222     3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,
02223     4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,
```

```
02224     3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,
02225     1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,
02226     6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,
02227     3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,
02228     1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,
02229     9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,
02230     4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,
02231     2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,
02232     1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,
02233     7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,
02234     5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,
02235     3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,
02236     2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,
02237     2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,
02238     2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,
02239     3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,
02240     4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,
02241     5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,
02242     6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,
02243     1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,
02244     2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,
02245     4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,
02246     9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,
02247     2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,
02248     3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,
02249     3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,
02250     5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,
02251     1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,
02252     1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,
02253     2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,
02254     1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,
02255     6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,
02256     3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,
02257     2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,
02258     2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,
02259     2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,
02260     2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,
02261     1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,
02262     6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,
02263     3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,
02264     1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,
02265     8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,
02266     4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,
02267     2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,
02268     1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,
02269     1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,
02270     1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,
02271     2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,
02272     4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,
02273     6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,
02274     9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,
02275     1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,
02276     3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,
02277     7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,
02278     1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,
02279     1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,
02280     7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,
02281     2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,
02282     1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,
02283     1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,
02284     1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,
02285     1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,
02286     9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,
02287     5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,
02288     2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,
02289     1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,
02290     5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,
02291     3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,
02292     1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,
02293     1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,
02294     8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,
02295     9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,
02296     1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,
02297     3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,
02298     5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,
02299     1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,
02300     2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,
02301     5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,
02302     1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,
02303     2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,
02304     3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,
02305     7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,
02306     9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,
02307     8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,
02308     4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,
02309     2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,
02310     1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,
```

```
02311     8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,
02312     9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,
02313     1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,
02314     1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,
02315     1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,
02316     9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,
02317     4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,
02318     2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,
02319     1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,
02320     5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,
02321     3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,
02322     1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,
02323     1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,
02324     1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,
02325     1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,
02326     2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,
02327     5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,
02328     8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,
02329     1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,
02330     4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,
02331     6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,
02332     7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,
02333     5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,
02334     2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,
02335     1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,
02336     6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,
02337     5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,
02338     8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,
02339     1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,
02340     1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,
02341     8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,
02342     3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,
02343     1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,
02344     9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,
02345     5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,
02346     2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,
02347     2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,
02348     2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,
02349     4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,
02350     9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,
02351     1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,
02352     3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,
02353     8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,
02354     2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,
02355     4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,
02356     7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,
02357     9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,
02358     8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,
02359     5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,
02360     2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,
02361     1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,
02362     5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,
02363     3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,
02364     4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,
02365     6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,
02366     1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,
02367     1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,
02368     1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,
02369     7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,
02370     3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,
02371     1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,
02372     8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,
02373     4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,
02374     2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,
02375     2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,
02376     2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,
02377     5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,
02378     1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,
02379     2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,
02380     4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02381     7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02382     9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02383     8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02384     5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02385     2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
02386     9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02387     5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02388     2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02389     2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02390     3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02391     6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02392     9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02393     9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
02394     5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02395     2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02396     1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02397     1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
```

```
02398     1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02399     3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02400     7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02401     1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02402     3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
02403     8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02404     1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02405     2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02406     2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02407     1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02408     7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02409     3.511e-11
02410   };
02411
02412   static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02413     .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02414     .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02415     .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02416     4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02417     1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02418     3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02419     1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02420     4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02421     1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,
02422     7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,
02423     2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,
02424     1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,
02425     5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,
02426     2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,
02427     1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,
02428     5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,
02429     2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,
02430     3.413e-9, 3.738e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,
02431     8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,
02432     3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,
02433     1.889e-7, 2.589e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,
02434     1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,
02435     7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,
02436     2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,
02437     4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,
02438     2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,
02439     3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,
02440     2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,
02441     8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,
02442     2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,
02443     7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,
02444     2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,
02445     7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,
02446     2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,
02447     8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,
02448     3.157e-9, 2.725e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,
02449     1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,
02450     5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,
02451     2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,
02452     1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,
02453     5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,
02454     1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,
02455     9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,
02456     1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,
02457     1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,
02458     1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,
02459     6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,
02460     1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,
02461     1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,
02462     7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,
02463     1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,
02464     7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,
02465     1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,
02466     7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,
02467     4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,
02468     5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,
02469     1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,
02470     4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,
02471     1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,
02472     7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,
02473     1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,
02474     1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,
02475     1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,
02476     6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,
02477     8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,
02478     1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,
02479     4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,
02480     1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,
02481     6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,
02482     2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,
02483     9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,
02484     3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,
```

```
02485     1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,
02486     9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,
02487     2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,
02488     3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,
02489     2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,
02490     2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,
02491     1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,
02492     1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,
02493     1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,
02494     4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,
02495     1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,
02496     7.611e-9, 9.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,
02497     4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,
02498     3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,
02499     7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,
02500     8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,
02501     1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,
02502     7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,
02503     1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,
02504     2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,
02505     6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,
02506     2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,
02507     8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,
02508     3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,
02509     1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,
02510     5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,
02511     2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,
02512     2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,
02513     1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,
02514     6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,
02515     6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,
02516     6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,
02517     3.311e-12, 3.346e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,
02518     4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,
02519     1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,
02520     4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,
02521     2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,
02522     1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,
02523     4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,
02524     1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,
02525     2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,
02526     3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,
02527     4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,
02528     3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,
02529     7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,
02530     2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,
02531     4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,
02532     4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,
02533     5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,
02534     2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,
02535     3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,
02536     7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,
02537     3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,
02538     1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,
02539     4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,
02540     1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,
02541     5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,
02542     2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,
02543     9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,
02544     4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,
02545     3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,
02546     1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,
02547     9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,
02548     2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,
02549     4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,
02550     6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,
02551     6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,
02552     9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,
02553     5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,
02554     4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02555     2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02556     1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02557     1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02558     2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02559     2.859e-8, 2.839e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
02560     1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02561     2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02562     8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02563     3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02564     1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02565     6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02566     1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02567     6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
02568     2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,
02569     7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,
02570     2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,
02571     1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,
```

```
02572     4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,
02573     4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,
02574     1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,
02575     1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,
02576     2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,
02577     1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,
02578     1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,
02579     1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,
02580     5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,
02581     3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,
02582     9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,
02583     1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,
02584     1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,
02585     2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,
02586     5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,
02587     9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,
02588     1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,
02589     1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,
02590     6.403e-9, 4.643e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,
02591     9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,
02592     4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,
02593     9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,
02594     1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,
02595     1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,
02596     1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,
02597     6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,
02598     7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,
02599     1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,
02600     5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,
02601     1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,
02602     5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,
02603     2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,
02604     1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,
02605     8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,
02606     5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,
02607     1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,
02608     1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,
02609     2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,
02610     3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,
02611     1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,
02612     4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,
02613     5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,
02614     7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,
02615     7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,
02616     2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,
02617     5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,
02618     3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,
02619     5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,
02620     5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,
02621     5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,
02622     5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,
02623     1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,
02624     1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,
02625     3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,
02626     1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,
02627     3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,
02628     1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,
02629     6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,
02630     3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,
02631     2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,
02632     1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,
02633     3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,
02634     5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,
02635     1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,
02636     1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,
02637     4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,
02638     6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,
02639     1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,
02640     4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,
02641     5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,
02642     5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,
02643     5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,
02644     1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,
02645     2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,
02646     2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,
02647     3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,
02648     6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,
02649     9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,
02650     8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,
02651     1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,
02652     2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,
02653     4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,
02654     2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,
02655     1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,
02656     4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,
02657     5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,
02658     1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,
```

```
02659     9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,
02660     5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,
02661     4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,
02662     1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,
02663     1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,
02664     7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,
02665     2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,
02666     3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,
02667     3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,
02668     4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,
02669     1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,
02670     4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,
02671     1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,
02672     1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,
02673     3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,
02674     5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,
02675     6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,
02676     8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,
02677     4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,
02678     4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,
02679     5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,
02680     1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,
02681     5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,
02682     2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,
02683     1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,
02684     8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,
02685     2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,
02686     1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,
02687     1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,
02688     1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,
02689     6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,
02690     2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,
02691     4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,
02692     5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,
02693     5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,
02694     5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,
02695     1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,
02696     2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,
02697     6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,
02698     5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,
02699     9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,
02700     2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,
02701     5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,
02702     7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,
02703     7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,
02704     8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,
02705     1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,
02706     2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,
02707     9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,
02708     2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,
02709     4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,
02710     2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,
02711     2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,
02712     3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,
02713     1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,
02714     6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,
02715     1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,
02716     3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,
02717     3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,
02718     3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,
02719     3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,
02720     1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,
02721     1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,
02722     3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,
02723     9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,
02724     1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,
02725     8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,
02726     2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,
02727     3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,
02728     3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02729     3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02730     5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02731     4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02732     1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02733     6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
02734     9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02735     1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02736     1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02737     3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02738     1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02739     4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02740     6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02741     6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
02742     7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02743     2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02744     4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02745   };
```

```
02746
02747   static double xfcrev[15] =
02748     { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02749     1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02750   };
02751
02752   double a1, a2, a3, dw, ew, dx, xw, xx, vf2, vf6, cw260, cw296,
02753     sfac, fscal, cwfrn, ctmpth, ctwfrn, ctwslf;
02754
02755   int iw, ix;
02756
02757   /* Get H2O continuum absorption... */
02758   xw = nu / 10 + 1;
02759   if (xw >= 1 && xw < 2001) {
02760     iw = (int) xw;
02761     dw = xw - iw;
02762     ew = 1 - dw;
02763     cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02764     cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02765     cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02766     if (nu <= 820 || nu >= 960) {
02767       sfac = 1;
02768     } else {
02769       xx = (nu - 820) / 10;
02770       ix = (int) xx;
02771       dx = xx - ix;
02772       sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];
02773     }
02774     ctwslf = sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02775     vf2 = POW2(nu - 370);
02776     vf6 = POW3(vf2);
02777     fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02778     ctwfrn = cwfrn * fscal;
02779     a1 = nu * u * tanh(.7193876 / t * nu);
02780     a2 = 296 / t;
02781     a3 = p / P0 * (q * ctwslf + (1 - q) * ctwfrn) * 1e-20;
02782     ctmpth = a1 * a2 * a3;
02783   } else
02784     ctmpth = 0;
02785   return ctmpth;
02786 }
```

5.5.2.8 double ctmn2 (double *nu*, double *p*, double *t*)

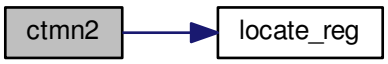
Compute nitrogen continuum (absorption coefficient).

Definition at line 2790 of file jurassic.c.

```
02793           {
02794
02795   static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02796     1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02797     2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02798     5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02799     7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02800     9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02801     1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02802     1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02803     1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02804     1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02805     7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,
02806     3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02807     1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02808     7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02809   };
02810
02811   static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02812     511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02813     233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02814     -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02815     -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02816     -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02817     -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02818     121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02819     133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,
02820     372., 449., 514., 569., 609., 642., 673., 673.
02821   };
02822
02823   static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02824     2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
```

```
02825     2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02826     2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02827     2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02828     2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02829     2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
02830     2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02831     2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02832     2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02833     2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02834   };
02835
02836   double b, beta, q_n2 = 0.79, t0 = 273, tr = 296;
02837
02838   int idx;
02839
02840   /* Check wavenumber range... */
02841   if (nu < nua[0] || nu > nua[97])
02842     return 0;
02843
02844   /* Interpolate B and beta... */
02845   idx = locate_reg(nua, 98, nu);
02846   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02847   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02848
02849   /* Compute absorption coefficient... */
02850   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02851     * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02852 }
```

Here is the call graph for this function:



5.5.2.9 double ctmo2 (double *nu*, double *p*, double *t*)

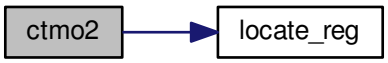
Compute oxygen continuum (absorption coefficient).

Definition at line 2856 of file jurassic.c.

```
02859             {
02860
02861   static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,
02862     .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,
02863     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02864     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02865     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02866     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02867     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02868     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02869     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02870     .071, .064, 0.
02871   };
02872
02873   static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02874     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02875     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02876     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,
02877     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02878     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02879     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02880     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02881     346., 322., 291., 290., 350., 371., 504., 504.
02882   };
02883
```

```
02884   static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02885     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02886     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02887     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02888     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
02889     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02890     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02891     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02892     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02893     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02894     1800., 1805.
02895   };
02896
02897   double b, beta, q_o2 = 0.21, t0 = 273, tr = 296;
02898
02899   int idx;
02900
02901   /* Check wavenumber range... */
02902   if (nu < nua[0] || nu > nua[89])
02903     return 0;
02904
02905   /* Interpolate B and beta... */
02906   idx = locate_reg(nua, 90, nu);
02907   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02908   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02909
02910   /* Compute absorption coefficient... */
02911   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02912     b;
02913 }
```

Here is the call graph for this function:



5.5.2.10 void copy_atm (ctl_t * *ctl*, atm_t * *atm_dest*, atm_t * *atm_src*, int *init*)

Copy and initialize atmospheric data.

Definition at line 2917 of file jurassic.c.

```
02921             {
02922
02923   int ig, ip, iw;
02924
02925   size_t s;
02926
02927   /* Data size... */
02928   s = (size_t) atm_src->np * sizeof(double);
02929
02930   /* Copy data... */
02931   atm_dest->np = atm_src->np;
02932   memcpy(atm_dest->time, atm_src->time, s);
02933   memcpy(atm_dest->z, atm_src->z, s);
02934   memcpy(atm_dest->lon, atm_src->lon, s);
02935   memcpy(atm_dest->lat, atm_src->lat, s);
02936   memcpy(atm_dest->p, atm_src->p, s);
02937   memcpy(atm_dest->t, atm_src->t, s);
02938   for (ig = 0; ig < ctl->ng; ig++)
02939     memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02940   for (iw = 0; iw < ctl->nw; iw++)
02941     memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02942
02943   /* Initialize... */
```

```
02944   if (init)
02945     for (ip = 0; ip < atm_dest->np; ip++) {
02946       atm_dest->p[ip] = 0;
02947       atm_dest->t[ip] = 0;
02948       for (ig = 0; ig < ctl->ng; ig++)
02949         atm_dest->q[ig][ip] = 0;
02950       for (iw = 0; iw < ctl->nw; iw++)
02951         atm_dest->k[iw][ip] = 0;
02952     }
02953 }
```

5.5.2.11 void copy_obs (ctl_t * *ctl*, obs_t * *obs_dest*, obs_t * *obs_src*, int *init*)

Copy and initialize observation data.

Definition at line 2957 of file jurassic.c.

```
02961             {
02962
02963   int id, ir;
02964
02965   size_t s;
02966
02967   /* Data size... */
02968   s = (size_t) obs_src->nr * sizeof(double);
02969
02970   /* Copy data... */
02971   obs_dest->nr = obs_src->nr;
02972   memcpy(obs_dest->time, obs_src->time, s);
02973   memcpy(obs_dest->obsz, obs_src->obsz, s);
02974   memcpy(obs_dest->obslon, obs_src->obslon, s);
02975   memcpy(obs_dest->obslat, obs_src->obslat, s);
02976   memcpy(obs_dest->vpz, obs_src->vpz, s);
02977   memcpy(obs_dest->vplon, obs_src->vplon, s);
02978   memcpy(obs_dest->vplat, obs_src->vplat, s);
02979   memcpy(obs_dest->tpz, obs_src->tpz, s);
02980   memcpy(obs_dest->tplon, obs_src->tplon, s);
02981   memcpy(obs_dest->tplat, obs_src->tplat, s);
02982   for (id = 0; id < ctl->nd; id++)
02983     memcpy(obs_dest->rad[id], obs_src->rad[id], s);
02984   for (id = 0; id < ctl->nd; id++)
02985     memcpy(obs_dest->tau[id], obs_src->tau[id], s);
02986
02987   /* Initialize... */
02988   if (init)
02989     for (id = 0; id < ctl->nd; id++)
02990       for (ir = 0; ir < obs_dest->nr; ir++)
02991         if (gsl_finite(obs_dest->rad[id][ir])) {
02992           obs_dest->rad[id][ir] = 0;
02993           obs_dest->tau[id][ir] = 0;
02994         }
02995 }
```

5.5.2.12 int find_emitter (ctl_t * *ctl*, const char * *emitter*)

Find index of an emitter.

Definition at line 2999 of file jurassic.c.

```
03001                         {
03002
03003   int ig;
03004
03005   for (ig = 0; ig < ctl->ng; ig++)
03006     if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03007       return ig;
03008
03009   return -1;
03010 }
```

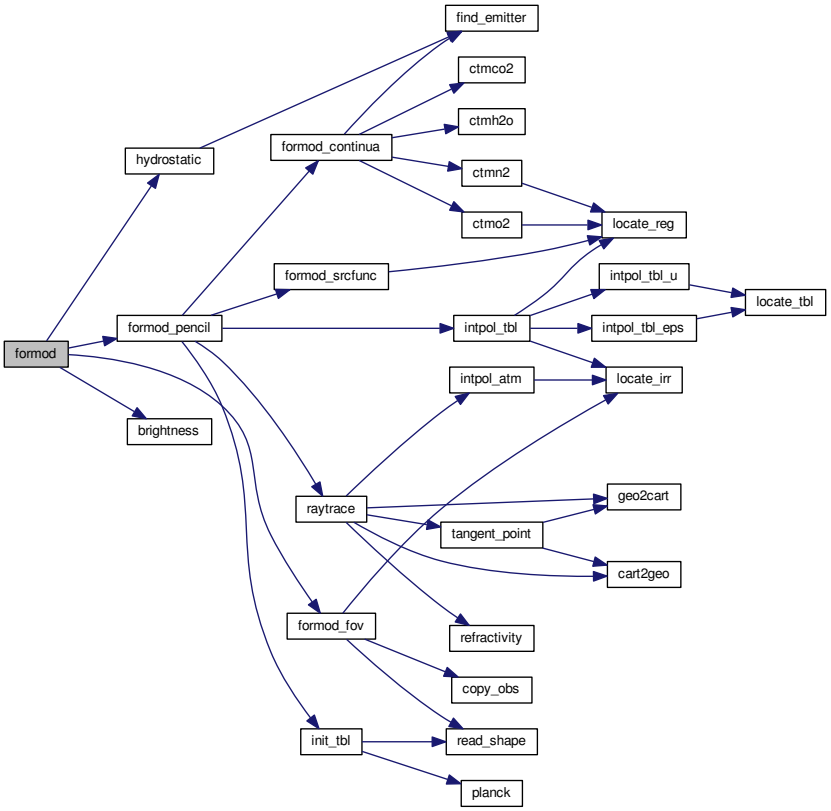
5.5.2.13 void formod (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*)

Determine ray paths and compute radiative transfer.

Definition at line 3014 of file jurassic.c.

```
03017                {
03018
03019   int id, ir, *mask;
03020
03021   /* Allocate... */
03022   ALLOC(mask, int,
03023         ND * NR);
03024
03025   /* Save observation mask... */
03026   for (id = 0; id < ctl->nd; id++)
03027     for (ir = 0; ir < obs->nr; ir++)
03028       mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03029
03030   /* Hydrostatic equilibrium... */
03031   hydrostatic(ctl, atm);
03032
03033   /* Calculate pencil beams... */
03034   for (ir = 0; ir < obs->nr; ir++)
03035     formod_pencil(ctl, atm, obs, ir);
03036
03037   /* Apply field-of-view convolution... */
03038   formod_fov(ctl, obs);
03039
03040   /* Convert radiance to brightness temperature... */
03041   if (ctl->write_bbt)
03042     for (id = 0; id < ctl->nd; id++)
03043       for (ir = 0; ir < obs->nr; ir++)
03044         obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03045
03046   /* Apply observation mask... */
03047   for (id = 0; id < ctl->nd; id++)
03048     for (ir = 0; ir < obs->nr; ir++)
03049       if (mask[id * NR + ir])
03050         obs->rad[id][ir] = GSL_NAN;
03051
03052   /* Free... */
03053   free(mask);
03054 }
```

Here is the call graph for this function:



5.5.2.14 void formod_continua (ctl_t * *ctl*, los_t * *los*, int *ip*, double * *beta*)

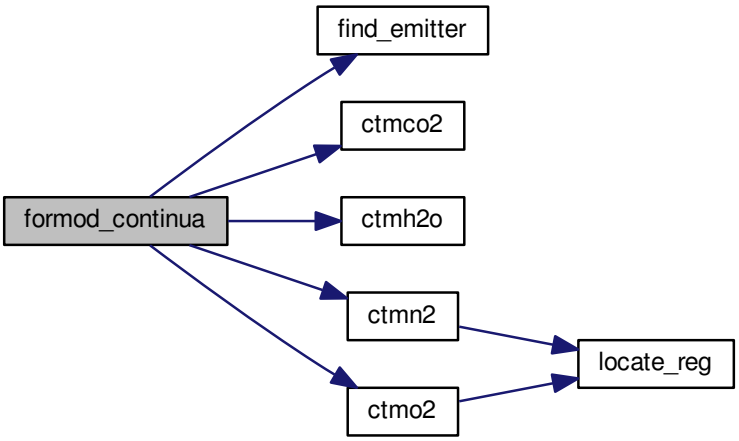
Compute absorption coefficient of continua.

Definition at line 3058 of file jurassic.c.

```
03062                {
03063
03064   static int ig_co2 = -999, ig_h2o = -999;
03065
03066   int id;
03067
03068   /* Extinction... */
03069   for (id = 0; id < ctl->nd; id++)
03070     beta[id] = los->k[ctl->window[id]][ip];
03071
03072   /* CO2 continuum... */
03073   if (ctl->ctm_co2) {
03074     if (ig_co2 == -999)
03075       ig_co2 = find_emitter(ctl, "CO2");
03076     if (ig_co2 >= 0)
03077       for (id = 0; id < ctl->nd; id++)
03078         beta[id] += ctmco2(ctl->nu[id], los->p[ip], los->t[ip],
03079                            los->u[ig_co2][ip]) / los->ds[ip];
03080   }
03081
03082   /* H2O continuum... */
03083   if (ctl->ctm_h2o) {
03084     if (ig_h2o == -999)
03085       ig_h2o = find_emitter(ctl, "H2O");
03086     if (ig_h2o >= 0)
03087       for (id = 0; id < ctl->nd; id++)
```

```
03088         beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03089                            los->q[ig_h2o][ip],
03090                            los->u[ig_h2o][ip]) / los->ds[ip];
03091     }
03092
03093     /* N2 continuum... */
03094     if (ctl->ctm_n2)
03095       for (id = 0; id < ctl->nd; id++)
03096         beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03097
03098     /* O2 continuum... */
03099     if (ctl->ctm_o2)
03100       for (id = 0; id < ctl->nd; id++)
03101         beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03102   }
```

Here is the call graph for this function:



5.5.2.15 void formod_fov (ctl_t * *ctl*, obs_t * *obs*)

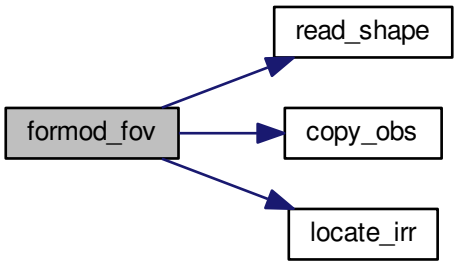
Apply field of view convolution.

Definition at line 3106 of file jurassic.c.

```
03108                 {
03109
03110     static double dz[NSHAPE], w[NSHAPE];
03111
03112     static int init = 0, n;
03113
03114     obs_t *obs2;
03115
03116     double rad[ND][NR], tau[ND][NR], wsum, z[NR], zfov;
03117
03118     int i, id, idx, ir, ir2, nz;
03119
03120     /* Do not take into account FOV... */
03121     if (ctl->fov[0] == '-')
03122       return;
03123
03124     /* Initialize FOV data... */
03125     if (!init) {
03126       init = 1;
03127       read_shape(ctl->fov, dz, w, &n);
03128     }
```

```
03129
03130   /* Allocate... */
03131   ALLOC(obs2, obs_t, 1);
03132
03133   /* Copy observation data... */
03134   copy_obs(ctl, obs2, obs, 0);
03135
03136   /* Loop over ray paths... */
03137   for (ir = 0; ir < obs->nr; ir++) {
03138
03139     /* Get radiance and transmittance profiles... */
03140     nz = 0;
03141     for (ir2 = GSL_MAX(ir - NFOV, 0); ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr);
03142          ir2++)
03143       if (obs->time[ir2] == obs->time[ir]) {
03144         z[nz] = obs2->vpz[ir2];
03145         for (id = 0; id < ctl->nd; id++) {
03146           rad[id][nz] = obs2->rad[id][ir2];
03147           tau[id][nz] = obs2->tau[id][ir2];
03148         }
03149         nz++;
03150       }
03151     if (nz < 2)
03152       ERRMSG("Cannot apply FOV convolution!");
03153
03154     /* Convolute profiles with FOV... */
03155     wsum = 0;
03156     for (id = 0; id < ctl->nd; id++) {
03157       obs->rad[id][ir] = 0;
03158       obs->tau[id][ir] = 0;
03159     }
03160     for (i = 0; i < n; i++) {
03161       zfov = obs->vpz[ir] + dz[i];
03162       idx = locate_irr(z, nz, zfov);
03163       for (id = 0; id < ctl->nd; id++) {
03164         obs->rad[id][ir] += w[i]
03165           * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
03166         obs->tau[id][ir] += w[i]
03167           * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
03168       }
03169       wsum += w[i];
03170     }
03171     for (id = 0; id < ctl->nd; id++) {
03172       obs->rad[id][ir] /= wsum;
03173       obs->tau[id][ir] /= wsum;
03174     }
03175   }
03176
03177   /* Free... */
03178   free(obs2);
03179 }
```

Here is the call graph for this function:



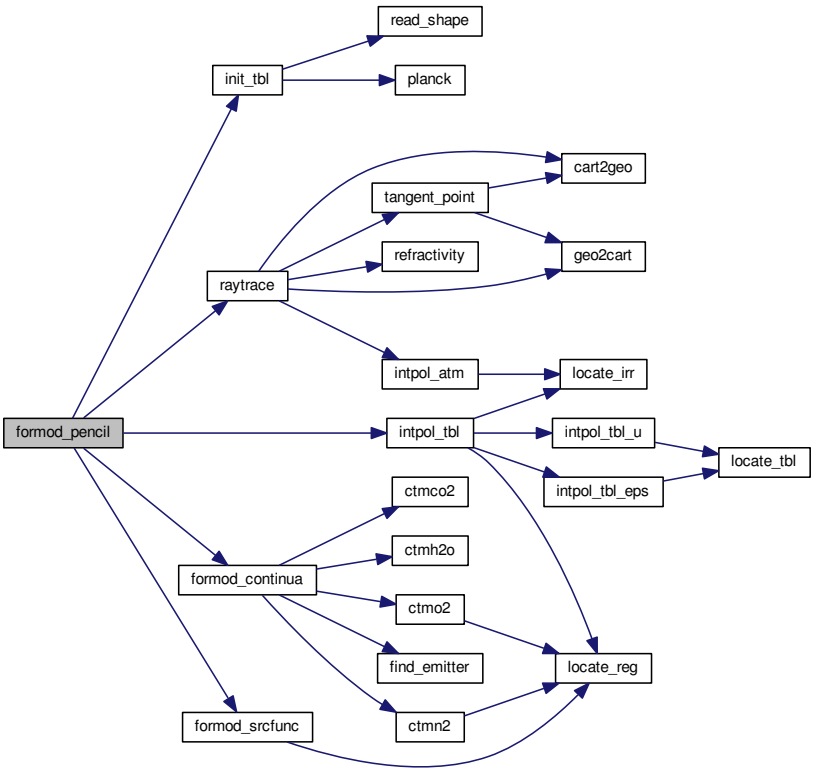
5.5.2.16 void formod_pencil (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, int *ir*)

Compute radiative transfer for a pencil beam.

Definition at line 3183 of file jurassic.c.

```
03187           {
03188
03189     static tbl_t *tbl;
03190
03191     static int init = 0;
03192
03193     los_t *los;
03194
03195     double beta_ctm[ND], eps, src_planck[ND], tau_path[NG][ND], tau_gas[ND];
03196
03197     int id, ip;
03198
03199     /* Initialize look-up tables... */
03200     if (!init) {
03201       init = 1;
03202       ALLOC(tbl, tbl_t, 1);
03203       init_tbl(ctl, tbl);
03204     }
03205
03206     /* Allocate... */
03207     ALLOC(los, los_t, 1);
03208
03209     /* Initialize... */
03210     for (id = 0; id < ctl->nd; id++) {
03211       obs->rad[id][ir] = 0;
03212       obs->tau[id][ir] = 1;
03213     }
03214
03215     /* Raytracing... */
03216     raytrace(ctl, atm, obs, los, ir);
03217
03218     /* Loop over LOS points... */
03219     for (ip = 0; ip < los->np; ip++) {
03220
03221       /* Get trace gas transmittance... */
03222       intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03223
03224       /* Get continuum absorption... */
03225       formod_continua(ctl, los, ip, beta_ctm);
03226
03227       /* Compute Planck function... */
03228       formod_srcfunc(ctl, tbl, los->t[ip], src_planck);
03229
03230       /* Loop over channels... */
03231       for (id = 0; id < ctl->nd; id++)
03232         if (tau_gas[id] > 0) {
03233
03234           /* Get segment emissivity... */
03235           eps = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03236
03237           /* Compute radiance... */
03238           obs->rad[id][ir] += src_planck[id] * eps * obs->tau[id][ir];
03239
03240           /* Compute path transmittance... */
03241           obs->tau[id][ir] *= (1 - eps);
03242         }
03243     }
03244
03245     /* Add surface... */
03246     if (los->tsurf > 0) {
03247       formod_srcfunc(ctl, tbl, los->tsurf, src_planck);
03248       for (id = 0; id < ctl->nd; id++)
03249         obs->rad[id][ir] += src_planck[id] * obs->tau[id][ir];
03250     }
03251
03252     /* Free... */
03253     free(los);
03254   }
```

Here is the call graph for this function:



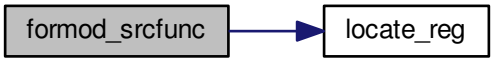
5.5.2.17 void formod_srcfunc (ctl_t * *ctl*, tbl_t * *tbl*, double *t*, double * *src*)

Compute Planck source function.

Definition at line 3258 of file jurassic.c.

```
03262                {
03263
03264   int id, it;
03265
03266   /* Determine index in temperature array... */
03267   it = locate_reg(tbl->st, TBLNS, t);
03268
03269   /* Interpolate Planck function value... */
03270   for (id = 0; id < ctl->nd; id++)
03271     src[id] = LIN(tbl->st[it], tbl->sr[id][it],
03272                   tbl->st[it + 1], tbl->sr[id][it + 1], t);
03273 }
```

Here is the call graph for this function:



5.5.2.18 void geo2cart (double *z*, double *lon*, double *lat*, double * *x*)

Convert geolocation to Cartesian coordinates.

Definition at line 3277 of file jurassic.c.

```
03281             {
03282
03283   double radius;
03284
03285   radius = z + RE;
03286   x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);
03287   x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);
03288   x[2] = radius * sin(lat / 180 * M_PI);
03289 }
```

5.5.2.19 void hydrostatic (ctl_t * *ctl*, atm_t * *atm*)

Set hydrostatic equilibrium.

Definition at line 3293 of file jurassic.c.

```
03295               {
03296
03297   static int ig_h2o = -999;
03298
03299   double dzmin = 1e99, e = 0, mean, mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03300
03301   int i, ip, ipref = 0, ipts = 20;
03302
03303   /* Check reference height... */
03304   if (ctl->hydz < 0)
03305     return;
03306
03307   /* Determine emitter index of H2O... */
03308   if (ig_h2o == -999)
03309     ig_h2o = find_emitter(ctl, "H2O");
03310
03311   /* Find air parcel next to reference height... */
03312   for (ip = 0; ip < atm->np; ip++)
03313     if (fabs(atm->z[ip] - ctl->hydz) < dzmin) {
03314       dzmin = fabs(atm->z[ip] - ctl->hydz);
03315       ipref = ip;
03316     }
03317
03318   /* Upper part of profile... */
03319   for (ip = ipref + 1; ip < atm->np; ip++) {
03320     mean = 0;
03321     for (i = 0; i < ipts; i++) {
03322       if (ig_h2o >= 0)
03323         e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03324                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03325       mean += (e * mmh2o + (1 - e) * mmair)
03326         * G0 / RI
03327         / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03328     }
03329
03330     /* Compute p(z,T)... */
03331     atm->p[ip] =
03332       exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03333   }
03334
03335   /* Lower part of profile... */
03336   for (ip = ipref - 1; ip >= 0; ip--) {
03337     mean = 0;
03338     for (i = 0; i < ipts; i++) {
03339       if (ig_h2o >= 0)
03340         e = LIN(0.0, atm->q[ig_h2o][ip + 1],
03341                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03342       mean += (e * mmh2o + (1 - e) * mmair)
03343         * G0 / RI
03344         / LIN(0.0, atm->t[ip + 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03345     }
03346
03347     /* Compute p(z,T)... */
03348     atm->p[ip] =
03349       exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03350   }
03351 }
```

Here is the call graph for this function:

5.5.2.20 void idx2name (ctl_t * *ctl*, int *idx*, char * *quantity*)

Determine name of state vector quantity for given index.

Definition at line 3355 of file jurassic.c.

```
03358                   {
03359
03360   int ig, iw;
03361
03362   if (idx == IDXP)
03363     sprintf(quantity, "PRESSURE");
03364
03365   if (idx == IDXT)
03366     sprintf(quantity, "TEMPERATURE");
03367
03368   for (ig = 0; ig < ctl->ng; ig++)
03369     if (idx == IDXQ(ig))
03370       sprintf(quantity, "%s", ctl->emitter[ig]);
03371
03372   for (iw = 0; iw < ctl->nw; iw++)
03373     if (idx == IDXK(iw))
03374       sprintf(quantity, "EXTINCT_WINDOW%d", iw);
03375 }
```

5.5.2.21 void init_tbl (ctl_t * *ctl*, tbl_t * *tbl*)

Initialize look-up tables.

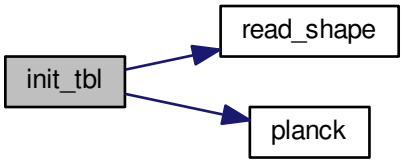
Definition at line 3379 of file jurassic.c.

```
03381                   {
03382
03383   FILE *in;
03384
03385   char filename[2 * LEN], line[LEN];
03386
03387   double eps, eps_old, press, press_old, temp, temp_old, u, u_old,
03388     f[NSHAPE], fsum, nu[NSHAPE];
03389
03390   int i, id, ig, ip, it, n;
03391
03392   /* Loop over trace gases and channels... */
03393   for (ig = 0; ig < ctl->ng; ig++)
03394 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(in,filename,line,eps,eps_old,press,
      press_old,temp,temp_old,u,u_old,id,ip,it)
03395     for (id = 0; id < ctl->nd; id++) {
03396
03397       /* Initialize... */
03398       tbl->np[ig][id] = -1;
03399       eps_old = -999;
03400       press_old = -999;
03401       temp_old = -999;
03402       u_old = -999;
03403
03404       /* Try to open file... */
```

```
03405       sprintf(filename, "%s_%.4f_%s.tab",
03406               ctl->tblbase, ctl->nu[id], ctl->emitter[ig]);
03407       if (!(in = fopen(filename, "r"))) {
03408         printf("Missing emissivity table: %s\n", filename);
03409         continue;
03410       }
03411       printf("Read emissivity table: %s\n", filename);
03412
03413       /* Read data... */
03414       while (fgets(line, LEN, in)) {
03415
03416         /* Parse line... */
03417         if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
03418           continue;
03419
03420         /* Determine pressure index... */
03421         if (press != press_old) {
03422           press_old = press;
03423           if ((++tbl->np[ig][id]) >= TBLNP)
03424             ERRMSG("Too many pressure levels!");
03425           tbl->nt[ig][id][tbl->np[ig][id]] = -1;
03426         }
03427
03428         /* Determine temperature index... */
03429         if (temp != temp_old) {
03430           temp_old = temp;
03431           if ((++tbl->nt[ig][id][tbl->np[ig][id]]) >= TBLNT)
03432             ERRMSG("Too many temperatures!");
03433           tbl->nu[ig][id][tbl->np[ig][id]]
03434             [tbl->nt[ig][id][tbl->np[ig][id]]] = -1;
03435         }
03436
03437         /* Determine column density index... */
03438         if ((eps > eps_old && u > u_old) || tbl->nu[ig][id][tbl->np[ig][id]]
03439             [tbl->nt[ig][id][tbl->np[ig][id]]] < 0) {
03440           eps_old = eps;
03441           u_old = u;
03442           if ((++tbl->nu[ig][id][tbl->np[ig][id]]
03443                [tbl->nt[ig][id][tbl->np[ig][id]]]) >= TBLNU) {
03444             tbl->nu[ig][id][tbl->np[ig][id]]
03445               [tbl->nt[ig][id][tbl->np[ig][id]]]--;
03446             continue;
03447           }
03448         }
03449
03450         /* Store data... */
03451         tbl->p[ig][id][tbl->np[ig][id]] = press;
03452         tbl->t[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03453           = temp;
03454         tbl->u[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03455           [tbl->nu[ig][id][tbl->np[ig][id]]
03456            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) u;
03457         tbl->eps[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03458           [tbl->nu[ig][id][tbl->np[ig][id]]
03459            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) eps;
03460       }
03461
03462       /* Increment counters... */
03463       tbl->np[ig][id]++;
03464       for (ip = 0; ip < tbl->np[ig][id]; ip++) {
03465         tbl->nt[ig][id][ip]++;
03466         for (it = 0; it < tbl->nt[ig][id][ip]; it++)
03467           tbl->nu[ig][id][ip][it]++;
03468       }
03469
03470       /* Close file... */
03471       fclose(in);
03472     }
03473
03474   /* Write info... */
03475   printf("Initialize source function table...\n");
03476
03477   /* Loop over channels... */
03478 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(filename,it,i,n,f,fsum,nu)
03479   for (id = 0; id < ctl->nd; id++) {
03480
03481     /* Read filter function... */
03482     sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03483     read_shape(filename, nu, f, &n);
03484
03485     /* Compute source function table... */
03486     for (it = 0; it < TBLNS; it++) {
03487
03488       /* Set temperature... */
03489       tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03490
03491       /* Integrate Planck function... */
```

```
03492       fsum = 0;
03493       tbl->sr[id][it] = 0;
03494       for (i = 0; i < n; i++) {
03495         fsum += f[i];
03496         tbl->sr[id][it] += f[i] * planck(tbl->st[it], nu[i]);
03497       }
03498       tbl->sr[id][it] /= fsum;
03499     }
03500   }
03501 }
```

Here is the call graph for this function:



5.5.2.22 void intpol_atm (ctl_t * *ctl*, atm_t * *atm*, double *z*, double * *p*, double * *t*, double * *q*, double * *k*)

Interpolate atmospheric data.

Definition at line 3505 of file jurassic.c.

```
03512             {
03513
03514   int ig, ip, iw;
03515
03516   /* Get array index... */
03517   ip = locate_irr(atm->z, atm->np, z);
03518
03519   /* Interpolate... */
03520   *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);
03521   *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03522   for (ig = 0; ig < ctl->ng; ig++)
03523     q[ig] =
03524       LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03525   for (iw = 0; iw < ctl->nw; iw++)
03526     k[iw] =
03527       LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03528 }
```

Here is the call graph for this function:

5.5.2.23 void intpol_tbl (ctl_t * *ctl*, tbl_t * *tbl*, los_t * *los*, int *ip*, double *tau_path[NG][ND]*, double *tau_seg[ND]*)

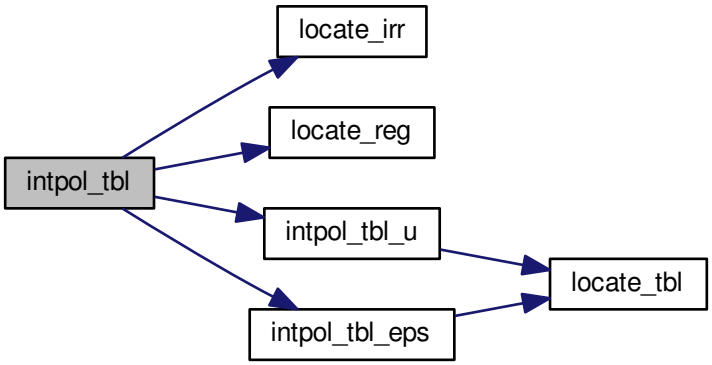
Get transmittance from look-up tables.

Definition at line 3532 of file jurassic.c.

```
03538                       {
03539
03540   double eps, eps00, eps01, eps10, eps11, u;
03541
03542   int id, ig, ipr, it0, it1;
03543
03544   /* Initialize... */
03545   if (ip <= 0)
03546     for (ig = 0; ig < ctl->ng; ig++)
03547       for (id = 0; id < ctl->nd; id++)
03548         tau_path[ig][id] = 1;
03549
03550   /* Loop over channels... */
03551   for (id = 0; id < ctl->nd; id++) {
03552
03553     /* Initialize... */
03554     tau_seg[id] = 1;
03555
03556     /* Loop over emitters.... */
03557     for (ig = 0; ig < ctl->ng; ig++) {
03558
03559       /* Check size of table (pressure)... */
03560       if (tbl->np[ig][id] < 2)
03561         eps = 0;
03562
03563       /* Check transmittance... */
03564       else if (tau_path[ig][id] < 1e-9)
03565         eps = 1;
03566
03567       /* Interpolate... */
03568       else {
03569
03570         /* Determine pressure and temperature indices... */
03571         ipr = locate_irr(tbl->p[ig][id], tbl->np[ig][id], los->p[ip]);
03572         it0 =
03573           locate_irr(tbl->t[ig][id][ipr], tbl->nt[ig][id][ipr], los->
      t[ip]);
03574         it1 =
03575           locate_reg(tbl->t[ig][id][ipr + 1], tbl->nt[ig][id][ipr + 1],
03576                      los->t[ip]);
03577
03578         /* Check size of table (temperature and column density)... */
03579         if (tbl->nt[ig][id][ipr] < 2 || tbl->nt[ig][id][ipr + 1] < 2
03580             || tbl->nu[ig][id][ipr][it0] < 2
03581             || tbl->nu[ig][id][ipr][it0 + 1] < 2
03582             || tbl->nu[ig][id][ipr + 1][it1] < 2
03583             || tbl->nu[ig][id][ipr + 1][it1 + 1] < 2)
03584           eps = 0;
03585
03586         else {
03587
03588           /* Get emissivities of extended path... */
03589           u = intpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[ig][id]);
03590           eps00 = intpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ig][ip]);
03591
03592           u = intpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[ig][id]);
03593           eps01 =
03594             intpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ig][ip]);
03595
03596           u = intpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[ig][id]);
03597           eps10 =
03598             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ig][ip]);
03599
03600           u =
03601             intpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[ig][id]);
03602           eps11 =
03603             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->
      u[ig][ip]);
03604
03605           /* Interpolate with respect to temperature... */
03606           eps00 = LIN(tbl->t[ig][id][ipr][it0], eps00,
03607                       tbl->t[ig][id][ipr][it0 + 1], eps01, los->t[ip]);
03608           eps11 = LIN(tbl->t[ig][id][ipr + 1][it1], eps10,
03609                       tbl->t[ig][id][ipr + 1][it1 + 1], eps11, los->t[ip]);
03610
03611           /* Interpolate with respect to pressure... */
03612           eps00 = LIN(tbl->p[ig][id][ipr], eps00,
```

```
03613                       tbl->p[ig][id][ipr + 1], eps11, los->p[ip]);
03614
03615           /* Check emssivity range... */
03616           eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03617
03618           /* Determine segment emissivity... */
03619           eps = 1 - (1 - eps00) / tau_path[ig][id];
03620         }
03621       }
03622
03623       /* Get transmittance of extended path... */
03624       tau_path[ig][id] *= (1 - eps);
03625
03626       /* Get segment transmittance... */
03627       tau_seg[id] *= (1 - eps);
03628     }
03629   }
03630 }
```

Here is the call graph for this function:



5.5.2.24 double intpol_tbl_eps (tbl_t * *tbl*, int *ig*, int *id*, int *ip*, int *it*, double *u*)

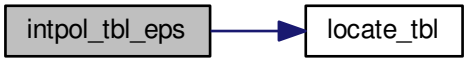
Interpolate emissivity from look-up tables.

Definition at line 3634 of file jurassic.c.

```
03640             {
03641
03642   int idx;
03643
03644   /* Lower boundary... */
03645   if (u < tbl->u[ig][id][ip][it][0])
03646     return LIN(0, 0, tbl->u[ig][id][ip][it][0], tbl->eps[ig][id][ip][it][0],
03647                u);
03648
03649   /* Upper boundary... */
03650   else if (u > tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
03651     return LIN(tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03652                tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03653                1e30, 1, u);
03654
03655   /* Interpolation... */
03656   else {
03657
03658     /* Get index... */
03659     idx = locate_tbl(tbl->u[ig][id][ip][it], tbl->nu[ig][id][ip][it], u);
03660
```

```
03661     /* Interpolate... */
03662     return
03663       LIN(tbl->u[ig][id][ip][it][idx], tbl->eps[ig][id][ip][it][idx],
03664           tbl->u[ig][id][ip][it][idx + 1], tbl->eps[ig][id][ip][it][idx + 1],
03665           u);
03666   }
03667 }
```

Here is the call graph for this function:



5.5.2.25 double intpol_tbl_u (tbl_t * *tbl*, int *ig*, int *id*, int *ip*, int *it*, double *eps*)

Interpolate column density from look-up tables.

Definition at line 3671 of file jurassic.c.

```
03677               {
03678
03679   int idx;
03680
03681   /* Lower boundary... */
03682   if (eps < tbl->eps[ig][id][ip][it][0])
03683     return LIN(0, 0, tbl->eps[ig][id][ip][it][0], tbl->u[ig][id][ip][it][0],
03684                eps);
03685
03686   /* Upper boundary... */
03687   else if (eps > tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
03688     return LIN(tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03689                tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03690                1, 1e30, eps);
03691
03692   /* Interpolation... */
03693   else {
03694
03695     /* Get index... */
03696     idx = locate_tbl(tbl->eps[ig][id][ip][it], tbl->nu[ig][id][ip][it], eps);
03697
03698     /* Interpolate... */
03699     return
03700       LIN(tbl->eps[ig][id][ip][it][idx], tbl->u[ig][id][ip][it][idx],
03701           tbl->eps[ig][id][ip][it][idx + 1], tbl->u[ig][id][ip][it][idx + 1],
03702           eps);
03703   }
03704 }
```

Here is the call graph for this function:

5.5.2.26 void jsec2time (double *jsec*, int * *year*, int * *mon*, int * *day*, int * *hour*, int * *min*, int * *sec*, double * *remain*)

Convert seconds to date.

Definition at line 3708 of file jurassic.c.

```
03716                   {
03717
03718   struct tm t0, *t1;
03719
03720   time_t jsec0;
03721
03722   t0.tm_year = 100;
03723   t0.tm_mon = 0;
03724   t0.tm_mday = 1;
03725   t0.tm_hour = 0;
03726   t0.tm_min = 0;
03727   t0.tm_sec = 0;
03728
03729   jsec0 = (time_t) jsec + timegm(&t0);
03730   t1 = gmtime(&jsec0);
03731
03732   *year = t1->tm_year + 1900;
03733   *mon = t1->tm_mon + 1;
03734   *day = t1->tm_mday;
03735   *hour = t1->tm_hour;
03736   *min = t1->tm_min;
03737   *sec = t1->tm_sec;
03738   *remain = jsec - floor(jsec);
03739 }
```

5.5.2.27 void kernel (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, gsl_matrix * *k*)

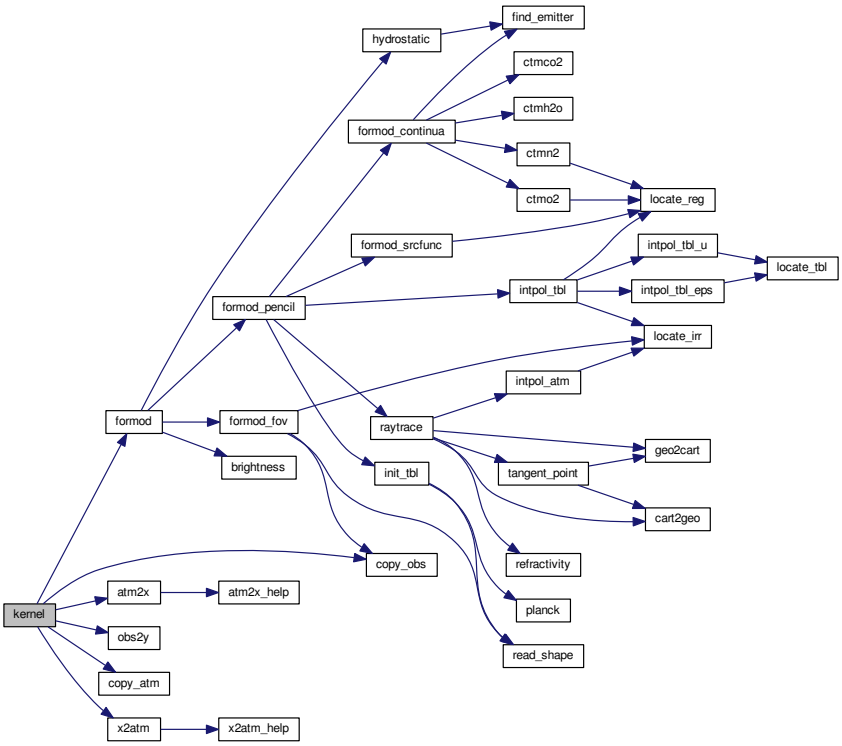
Compute Jacobians.

Definition at line 3743 of file jurassic.c.

```
03747                   {
03748
03749   atm_t *atm1;
03750   obs_t *obs1;
03751
03752   gsl_vector *x0, *x1, *yy0, *yy1;
03753
03754   int *iqa, j;
03755
03756   double h;
03757
03758   size_t i, n, m;
03759
03760   /* Get sizes... */
03761   m = k->size1;
03762   n = k->size2;
03763
03764   /* Allocate... */
03765   x0 = gsl_vector_alloc(n);
03766   yy0 = gsl_vector_alloc(m);
03767   ALLOC(iqa, int,
03768         N);
03769
03770   /* Compute radiance for undisturbed atmospheric data... */
03771   formod(ctl, atm, obs);
03772
03773   /* Compose vectors... */
03774   atm2x(ctl, atm, x0, iqa, NULL);
03775   obs2y(ctl, obs, yy0, NULL, NULL);
03776
03777   /* Initialize kernel matrix... */
03778   gsl_matrix_set_zero(k);
03779
03780   /* Loop over state vector elements... */
03781 #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(i, j, h, x1, yy1, atm1,
      obs1)
03782   for (j = 0; j < (int) n; j++) {
03783
03784     /* Allocate... */
```

```
03785     x1 = gsl_vector_alloc(n);
03786     yy1 = gsl_vector_alloc(m);
03787     ALLOC(atm1, atm_t, 1);
03788     ALLOC(obs1, obs_t, 1);
03789
03790     /* Set perturbation size... */
03791     if (iqa[j] == IDXP)
03792       h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-7);
03793     else if (iqa[j] == IDXT)
03794       h = 1;
03795     else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03796       h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-15);
03797     else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03798       h = 1e-4;
03799     else
03800       ERRMSG("Cannot set perturbation size!");
03801
03802     /* Disturb state vector element... */
03803     gsl_vector_memcpy(x1, x0);
03804     gsl_vector_set(x1, (size_t) j, gsl_vector_get(x1, (size_t) j) + h);
03805     copy_atm(ctl, atm1, atm, 0);
03806     copy_obs(ctl, obs1, obs, 0);
03807     x2atm(ctl, x1, atm1);
03808
03809     /* Compute radiance for disturbed atmospheric data... */
03810     formod(ctl, atm1, obs1);
03811
03812     /* Compose measurement vector for disturbed radiance data... */
03813     obs2y(ctl, obs1, yy1, NULL, NULL);
03814
03815     /* Compute derivatives... */
03816     for (i = 0; i < m; i++)
03817       gsl_matrix_set(k, i, (size_t) j,
03818                      (gsl_vector_get(yy1, i) - gsl_vector_get(yy0, i)) / h);
03819
03820     /* Free... */
03821     gsl_vector_free(x1);
03822     gsl_vector_free(yy1);
03823     free(atm1);
03824     free(obs1);
03825   }
03826
03827   /* Free... */
03828   gsl_vector_free(x0);
03829   gsl_vector_free(yy0);
03830   free(iqa);
03831 }
```

Here is the call graph for this function:



5.5.2.28 int locate_irr (double * *xx*, int *n*, double *x*)

Find array index for irregular grid.

Definition at line 3835 of file jurassic.c.

```
03838             {
03839
03840   int i, ilo, ihi;
03841
03842   ilo = 0;
03843   ihi = n - 1;
03844   i = (ihi + ilo) >> 1;
03845
03846   if (xx[i] < xx[i + 1])
03847     while (ihi > ilo + 1) {
03848       i = (ihi + ilo) >> 1;
03849       if (xx[i] > x)
03850         ihi = i;
03851       else
03852         ilo = i;
03853   } else
03854     while (ihi > ilo + 1) {
03855       i = (ihi + ilo) >> 1;
03856       if (xx[i] <= x)
03857         ihi = i;
03858       else
03859         ilo = i;
03860     }
03861
03862   return ilo;
03863 }
```

5.5.2.29 int locate_reg (double * *xx*, int *n*, double *x*)

Find array index for regular grid.

Definition at line 3867 of file jurassic.c.

```
03870             {
03871
03872   int i;
03873
03874   /* Calculate index... */
03875   i = (int) ((x - xx[0]) / (xx[1] - xx[0]));
03876
03877   /* Check range... */
03878   if (i < 0)
03879     i = 0;
03880   else if (i >= n - 2)
03881     i = n - 2;
03882
03883   return i;
03884 }
```

5.5.2.30 int locate_tbl (float * *xx*, int *n*, double *x*)

Find array index in float array.

Definition at line 3888 of file jurassic.c.

```
03891             {
03892
03893   int i, ilo, ihi;
03894
03895   ilo = 0;
03896   ihi = n - 1;
03897   i = (ihi + ilo) >> 1;
03898
03899   while (ihi > ilo + 1) {
03900     i = (ihi + ilo) >> 1;
03901     if (xx[i] > x)
03902       ihi = i;
03903     else
03904       ilo = i;
03905   }
03906
03907   return ilo;
03908 }
```

5.5.2.31 size_t obs2y (ctl_t * *ctl*, obs_t * *obs*, gsl_vector * *y*, int * *ida*, int * *ira*)

Compose measurement vector.

Definition at line 3912 of file jurassic.c.

```
03917             {
03918
03919   int id, ir;
03920
03921   size_t m = 0;
03922
03923   /* Determine measurement vector... */
03924   for (ir = 0; ir < obs->nr; ir++)
03925     for (id = 0; id < ctl->nd; id++)
03926       if (gsl_finite(obs->rad[id][ir])) {
03927         if (y != NULL)
03928           gsl_vector_set(y, m, obs->rad[id][ir]);
03929         if (ida != NULL)
03930           ida[m] = id;
03931         if (ira != NULL)
03932           ira[m] = ir;
03933         m++;
03934       }
03935
03936   return m;
03937 }
```

5.5.2.32 double planck (double *t*, double *nu*)

Compute Planck function.

Definition at line 3941 of file jurassic.c.

```
03943             {
03944
03945   return C1 * POW3(nu) / gsl_expm1(C2 * nu / t);
03946 }
```

5.5.2.33 void raytrace (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, los_t * *los*, int *ir*)

Do ray-tracing to determine LOS.

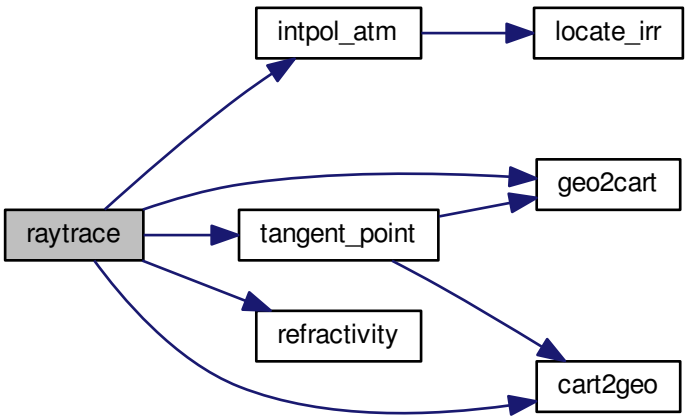
Definition at line 3950 of file jurassic.c.

```
03955           {
03956
03957   double cosa, d, dmax, dmin = 0, ds, ex0[3], ex1[3], frac, h = 0.02, k[NW],
03958     lat, lon, n, naux, ng[3], norm, p, q[NG], t, x[3], xh[3],
03959     xobs[3], xvp[3], z = 1e99, zmax, zmin, zrefrac = 60;
03960
03961   int i, ig, ip, iw, stop = 0;
03962
03963   /* Initialize... */
03964   los->np = 0;
03965   los->tsurf = -999;
03966   obs->tpz[ir] = obs->vpz[ir];
03967   obs->tplon[ir] = obs->vplon[ir];
03968   obs->tplat[ir] = obs->vplat[ir];
03969
03970   /* Get altitude range of atmospheric data... */
03971   gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03972
03973   /* Check observer altitude... */
03974   if (obs->obsz[ir] < zmin)
03975     ERRMSG("Observer below surface!");
03976
03977   /* Check view point altitude... */
03978   if (obs->vpz[ir] > zmax)
03979     return;
03980
03981   /* Determine Cartesian coordinates for observer and view point... */
03982   geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
03983   geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
03984
03985   /* Determine initial tangent vector... */
03986   for (i = 0; i < 3; i++)
03987     ex0[i] = xvp[i] - xobs[i];
03988   norm = NORM(ex0);
03989   for (i = 0; i < 3; i++)
03990     ex0[i] /= norm;
03991
03992   /* Observer within atmosphere... */
03993   for (i = 0; i < 3; i++)
03994     x[i] = xobs[i];
03995
03996   /* Observer above atmosphere (search entry point)... */
03997   if (obs->obsz[ir] > zmax) {
03998     dmax = norm;
03999     while (fabs(dmin - dmax) > 0.001) {
04000       d = (dmax + dmin) / 2;
04001       for (i = 0; i < 3; i++)
04002         x[i] = xobs[i] + d * ex0[i];
04003       cart2geo(x, &z, &lon, &lat);
04004       if (z <= zmax && z > zmax - 0.001)
04005         break;
04006       if (z < zmax - 0.0005)
04007         dmax = d;
04008       else
04009         dmin = d;
04010     }
04011   }
04012
04013   /* Ray-tracing... */
```

```
04014   while (1) {
04015
04016     /* Set step length... */
04017     ds = ctl->rayds;
04018     if (ctl->raydz > 0) {
04019       norm = NORM(x);
04020       for (i = 0; i < 3; i++)
04021         xh[i] = x[i] / norm;
04022       cosa = fabs(DOTP(ex0, xh));
04023       if (cosa != 0)
04024         ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04025     }
04026
04027     /* Determine geolocation... */
04028     cart2geo(x, &z, &lon, &lat);
04029
04030     /* Check if LOS hits the ground or has left atmosphere... */
04031     if (z < zmin || z > zmax) {
04032       stop = (z < zmin ? 2 : 1);
04033       frac =
04034         ((z <
04035           zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04036                                                                    1]);
04037       geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04038                los->lat[los->np - 1], xh);
04039       for (i = 0; i < 3; i++)
04040         x[i] = xh[i] + frac * (x[i] - xh[i]);
04041       cart2geo(x, &z, &lon, &lat);
04042       los->ds[los->np - 1] = ds * frac;
04043       ds = 0;
04044     }
04045
04046     /* Interpolate atmospheric data... */
04047     intpol_atm(ctl, atm, z, &p, &t, q, k);
04048
04049     /* Save data... */
04050     los->lon[los->np] = lon;
04051     los->lat[los->np] = lat;
04052     los->z[los->np] = z;
04053     los->p[los->np] = p;
04054     los->t[los->np] = t;
04055     for (ig = 0; ig < ctl->ng; ig++)
04056       los->q[ig][los->np] = q[ig];
04057     for (iw = 0; iw < ctl->nw; iw++)
04058       los->k[iw][los->np] = k[iw];
04059     los->ds[los->np] = ds;
04060
04061     /* Increment and check number of LOS points... */
04062     if ((++los->np) > NLOS)
04063       ERRMSG("Too many LOS points!");
04064
04065     /* Check stop flag... */
04066     if (stop) {
04067       los->tsurf = (stop == 2 ? t : -999);
04068       break;
04069     }
04070
04071     /* Determine refractivity... */
04072     if (ctl->refrac && z <= zrefrac)
04073       n = 1 + refractivity(p, t);
04074     else
04075       n = 1;
04076
04077     /* Construct new tangent vector (first term)... */
04078     for (i = 0; i < 3; i++)
04079       ex1[i] = ex0[i] * n;
04080
04081     /* Compute gradient of refractivity... */
04082     if (ctl->refrac && z <= zrefrac) {
04083       for (i = 0; i < 3; i++)
04084         xh[i] = x[i] + 0.5 * ds * ex0[i];
04085       cart2geo(xh, &z, &lon, &lat);
04086       intpol_atm(ctl, atm, z, &p, &t, q, k);
04087       n = refractivity(p, t);
04088       for (i = 0; i < 3; i++) {
04089         xh[i] += h;
04090         cart2geo(xh, &z, &lon, &lat);
04091         intpol_atm(ctl, atm, z, &p, &t, q, k);
04092         naux = refractivity(p, t);
04093         ng[i] = (naux - n) / h;
04094         xh[i] -= h;
04095       }
04096     } else
04097       for (i = 0; i < 3; i++)
04098         ng[i] = 0;
04099
04100     /* Construct new tangent vector (second term)... */
```

```
04101     for (i = 0; i < 3; i++)
04102       ex1[i] += ds * ng[i];
04103
04104     /* Normalize new tangent vector... */
04105     norm = NORM(ex1);
04106     for (i = 0; i < 3; i++)
04107       ex1[i] /= norm;
04108
04109     /* Determine next point of LOS... */
04110     for (i = 0; i < 3; i++)
04111       x[i] += 0.5 * ds * (ex0[i] + ex1[i]);
04112
04113     /* Copy tangent vector... */
04114     for (i = 0; i < 3; i++)
04115       ex0[i] = ex1[i];
04116   }
04117
04118   /* Get tangent point (to be done before changing segment lengths!)... */
04119   tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->
      tplat[ir]);
04120
04121   /* Change segment lengths according to trapezoid rule... */
04122   for (ip = los->np - 1; ip >= 1; ip--)
04123     los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04124   los->ds[0] *= 0.5;
04125
04126   /* Compute column density... */
04127   for (ip = 0; ip < los->np; ip++)
04128     for (ig = 0; ig < ctl->ng; ig++)
04129       los->u[ig][ip] = 10 * los->q[ig][ip] * los->p[ip]
04130         / (KB * los->t[ip]) * los->ds[ip];
04131 }
```

Here is the call graph for this function:



5.5.2.34 void read_atm (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, atm_t * *atm*)

Read atmospheric data.

Definition at line 4135 of file jurassic.c.

```
04139               {
04140
04141   FILE *in;
04142
04143   char file[LEN], line[LEN], *tok;
```

```
04144
04145   int ig, iw;
04146
04147   /* Init... */
04148   atm->np = 0;
04149
04150   /* Set filename... */
04151   if (dirname != NULL)
04152     sprintf(file, "%s/%s", dirname, filename);
04153   else
04154     sprintf(file, "%s", filename);
04155
04156   /* Write info... */
04157   printf("Read atmospheric data: %s\n", file);
04158
04159   /* Open file... */
04160   if (!(in = fopen(file, "r")))
04161     ERRMSG("Cannot open file!");
04162
04163   /* Read line... */
04164   while (fgets(line, LEN, in)) {
04165
04166     /* Read data... */
04167     TOK(line, tok, "%lg", atm->time[atm->np]);
04168     TOK(NULL, tok, "%lg", atm->z[atm->np]);
04169     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04170     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04171     TOK(NULL, tok, "%lg", atm->p[atm->np]);
04172     TOK(NULL, tok, "%lg", atm->t[atm->np]);
04173     for (ig = 0; ig < ctl->ng; ig++)
04174       TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04175     for (iw = 0; iw < ctl->nw; iw++)
04176       TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04177
04178     /* Increment data point counter... */
04179     if ((++atm->np) > NP)
04180       ERRMSG("Too many data points!");
04181   }
04182
04183   /* Close file... */
04184   fclose(in);
04185
04186   /* Check number of points... */
04187   if (atm->np < 1)
04188     ERRMSG("Could not read any data!");
04189 }
```

5.5.2.35 void read_ctl (int *argc*, char * *argv[]*, ctl_t * *ctl*)

Read forward model control parameters.

Definition at line 4193 of file jurassic.c.

```
04196               {
04197
04198   int id, ig, iw;
04199
04200   /* Write info... */
04201   printf("\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04202          "(executable: %s | compiled: %s, %s)\n\n",
04203          argv[0], __DATE__, __TIME__);
04204
04205   /* Emitters... */
04206   ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04207   if (ctl->ng < 0 || ctl->ng > NG)
04208     ERRMSG("Set 0 <= NG <= MAX!");
04209   for (ig = 0; ig < ctl->ng; ig++)
04210     scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04211
04212   /* Radiance channels... */
04213   ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04214   if (ctl->nd < 0 || ctl->nd > ND)
04215     ERRMSG("Set 0 <= ND <= MAX!");
04216   for (id = 0; id < ctl->nd; id++)
04217     ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04218
04219   /* Spectral windows... */
04220   ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04221   if (ctl->nw < 0 || ctl->nw > NW)
04222     ERRMSG("Set 0 <= NW <= MAX!");
```

```
04223   for (id = 0; id < ctl->nd; id++)
04224     ctl->window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04225
04226   /* Emissivity look-up tables... */
04227   scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04228
04229   /* Hydrostatic equilibrium... */
04230   ctl->hydz = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04231
04232   /* Continua... */
04233   ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04234   ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04235   ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04236   ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04237
04238   /* Ray-tracing... */
04239   ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04240   ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04241   ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.5", NULL);
04242
04243   /* Field of view... */
04244   scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04245
04246   /* Retrieval interface... */
04247   ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04248   ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04249   ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04250   ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04251   for (ig = 0; ig < ctl->ng; ig++) {
04252     ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETQ_ZMIN", ig, "-999", NULL);
04253     ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETQ_ZMAX", ig, "-999", NULL);
04254   }
04255   for (iw = 0; iw < ctl->nw; iw++) {
04256     ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04257     ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04258   }
04259
04260   /* Output flags... */
04261   ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04262   ctl->write_matrix =
04263     (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04264 }
```

Here is the call graph for this function:

5.5.2.36 void read_matrix (const char * *dirname*, const char * *filename*, gsl_matrix * *matrix*)

Read matrix.

Definition at line 4268 of file jurassic.c.

```
04271                         {
04272
04273   FILE *in;
04274
04275   char dum[LEN], file[LEN], line[LEN];
04276
04277   double value;
04278
04279   int i, j;
04280
04281   /* Set filename... */
```

```
04282   if (dirname != NULL)
04283     sprintf(file, "%s/%s", dirname, filename);
04284   else
04285     sprintf(file, "%s", filename);
04286
04287   /* Write info... */
04288   printf("Read matrix: %s\n", file);
04289
04290   /* Open file... */
04291   if (!(in = fopen(file, "r")))
04292     ERRMSG("Cannot open file!");
04293
04294   /* Read data... */
04295   gsl_matrix_set_zero(matrix);
04296   while (fgets(line, LEN, in))
04297     if (sscanf(line, "%d %s %s %s %s %s %d %s %s %s %s %s %lg",
04298                &i, dum, dum, dum, dum, dum,
04299                &j, dum, dum, dum, dum, dum, &value) == 13)
04300       gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04301
04302   /* Close file... */
04303   fclose(in);
04304 }
```

5.5.2.37 void read_obs (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, obs_t * *obs*)

Read observation data.

Definition at line 4308 of file jurassic.c.

```
04312                {
04313
04314   FILE *in;
04315
04316   char file[LEN], line[LEN], *tok;
04317
04318   int id;
04319
04320   /* Init... */
04321   obs->nr = 0;
04322
04323   /* Set filename... */
04324   if (dirname != NULL)
04325     sprintf(file, "%s/%s", dirname, filename);
04326   else
04327     sprintf(file, "%s", filename);
04328
04329   /* Write info... */
04330   printf("Read observation data: %s\n", file);
04331
04332   /* Open file... */
04333   if (!(in = fopen(file, "r")))
04334     ERRMSG("Cannot open file!");
04335
04336   /* Read line... */
04337   while (fgets(line, LEN, in)) {
04338
04339     /* Read data... */
04340     TOK(line, tok, "%lg", obs->time[obs->nr]);
04341     TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04342     TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04343     TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04344     TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04345     TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04346     TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04347     TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04348     TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04349     TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04350     for (id = 0; id < ctl->nd; id++)
04351       TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04352     for (id = 0; id < ctl->nd; id++)
04353       TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04354
04355     /* Increment counter... */
04356     if ((++obs->nr) > NR)
04357       ERRMSG("Too many rays!");
04358   }
04359
04360   /* Close file... */
04361   fclose(in);
```

```
04362
04363   /* Check number of points... */
04364   if (obs->nr < 1)
04365     ERRMSG("Could not read any data!");
04366 }
```

5.5.2.38 void read_shape (const char * *filename*, double * *x*, double * *y*, int * *n*)

Read shape function.

Definition at line 4370 of file jurassic.c.

```
04374           {
04375
04376   FILE *in;
04377
04378   char line[LEN];
04379
04380   /* Write info... */
04381   printf("Read shape function: %s\n", filename);
04382
04383   /* Open file... */
04384   if (!(in = fopen(filename, "r")))
04385     ERRMSG("Cannot open file!");
04386
04387   /* Read data... */
04388   *n = 0;
04389   while (fgets(line, LEN, in))
04390     if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04391       if ((++(*n)) > NSHAPE)
04392         ERRMSG("Too many data points!");
04393
04394   /* Check number of points... */
04395   if (*n < 1)
04396     ERRMSG("Could not read any data!");
04397
04398   /* Close file... */
04399   fclose(in);
04400 }
```

5.5.2.39 double refractivity (double *p*, double *t*)

Compute refractivity (return value is n - 1).

Definition at line 4404 of file jurassic.c.

```
04406             {
04407
04408   /* Refractivity of air at 4 to 15 micron... */
04409   return 7.753e-05 * p / t;
04410 }
```

5.5.2.40 double scan_ctl (int *argc*, char * *argv[]*, const char * *varname*, int *arridx*, const char * *defvalue*, char * *value*)

Search control parameter file for variable entry.

Definition at line 4414 of file jurassic.c.

```
04420               {
04421
04422   FILE *in = NULL;
04423
04424   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04425     msg[2 * LEN], rvarname[LEN], rval[LEN];
04426
04427   int contain = 0, i;
04428
04429   /* Open file... */
04430   if (argv[1][0] != '-')
04431     if (!(in = fopen(argv[1], "r")))
04432       ERRMSG("Cannot open file!");
04433
04434   /* Set full variable name... */
04435   if (arridx >= 0) {
04436     sprintf(fullname1, "%s[%d]", varname, arridx);
04437     sprintf(fullname2, "%s[*]", varname);
04438   } else {
04439     sprintf(fullname1, "%s", varname);
04440     sprintf(fullname2, "%s", varname);
04441   }
04442
04443   /* Read data... */
04444   if (in != NULL)
04445     while (fgets(line, LEN, in))
04446       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04447         if (strcasecmp(rvarname, fullname1) == 0 ||
04448             strcasecmp(rvarname, fullname2) == 0) {
04449           contain = 1;
04450           break;
04451         }
04452   for (i = 1; i < argc - 1; i++)
04453     if (strcasecmp(argv[i], fullname1) == 0 ||
04454         strcasecmp(argv[i], fullname2) == 0) {
04455       sprintf(rval, "%s", argv[i + 1]);
04456       contain = 1;
04457       break;
04458     }
04459
04460   /* Close file... */
04461   if (in != NULL)
04462     fclose(in);
04463
04464   /* Check for missing variables... */
04465   if (!contain) {
04466     if (strlen(defvalue) > 0)
04467       sprintf(rval, "%s", defvalue);
04468     else {
04469       sprintf(msg, "Missing variable %s!\n", fullname1);
04470       ERRMSG(msg);
04471     }
04472   }
04473
04474   /* Write info... */
04475   printf("%s = %s\n", fullname1, rval);
04476
04477   /* Return values... */
04478   if (value != NULL)
04479     sprintf(value, "%s", rval);
04480   return atof(rval);
04481 }
```

5.5.2.41 void tangent_point (los_t * *los*, double * *tpz*, double * *tplon*, double * *tplat*)

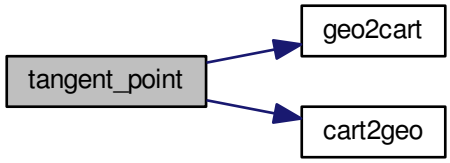
Find tangent point of a given LOS.

Definition at line 4485 of file jurassic.c.

```
04489               {
04490
04491   double a, b, c, dummy, v[3], v0[3], v2[3], x, x1, x2, yy0, yy1, yy2;
04492
04493   size_t i, ip;
04494
04495   /* Find minimum altitude... */
04496   ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
04497
04498   /* Nadir or zenith... */
04499   if (ip <= 0 || ip >= (size_t) los->np - 1) {
```

```
04500     *tpz = los->z[los->np - 1];
04501     *tplon = los->lon[los->np - 1];
04502     *tplat = los->lat[los->np - 1];
04503   }
04504
04505   /* Limb... */
04506   else {
04507
04508     /* Determine interpolating polynomial y=a*x^2+b*x+c... */
04509     yy0 = los->z[ip - 1];
04510     yy1 = los->z[ip];
04511     yy2 = los->z[ip + 1];
04512     x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
04513     x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
04514     a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);
04515     b = -(yy0 - yy1) / x1 - a * x1;
04516     c = yy0;
04517
04518     /* Get tangent point location... */
04519     x = -b / (2 * a);
04520     *tpz = a * x * x + b * x + c;
04521     geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
04522     geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);
04523     for (i = 0; i < 3; i++)
04524       v[i] = LIN(0.0, v0[i], x2, v2[i], x);
04525     cart2geo(v, &dummy, tplon, tplat);
04526   }
04527 }
```

Here is the call graph for this function:



5.5.2.42 void time2jsec (int *year*, int *mon*, int *day*, int *hour*, int *min*, int *sec*, double *remain*, double * *jsec*)

Convert date to seconds.

Definition at line 4531 of file jurassic.c.

```
04539                 {
04540
04541   struct tm t0, t1;
04542
04543   t0.tm_year = 100;
04544   t0.tm_mon = 0;
04545   t0.tm_mday = 1;
04546   t0.tm_hour = 0;
04547   t0.tm_min = 0;
04548   t0.tm_sec = 0;
04549
04550   t1.tm_year = year - 1900;
04551   t1.tm_mon = mon - 1;
04552   t1.tm_mday = day;
04553   t1.tm_hour = hour;
04554   t1.tm_min = min;
04555   t1.tm_sec = sec;
04556
04557   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
04558 }
```

5.5.2.43 void timer (const char * *name*, const char * *file*, const char * *func*, int *line*, int *mode*)

Measure wall-clock time.

Definition at line 4562 of file jurassic.c.

```
04567            {
04568
04569   static double w0[10];
04570
04571   static int l0[10], nt;
04572
04573   /* Start new timer... */
04574   if (mode == 1) {
04575     w0[nt] = omp_get_wtime();
04576     l0[nt] = line;
04577     if ((++nt) >= 10)
04578       ERRMSG("Too many timers!");
04579   }
04580
04581   /* Write elapsed time... */
04582   else {
04583
04584     /* Check timer index... */
04585     if (nt - 1 < 0)
04586       ERRMSG("Coding error!");
04587
04588     /* Write elapsed time... */
04589     printf("Timer '%s' (%s, %s, l%d-%d): %.3f sec\n",
04590            name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
04591   }
04592
04593   /* Stop timer... */
04594   if (mode == 3)
04595     nt--;
04596 }
```

5.5.2.44 void write_atm (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, atm_t * *atm*)

Write atmospheric data.

Definition at line 4600 of file jurassic.c.

```
04604               {
04605
04606   FILE *out;
04607
04608   char file[LEN];
04609
04610   int ig, ip, iw, n = 6;
04611
04612   /* Set filename... */
04613   if (dirname != NULL)
04614     sprintf(file, "%s/%s", dirname, filename);
04615   else
04616     sprintf(file, "%s", filename);
04617
04618   /* Write info... */
04619   printf("Write atmospheric data: %s\n", file);
04620
04621   /* Create file... */
04622   if (!(out = fopen(file, "w")))
04623     ERRMSG("Cannot create file!");
04624
04625   /* Write header... */
04626   fprintf(out,
04627           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04628           "# $2 = altitude [km]\n"
04629           "# $3 = longitude [deg]\n"
04630           "# $4 = latitude [deg]\n"
04631           "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
04632   for (ig = 0; ig < ctl->ng; ig++)
04633     fprintf(out, "# $%d = %s volume mixing ratio\n", ++n, ctl->emitter[ig]);
04634   for (iw = 0; iw < ctl->nw; iw++)
04635     fprintf(out, "# $%d = window %d: extinction [1/km]\n", ++n, iw);
04636
```

```
04637   /* Write data... */
04638   for (ip = 0; ip < atm->np; ip++) {
04639     if (ip == 0 || atm->lat[ip] != atm->lat[ip - 1]
04640         || atm->lon[ip] != atm->lon[ip - 1])
04641       fprintf(out, "\n");
04642     fprintf(out, "%.2f %g %g %g %g %g", atm->time[ip], atm->z[ip],
04643             atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
04644     for (ig = 0; ig < ctl->ng; ig++)
04645       fprintf(out, " %g", atm->q[ig][ip]);
04646     for (iw = 0; iw < ctl->nw; iw++)
04647       fprintf(out, " %g", atm->k[iw][ip]);
04648     fprintf(out, "\n");
04649   }
04650
04651   /* Close file... */
04652   fclose(out);
04653 }
```

5.5.2.45 void write_matrix (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, gsl_matrix * *matrix*, atm_t * *atm*, obs_t * *obs*, const char * *rowspace*, const char * *colspace*, const char * *sort*)

Write matrix.

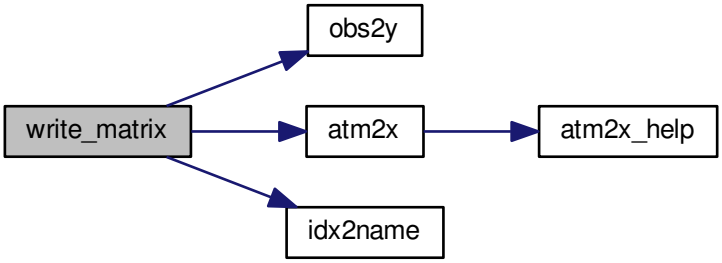
Definition at line 4657 of file jurassic.c.

```
04666                   {
04667
04668   FILE *out;
04669
04670   char file[LEN], quantity[LEN];
04671
04672   int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
04673
04674   size_t i, j, nc, nr;
04675
04676   /* Check output flag... */
04677   if (!ctl->write_matrix)
04678     return;
04679
04680   /* Allocate... */
04681   ALLOC(cida, int, M);
04682   ALLOC(ciqa, int,
04683         N);
04684   ALLOC(cipa, int,
04685         N);
04686   ALLOC(cira, int,
04687         M);
04688   ALLOC(rida, int,
04689         M);
04690   ALLOC(riqa, int,
04691         N);
04692   ALLOC(ripa, int,
04693         N);
04694   ALLOC(rira, int,
04695         M);
04696
04697   /* Set filename... */
04698   if (dirname != NULL)
04699     sprintf(file, "%s/%s", dirname, filename);
04700   else
04701     sprintf(file, "%s", filename);
04702
04703   /* Write info... */
04704   printf("Write matrix: %s\n", file);
04705
04706   /* Create file... */
04707   if (!(out = fopen(file, "w")))
04708     ERRMSG("Cannot create file!");
04709
04710   /* Write header (row space)... */
04711   if (rowspace[0] == 'y') {
04712
04713     fprintf(out,
04714             "# $1 = Row: index (measurement space)\n"
04715             "# $2 = Row: channel wavenumber [cm^-1]\n"
04716             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
04717             "# $4 = Row: view point altitude [km]\n"
04718             "# $5 = Row: view point longitude [deg]\n"
04719             "# $6 = Row: view point latitude [deg]\n");
04720
```

```
04721     /* Get number of rows... */
04722     nr = obs2y(ctl, obs, NULL, rida, rira);
04723
04724   } else {
04725
04726     fprintf(out,
04727             "# $1 = Row: index (state space)\n"
04728             "# $2 = Row: name of quantity\n"
04729             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
04730             "# $4 = Row: altitude [km]\n"
04731             "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
04732
04733     /* Get number of rows... */
04734     nr = atm2x(ctl, atm, NULL, riqa, ripa);
04735   }
04736
04737   /* Write header (column space)... */
04738   if (colspace[0] == 'y') {
04739
04740     fprintf(out,
04741             "# $7 = Col: index (measurement space)\n"
04742             "# $8 = Col: channel wavenumber [cm^-1]\n"
04743             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04744             "# $10 = Col: view point altitude [km]\n"
04745             "# $11 = Col: view point longitude [deg]\n"
04746             "# $12 = Col: view point latitude [deg]\n");
04747
04748     /* Get number of columns... */
04749     nc = obs2y(ctl, obs, NULL, cida, cira);
04750
04751   } else {
04752
04753     fprintf(out,
04754             "# $7 = Col: index (state space)\n"
04755             "# $8 = Col: name of quantity\n"
04756             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04757             "# $10 = Col: altitude [km]\n"
04758             "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
04759
04760     /* Get number of columns... */
04761     nc = atm2x(ctl, atm, NULL, ciqa, cipa);
04762   }
04763
04764   /* Write header entry... */
04765   fprintf(out, "# $13 = Matrix element\n\n");
04766
04767   /* Write matrix data... */
04768   i = j = 0;
04769   while (i < nr && j < nc) {
04770
04771     /* Write info about the row... */
04772     if (rowspace[0] == 'y')
04773       fprintf(out, "%d %g %.2f %g %g %g",
04774               (int) i, ctl->nu[rida[i]],
04775               obs->time[rira[i]], obs->vpz[rira[i]],
04776               obs->vplon[rira[i]], obs->vplat[rira[i]]);
04777     else {
04778       idx2name(ctl, riqa[i], quantity);
04779       fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
04780               atm->time[ripa[i]], atm->z[ripa[i]],
04781               atm->lon[ripa[i]], atm->lat[ripa[i]]);
04782     }
04783
04784     /* Write info about the column... */
04785     if (colspace[0] == 'y')
04786       fprintf(out, " %d %g %.2f %g %g %g",
04787               (int) j, ctl->nu[cida[j]],
04788               obs->time[cira[j]], obs->vpz[cira[j]],
04789               obs->vplon[cira[j]], obs->vplat[cira[j]]);
04790     else {
04791       idx2name(ctl, ciqa[j], quantity);
04792       fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
04793               atm->time[cipa[j]], atm->z[cipa[j]],
04794               atm->lon[cipa[j]], atm->lat[cipa[j]]);
04795     }
04796
04797     /* Write matrix entry... */
04798     fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
04799
04800     /* Set matrix indices... */
04801     if (sort[0] == 'r') {
04802       j++;
04803       if (j >= nc) {
04804         j = 0;
04805         i++;
04806         fprintf(out, "\n");
04807       }
```

```
04808     } else {
04809       i++;
04810       if (i >= nr) {
04811         i = 0;
04812         j++;
04813         fprintf(out, "\n");
04814       }
04815     }
04816   }
04817
04818   /* Close file... */
04819   fclose(out);
04820
04821   /* Free... */
04822   free(cida);
04823   free(ciqa);
04824   free(cipa);
04825   free(cira);
04826   free(rida);
04827   free(riqa);
04828   free(ripa);
04829   free(rira);
04830 }
```

Here is the call graph for this function:



5.5.2.46 void write_obs (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, obs_t * *obs*)

Write observation data.

Definition at line 4834 of file jurassic.c.

```
04838                {
04839
04840   FILE *out;
04841
04842   char file[LEN];
04843
04844   int id, ir, n = 10;
04845
04846   /* Set filename... */
04847   if (dirname != NULL)
04848     sprintf(file, "%s/%s", dirname, filename);
04849   else
04850     sprintf(file, "%s", filename);
04851
04852   /* Write info... */
04853   printf("Write observation data: %s\n", file);
04854
04855   /* Create file... */
04856   if (!(out = fopen(file, "w")))
04857     ERRMSG("Cannot create file!");
04858
```

```
04859   /* Write header... */
04860   fprintf(out,
04861           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04862           "# $2 = observer altitude [km]\n"
04863           "# $3 = observer longitude [deg]\n"
04864           "# $4 = observer latitude [deg]\n"
04865           "# $5 = view point altitude [km]\n"
04866           "# $6 = view point longitude [deg]\n"
04867           "# $7 = view point latitude [deg]\n"
04868           "# $8 = tangent point altitude [km]\n"
04869           "# $9 = tangent point longitude [deg]\n"
04870           "# $10 = tangent point latitude [deg]\n");
04871   for (id = 0; id < ctl->nd; id++)
04872     fprintf(out, "# $%d = channel %g: radiance [W/(m^2 sr cm^-1)]\n",
04873             ++n, ctl->nu[id]);
04874   for (id = 0; id < ctl->nd; id++)
04875     fprintf(out, "# $%d = channel %g: transmittance\n", ++n, ctl->nu[id]);
04876
04877   /* Write data... */
04878   for (ir = 0; ir < obs->nr; ir++) {
04879     if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
04880       fprintf(out, "\n");
04881     fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
04882             obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
04883             obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
04884             obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
04885     for (id = 0; id < ctl->nd; id++)
04886       fprintf(out, " %g", obs->rad[id][ir]);
04887     for (id = 0; id < ctl->nd; id++)
04888       fprintf(out, " %g", obs->tau[id][ir]);
04889     fprintf(out, "\n");
04890   }
04891
04892   /* Close file... */
04893   fclose(out);
04894 }
```

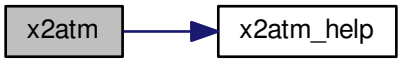
5.5.2.47 void x2atm (ctl_t * *ctl*, gsl_vector * *x*, atm_t * *atm*)

Decompose parameter vector or state vector.

Definition at line 4898 of file jurassic.c.

```
04901               {
04902
04903   int ig, iw;
04904
04905   size_t n = 0;
04906
04907   /* Set pressure... */
04908   x2atm_help(atm, ctl->retp_zmin, ctl->retp_zmax, atm->
      p, x, &n);
04909
04910   /* Set temperature... */
04911   x2atm_help(atm, ctl->rett_zmin, ctl->rett_zmax, atm->
      t, x, &n);
04912
04913   /* Set volume mixing ratio... */
04914   for (ig = 0; ig < ctl->ng; ig++)
04915     x2atm_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
04916                atm->q[ig], x, &n);
04917
04918   /* Set extinction... */
04919   for (iw = 0; iw < ctl->nw; iw++)
04920     x2atm_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
04921                atm->k[iw], x, &n);
04922 }
```

Here is the call graph for this function:



5.5.2.48 void x2atm_help (atm_t * *atm*, double *zmin*, double *zmax*, double * *value*, gsl_vector * *x*, size_t * *n*)

Extract elements from state vector.

Definition at line 4926 of file jurassic.c.

```
04932               {
04933
04934   int ip;
04935
04936   /* Extract state vector elements... */
04937   for (ip = 0; ip < atm->np; ip++)
04938     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
04939       value[ip] = gsl_vector_get(x, *n);
04940       (*n)++;
04941     }
04942 }
```

5.5.2.49 void y2obs (ctl_t * *ctl*, gsl_vector * *y*, obs_t * *obs*)

Decompose measurement vector.

Definition at line 4946 of file jurassic.c.

```
04949               {
04950
04951   int id, ir;
04952
04953   size_t m = 0;
04954
04955   /* Decompose measurement vector... */
04956   for (ir = 0; ir < obs->nr; ir++)
04957     for (id = 0; id < ctl->nd; id++)
04958       if (gsl_finite(obs->rad[id][ir])) {
04959         obs->rad[id][ir] = gsl_vector_get(y, m);
04960         m++;
04961       }
04962 }
```

5.6 jurassic.c

```
00001 /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU General Public License
00015   along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2003-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 /*****************************************************************************/
00028
00029 size_t atm2x(
00030   ctl_t * ctl,
00031   atm_t * atm,
00032   gsl_vector * x,
00033   int *iqa,
00034   int *ipa) {
00035
00036   int ig, iw;
```

```
00037
00038   size_t n = 0;
00039
00040   /* Add pressure... */
00041   atm2x_help(atm, ctl->retp_zmin, ctl->retp_zmax,
00042              atm->p, IDXP, x, iqa, ipa, &n);
00043
00044   /* Add temperature... */
00045   atm2x_help(atm, ctl->rett_zmin, ctl->rett_zmax,
00046              atm->t, IDXT, x, iqa, ipa, &n);
00047
00048   /* Add volume mixing ratios... */
00049   for (ig = 0; ig < ctl->ng; ig++)
00050     atm2x_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
00051                atm->q[ig], IDXQ(ig), x, iqa, ipa, &n);
00052
00053   /* Add extinction... */
00054   for (iw = 0; iw < ctl->nw; iw++)
00055     atm2x_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
00056                atm->k[iw], IDXK(iw), x, iqa, ipa, &n);
00057
00058   return n;
00059 }
00060
00061 /*****************************************************************************/
00062
00063 void atm2x_help(
00064   atm_t * atm,
00065   double zmin,
00066   double zmax,
00067   double *value,
00068   int val_iqa,
00069   gsl_vector * x,
00070   int *iqa,
00071   int *ipa,
00072   size_t * n) {
00073
00074   int ip;
00075
00076   /* Add elements to state vector... */
00077   for (ip = 0; ip < atm->np; ip++)
00078     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
00079       if (x != NULL)
00080         gsl_vector_set(x, *n, value[ip]);
00081       if (iqa != NULL)
00082         iqa[*n] = val_iqa;
00083       if (ipa != NULL)
00084         ipa[*n] = ip;
00085       (*n)++;
00086     }
00087 }
00088
00089 /*****************************************************************************/
00090
00091 double brightness(
00092   double rad,
00093   double nu) {
00094
00095   return C2 * nu / gsl_log1p(C1 * POW3(nu) / rad);
00096 }
00097
00098
00099 /*****************************************************************************/
00100
00101 void cart2geo(
00102   double *x,
00103   double *z,
00104   double *lon,
00105   double *lat) {
00106
00107   double radius;
00108
00109   radius = NORM(x);
00110   *lat = asin(x[2] / radius) * 180 / M_PI;
00111   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00112   *z = radius - RE;
00113 }
00114
00115 /*****************************************************************************/
00116
00117 void climatology(
00118   ctl_t * ctl,
00119   atm_t * atm) {
00120
00121   static double z[121] = {
00122     0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00123     20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
```

```
00124     38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00125     56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00126     74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00127     92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00128     108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00129   };
00130
00131   static double pre[121] = {
00132     1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00133     357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00134     104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00135     29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00136     10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00137     3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00138     1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00139     0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00140     0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00141     0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00142     0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00143     0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00144     0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00145     0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00146     0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00147     0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00148     9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00149     4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00150     2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00151   };
00152
00153   static double tem[121] = {
00154     285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00155     229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00156     215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00157     222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00158     241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00159     262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00160     258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,
00161     237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00162     220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00163     207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00164     190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00165     178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00166     201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00167     272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00168   };
00169
00170   static double c2h2[121] = {
00171     1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00172     2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
00173     5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00174     2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00175     9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00176     1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00177     1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00178     1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00179     2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00180     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00181     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00182     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00183   };
00184
00185   static double c2h6[121] = {
00186     2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00187     1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00188     5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00189     2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00190     2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00191     1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00192     5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00193     2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00194     1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00195     7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,
00196     3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00197     1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00198     4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00199     1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00200     3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00201     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00202     0, 0, 0, 0, 0, 0, 0, 0, 0
00203   };
00204
00205   static double ccl4[121] = {
00206     1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00207     1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00208     8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00209     3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00210     3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
```

```
00211     4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00212     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00213     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00214     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00215     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00216     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00217     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00218     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00219     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00220     1e-14, 1e-14, 1e-14
00221   };
00222
00223   static double ch4[121] = {
00224     1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00225     1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00226     1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00227     1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00228     1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00229     8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00230     6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00231     4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00232     3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00233     2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00234     1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00235     1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00236     1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00237     9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00238     7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00239     5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00240     4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00241     3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00242     2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00243     2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00244     1.782e-08
00245   };
00246
00247   static double clo[121] = {
00248     7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00249     6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00250     8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00251     2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00252     1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00253     2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00254     4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00255     5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,
00256     3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00257     1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00258     6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00259     2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
00260     8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00261     3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00262     1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00263     3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00264     1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00265     3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00266     1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00267     5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00268     3.148e-15
00269   };
00270
00271   static double clono2[121] = {
00272     1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00273     1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00274     2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00275     2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00276     8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00277     6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00278     1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00279     1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00280     1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00281     1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00282     9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
00283     6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00284     3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00285     1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00286     8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00287     3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00288     9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00289     3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00290     2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00291     2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00292     4.041e-27
00293   };
00294
00295   static double co[121] = {
00296     1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00297     9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
```

```
00298     5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00299     2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00300     1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00301     2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00302     3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00303     3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00304     6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00305     2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00306     8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00307     2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00308     3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00309     6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00310     1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00311     1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00312     3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00313     5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00314     6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00315     7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05
00316   };
00317
00318   static double cof2[121] = {
00319     7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00320     6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00321     7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00322     4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00323     1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00324     1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00325     1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00326     8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00327     5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00328     2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00329     7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00330     1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00331     4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00332     1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00333     2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00334     4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
00335     7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00336     1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00337     3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00338     1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00339     4.662e-18
00340   };
00341
00342   static double f11[121] = {
00343     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00344     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00345     2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00346     1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
00347     7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00348     5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00349     1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00350     3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00351     6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00352     1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00353     1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00354     2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00355     2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00356     2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00357     2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00358     1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00359     1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00360     1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00361     2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00362     4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00363   };
00364
00365   static double f12[121] = {
00366     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00367     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00368     5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00369     4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
00370     2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00371     5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00372     2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00373     8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00374     3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00375     1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00376     8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00377     4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00378     2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00379     9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00380     4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00381     1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00382     7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00383     3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00384     1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
```

```
00385     1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00386   };
00387
00388   static double f14[121] = {
00389     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00390     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00391     8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00392     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00393     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00394     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00395     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00396     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00397     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00398     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00399     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00400     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00401     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00402     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00403     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00404     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00405     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00406   };
00407
00408   static double f22[121] = {
00409     1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00410     1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00411     1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00412     7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00413     4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00414     3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00415     1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00416     1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00417     8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00418     5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00419     4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00420     3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00421     3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
00422     2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00423     2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00424     2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00425     1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00426     1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00427     1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00428     1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00429   };
00430
00431   static double h2o[121] = {
00432     0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00433     0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
00434     6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00435     4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00436     4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00437     5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00438     5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00439     6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00440     6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00441     6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00442     5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00443     4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00444     3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00445     2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00446     1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00447     5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00448     1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00449     7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00450     3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00451     1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00452   };
00453
00454   static double h2o2[121] = {
00455     1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00456     4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
00457     3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00458     1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00459     8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00460     1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00461     1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00462     6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00463     5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00464     4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00465     3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00466     2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00467     1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00468     1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00469     9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00470     7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00471     4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
```

```
00472     3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00473     2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00474     2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00475     1.775e-12
00476   };
00477
00478   static double hcn[121] = {
00479     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00480     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00481     5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00482     1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00483     1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00484     1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00485     1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00486     1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00487     1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00488     9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00489     8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00490     7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00491     6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00492     6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00493     6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00494     6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00495     5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00496     5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00497     5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00498     5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00499   };
00500
00501   static double hno3[121] = {
00502     1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00503     2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00504     5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00505     3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00506     8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00507     3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00508     8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
00509     1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00510     6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00511     3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00512     1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00513     9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00514     5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00515     2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00516     1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
00517     5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00518     2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00519     1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00520     5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
00521     3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00522     2.332e-14
00523   };
00524
00525   static double hno4[121] = {
00526     6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00527     1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00528     3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00529     1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00530     2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00531     1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00532     3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00533     2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00534     1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00535     2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00536     1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00537     5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00538     2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00539     8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00540     3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00541     1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00542     3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00543     1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
00544     5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00545     2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00546     1.64e-18
00547   };
00548
00549   static double hocl[121] = {
00550     1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00551     2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00552     5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00553     1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00554     4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00555     7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00556     1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00557     6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00558     1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
```

```
00559     3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00560     6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00561     1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00562     1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00563     2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00564     3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00565     4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00566     5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00567     6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00568     9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00569     2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00570     7.881e-21
00571   };
00572
00573   static double n2o[121] = {
00574     3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00575     3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00576     2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00577     2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00578     1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00579     7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00580     2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00581     1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00582     5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00583     2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00584     1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00585     1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00586     9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00587     7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00588     5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00589     4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00590     3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00591     2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00592     2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00593     2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00594   };
00595
00596   static double n2o5[121] = {
00597     1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00598     1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00599     4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00600     7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00601     3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00602     2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00603     2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
00604     6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00605     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00606     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00607     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00608     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00609     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00610     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00611     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00612     1e-16, 1e-16
00613   };
00614
00615   static double nh3[121] = {
00616     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00617     1e-10, 1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00618     4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00619     5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00620     6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00621     1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00622     1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00623     1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00624     2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00625     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00626     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00627     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00628     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00629     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00630     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00631     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00632     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00633     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00634     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00635     1.914e-17
00636   };
00637
00638   static double no[121] = {
00639     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00640     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00641     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00642     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00643     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00644     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00645     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
```

```
00646     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00647     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00648     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00649     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00650     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00651     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00652     1.077e-08, 1.367e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00653     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00654     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00655     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00656     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00657     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00658     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00659     0.0001133
00660   };
00661
00662   static double no2[121] = {
00663     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00664     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00665     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00666     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00667     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00668     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00669     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00670     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00671     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00672     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00673     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00674     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00675     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00676     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00677     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00678     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00679     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00680     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00681   };
00682
00683   static double o3[121] = {
00684     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00685     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00686     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00687     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00688     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00689     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00690     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
00691     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00692     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00693     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00694     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
00695     4.509e-07, 3.922e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00696     2.476e-07, 2.284e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00697     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00698     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00699     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00700     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00701     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00702     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00703     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00704     3.665e-10
00705   };
00706
00707   static double ocs[121] = {
00708     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00709     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00710     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00711     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00712     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00713     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00714     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00715     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00716     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00717     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00718     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00719     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00720     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00721     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00722     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00723     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00724     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00725     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00726     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00727     1.091e-14, 1.091e-14, 1.091e-14
00728   };
00729
00730   static double sf6[121] = {
00731     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00732     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
```

```
00733     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00734     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00735     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00736     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00737     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00738     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00739     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00740     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00741     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00742     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00743     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00744     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00745     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00746     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00747     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00748     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00749     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00750   };
00751
00752   static double so2[121] = {
00753     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00754     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00755     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00756     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00757     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00758     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00759     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00760     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00761     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00762     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00763     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00764     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00765     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00766     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00767     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00768   };
00769
00770   static int ig_co2 = -999;
00771
00772   double co2, *q[NG] = { NULL };
00773
00774   int ig, ip, iw, iz;
00775
00776   /* Find emitter index of CO2... */
00777   if (ig_co2 == -999)
00778     ig_co2 = find_emitter(ctl, "CO2");
00779
00780   /* Identify variable... */
00781   for (ig = 0; ig < ctl->ng; ig++) {
00782     q[ig] = NULL;
00783     if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00784       q[ig] = c2h2;
00785     if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00786       q[ig] = c2h6;
00787     if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00788       q[ig] = ccl4;
00789     if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00790       q[ig] = ch4;
00791     if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00792       q[ig] = clo;
00793     if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00794       q[ig] = clono2;
00795     if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00796       q[ig] = co;
00797     if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00798       q[ig] = cof2;
00799     if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00800       q[ig] = f11;
00801     if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00802       q[ig] = f12;
00803     if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00804       q[ig] = f14;
00805     if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00806       q[ig] = f22;
00807     if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00808       q[ig] = h2o;
00809     if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00810       q[ig] = h2o2;
00811     if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00812       q[ig] = hcn;
00813     if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00814       q[ig] = hno3;
00815     if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00816       q[ig] = hno4;
00817     if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00818       q[ig] = hocl;
00819     if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
```

```
00820       q[ig] = n2o;
00821     if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00822       q[ig] = n2o5;
00823     if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00824       q[ig] = nh3;
00825     if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00826       q[ig] = no;
00827     if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00828       q[ig] = no2;
00829     if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00830       q[ig] = o3;
00831     if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00832       q[ig] = ocs;
00833     if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00834       q[ig] = sf6;
00835     if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00836       q[ig] = so2;
00837   }
00838
00839   /* Loop over atmospheric data points... */
00840   for (ip = 0; ip < atm->np; ip++) {
00841
00842     /* Get altitude index... */
00843     iz = locate_reg(z, 121, atm->z[ip]);
00844
00845     /* Interpolate pressure... */
00846     atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00847
00848     /* Interpolate temperature... */
00849     atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00850
00851     /* Interpolate trace gases... */
00852     for (ig = 0; ig < ctl->ng; ig++)
00853       if (q[ig] != NULL)
00854         atm->q[ig][ip] =
00855           LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00856       else
00857         atm->q[ig][ip] = 0;
00858
00859     /* Set CO2... */
00860     if (ig_co2 >= 0) {
00861       co2 =
00862         371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00863       atm->q[ig_co2][ip] = co2;
00864     }
00865
00866     /* Set extinction to zero... */
00867     for (iw = 0; iw < ctl->nw; iw++)
00868       atm->k[iw][ip] = 0;
00869   }
00870 }
00871
00872 /*****************************************************************************/
00873
00874 double ctmco2(
00875   double nu,
00876   double p,
00877   double t,
00878   double u) {
00879
00880   static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00881     1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00882     1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00883     1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00884     2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00885     3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00886     4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00887     5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00888     7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00889     .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00890     .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00891     .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00892     .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00893     .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00894     .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00895     .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00896     .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00897     .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00898     .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00899     .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00900     .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00901     .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00902     .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00903     .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00904     .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00905     .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00906     .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
```

```
00907     .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00908     .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,
00909     .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,
00910     1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,
00911     2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,
00912     3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,
00913     4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,
00914     7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,
00915     12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,
00916     21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,
00917     35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,
00918     60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,
00919     107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,
00920     199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,
00921     386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,
00922     756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,
00923     1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,
00924     3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,
00925     5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,
00926     2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,
00927     820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,
00928     1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,
00929     6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,
00930     4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,
00931     1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,
00932     999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,
00933     476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,
00934     251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,
00935     133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,
00936     73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,
00937     42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,
00938     25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,
00939     15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,
00940     9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,
00941     5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,
00942     3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,
00943     2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,
00944     1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,
00945     1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,
00946     .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,
00947     .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,
00948     .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,
00949     .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,
00950     .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,
00951     .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,
00952     .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,
00953     .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,
00954     .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,
00955     .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,
00956     .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,
00957     .099745, .091118, .083404, .076494, .070292, .064716, .059697,
00958     .055173, .051093, .047411, .044089, .041092, .038392, .035965,
00959     .033789, .031846, .030122, .028607, .02729, .026169, .025209,
00960     .024405, .023766, .023288, .022925, .022716, .022681, .022685,
00961     .022768, .023133, .023325, .023486, .024004, .024126, .024083,
00962     .023785, .024023, .023029, .021649, .021108, .019454, .017809,
00963     .017292, .016635, .017037, .018068, .018977, .018756, .017847,
00964     .016557, .016142, .014459, .012869, .012381, .010875, .0098701,
00965     .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,
00966     .014362, .015017, .016507, .017741, .01768, .017784, .0171,
00967     .016357, .016172, .017257, .018978, .020935, .021741, .023567,
00968     .025183, .025589, .026732, .027648, .028278, .028215, .02856,
00969     .029015, .029062, .028851, .028497, .027825, .027801, .026523,
00970     .02487, .022967, .022168, .020194, .018605, .017903, .018439,
00971     .019697, .020311, .020855, .020057, .018608, .016738, .015963,
00972     .013844, .011801, .011134, .0097573, .0086007, .0086226,
00973     .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,
00974     .014657, .015771, .016351, .016079, .014829, .013431, .013185,
00975     .013207, .01448, .016176, .017971, .018265, .019526, .020455,
00976     .019797, .019802, .0194, .018176, .017505, .016197, .015339,
00977     .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,
00978     .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,
00979     .0041501, .003752, .0033996, .0030865, .0028077, .0025586,
00980     .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,
00981     .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,
00982     8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,
00983     5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,
00984     3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,
00985     2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,
00986     2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,
00987     1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,
00988     1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,
00989     1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,
00990     1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,
00991     2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,
00992     2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,
00993     3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,
```

```
00994     4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,
00995     6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,
00996     8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
00997     .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
00998     .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
00999     .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01000     .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01001     .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01002     .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01003     .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01004     .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01005     .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01006     .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01007     .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01008     .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01009     .056326, .0602, .064093, .068521, .073278, .077734, .083064,
01010     .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,
01011     .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,
01012     .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,
01013     .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,
01014     .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,
01015     .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,
01016     .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,
01017     1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,
01018     1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,
01019     1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,
01020     2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,
01021     4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,
01022     7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,
01023     9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,
01024     10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,
01025     16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,
01026     27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,
01027     43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,
01028     63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,
01029     98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,
01030     159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,
01031     264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,
01032     449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,
01033     789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,
01034     1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,
01035     2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,
01036     4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,
01037     9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,
01038     20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,
01039     40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,
01040     43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,
01041     44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,
01042     21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,
01043     28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,
01044     31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,
01045     54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,
01046     29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,
01047     11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,
01048     5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,
01049     2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,
01050     1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,
01051     845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,
01052     489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,
01053     291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,
01054     176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,
01055     108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,
01056     67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,
01057     42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,
01058     27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,
01059     17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,
01060     11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,
01061     7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,
01062     4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,
01063     3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,
01064     2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,
01065     1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,
01066     .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,
01067     .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,
01068     .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,
01069     .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,
01070     .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,
01071     .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,
01072     .093203, .088815, .084641, .080671, .076892, .073296, .069873,
01073     .066613, .06351, .060555, .05774, .055058, .052504, .050071,
01074     .047752, .045543, .043438, .041432, .039521, .037699, .035962,
01075     .034307, .032729, .031225, .029791, .028423, .02712, .025877,
01076     .024692, .023563, .022485, .021458, .020478, .019543, .018652,
01077     .017802, .016992, .016219, .015481, .014778, .014107, .013467,
01078     .012856, .012274, .011718, .011188, .010682, .0102, .0097393,
01079     .0093001, .008881, .0084812, .0080997, .0077358, .0073885,
01080     .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,
```

```
01081     .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,
01082     .0040884, .0039088, .0037373, .0035739, .003418, .0032693,
01083     .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,
01084     .0024074, .0023066, .0022108, .0021196, .002033, .0019507,
01085     .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,
01086     .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,
01087     .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,
01088     .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,
01089     9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,
01090     .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,
01091     .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,
01092     .0016144, .0016312, .001705, .0017892, .0018499, .0019715,
01093     .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,
01094     .0031193, .003346, .0034552, .0036906, .0037584, .0040084,
01095     .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,
01096     .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,
01097     .0060972, .0055539, .0055653, .0055772, .005331, .0054953,
01098     .0055919, .0058684, .006183, .0066675, .0069808, .0075142,
01099     .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,
01100     .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,
01101     .0105, .010617, .010706, .01078, .011177, .011212, .011304,
01102     .011446, .011603, .011816, .012165, .012545, .013069, .013539,
01103     .01411, .014776, .016103, .017016, .017994, .018978, .01998,
01104     .021799, .022745, .023681, .024627, .025562, .026992, .027958,
01105     .029013, .030154, .031402, .03228, .033651, .035272, .037088,
01106     .039021, .041213, .043597, .045977, .04877, .051809, .054943,
01107     .058064, .061528, .06537, .069309, .071928, .075752, .079589,
01108     .083352, .084096, .087497, .090817, .091198, .094966, .099045,
01109     .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,
01110     .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,
01111     .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,
01112     .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01113     .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01114     .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01115     1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01116     2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01117     3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01118     5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01119     10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01120     18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01121     30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01122     57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01123     107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01124     205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01125     394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01126     501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01127     455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01128     376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01129     367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01130     588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01131     962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01132     1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01133     751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01134     777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01135     322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01136     654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01137     679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01138     818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01139     364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01140     155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01141     74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01142     38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01143     21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01144     12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01145     7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01146     4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01147     2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01148     1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01149     1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01150     .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01151     .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01152     .26732, .25337, .24017, .22774, .21601, .20479, .19426
01153   };
01154
01155   static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01156     6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01157     9.2421e-5, 9.6846e-5, 1.0149e-4, 1.0635e-4, 1.1145e-4, 1.1679e-4,
01158     1.224e-4, 1.2828e-4, 1.3444e-4, 1.409e-4, 1.4768e-4, 1.5479e-4,
01159     1.6224e-4, 1.7006e-4, 1.7826e-4, 1.8685e-4, 1.9587e-4, 2.0532e-4,
01160     2.1524e-4, 2.2565e-4, 2.3656e-4, 2.48e-4, 2.6001e-4, 2.7261e-4,
01161     2.8582e-4, 2.9968e-4, 3.1422e-4, 3.2948e-4, 3.4548e-4, 3.6228e-4,
01162     3.799e-4, 3.9838e-4, 4.1778e-4, 4.3814e-4, 4.595e-4, 4.8191e-4,
01163     5.0543e-4, 5.3012e-4, 5.5603e-4, 5.8321e-4, 6.1175e-4, 6.417e-4,
01164     6.7314e-4, 7.0614e-4, 7.4078e-4, 7.7714e-4, 8.1531e-4, 8.5538e-4,
01165     8.9745e-4, 9.4162e-4, 9.8798e-4, .0010367, .0010878, .0011415,
01166     .0011978, .001257, .0013191, .0013844, .001453, .0015249,
01167     .0016006, .00168, .0017634, .001851, .001943, .0020397, .0021412,
```

```
01168     .0022479, .00236, .0024778, .0026015, .0027316, .0028682,
01169     .0030117, .0031626, .0033211, .0034877, .0036628, .0038469,
01170     .0040403, .0042436, .0044574, .004682, .0049182, .0051665,
01171     .0054276, .0057021, .0059907, .0062942, .0066133, .0069489,
01172     .0073018, .0076729, .0080632, .0084738, .0089056, .0093599,
01173     .0098377, .01034, .010869, .011426, .012011, .012627, .013276,
01174     .013958, .014676, .015431, .016226, .017063, .017944, .018872,
01175     .019848, .020876, .021958, .023098, .024298, .025561, .026892,
01176     .028293, .029769, .031323, .032961, .034686, .036503, .038418,
01177     .040435, .042561, .044801, .047161, .049649, .052271, .055035,
01178     .057948, .061019, .064256, .06767, .07127, .075066, .079069,
01179     .083291, .087744, .092441, .097396, .10262, .10814, .11396,
01180     .1201, .12658, .13342, .14064, .14826, .1563, .1648, .17376,
01181     .18323, .19324, .2038, .21496, .22674, .23919, .25234, .26624,
01182     .28093, .29646, .31287, .33021, .34855, .36794, .38844, .41012,
01183     .43305, .45731, .48297, .51011, .53884, .56924, .60141, .63547,
01184     .67152, .70969, .75012, .79292, .83826, .8863, .93718, .99111,
01185     1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,
01186     1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,
01187     2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,
01188     4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,
01189     6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,
01190     11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,
01191     18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,
01192     31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,
01193     54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,
01194     97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,
01195     182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,
01196     356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,
01197     716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,
01198     1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,
01199     3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,
01200     6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,
01201     2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,
01202     843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,
01203     1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381.,
01204     9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,
01205     4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,
01206     2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,
01207     1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,
01208     525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,
01209     268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,
01210     139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,
01211     76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,
01212     43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,
01213     25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,
01214     14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,
01215     9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,
01216     5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,
01217     3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,
01218     2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,
01219     1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,
01220     .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,
01221     .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,
01222     .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,
01223     .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,
01224     .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,
01225     .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,
01226     .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,
01227     .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,
01228     .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,
01229     .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,
01230     .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,
01231     .1152, .10304, .092437, .083163, .075031, .067878, .061564,
01232     .055976, .051018, .046609, .042679, .03917, .036032, .033223,
01233     .030706, .02845, .026428, .024617, .022998, .021554, .02027,
01234     .019136, .018141, .017278, .016541, .015926, .015432, .015058,
01235     .014807, .014666, .014635, .014728, .014947, .01527, .015728,
01236     .016345, .017026, .017798, .018839, .019752, .020636, .021886,
01237     .022695, .02327, .023478, .024292, .023544, .022222, .021932,
01238     .020052, .018143, .017722, .017031, .017782, .01938, .020734,
01239     .020476, .019255, .017477, .016878, .014617, .012489, .011765,
01240     .0099077, .0086446, .0079446, .0078644, .0079763, .008671,
01241     .01001, .0108, .012933, .015349, .016341, .018484, .020254,
01242     .020254, .020478, .019591, .018595, .018385, .019913, .022254,
01243     .024847, .025809, .028053, .029924, .030212, .031367, .03222,
01244     .032739, .032537, .03286, .033344, .033507, .033499, .033339,
01245     .032809, .033041, .031723, .029837, .027511, .026603, .024032,
01246     .021914, .020948, .021701, .023425, .024259, .024987, .023818,
01247     .021768, .019223, .018144, .015282, .012604, .01163, .0097907,
01248     .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,
01249     .014378, .016761, .01726, .018997, .019998, .019809, .01819,
01250     .016358, .016099, .01617, .017939, .020223, .022521, .02277,
01251     .024279, .025247, .024222, .023989, .023224, .021493, .020362,
01252     .018596, .017309, .015975, .014466, .013171, .011921, .01078,
01253     .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,
01254     .0051459, .0046273, .0041712, .0037686, .0034119, .003095,
```

```
01255     .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,
01256     .001636, .0015017, .00138, .0012697, .0011694, .0010782,
01257     9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,
01258     6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,
01259     4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,
01260     2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,
01261     1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,
01262     1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,
01263     1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,
01264     1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,
01265     1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,
01266     1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,
01267     1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,
01268     2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,
01269     2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,
01270     3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,
01271     4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,
01272     6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,
01273     8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,
01274     .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,
01275     .0014953, .001572, .0016529, .0017381, .0018279, .0019226,
01276     .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,
01277     .002747, .0028921, .0030453, .0032071, .003378, .0035586,
01278     .0037494, .003951, .0041642, .0043897, .0046282, .0048805,
01279     .0051476, .0054304, .00573, .0060473, .0063837, .0067404,
01280     .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,
01281     .0099366, .010516, .011134, .011792, .012494, .013244, .014046,
01282     .014898, .015808, .016781, .017822, .018929, .020108, .02138,
01283     .022729, .02419, .02576, .027412, .029233, .031198, .033301,
01284     .035594, .038092, .040767, .04372, .046918, .050246, .053974,
01285     .058009, .061976, .066586, .071537, .076209, .081856, .087998,
01286     .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,
01287     .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,
01288     .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,
01289     .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,
01290     .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,
01291     .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,
01292     .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,
01293     .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,
01294     1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,
01295     1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,
01296     2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,
01297     5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,
01298     7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,
01299     10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,
01300     11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,
01301     18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,
01302     29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,
01303     46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,
01304     70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,
01305     114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,
01306     190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,
01307     324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,
01308     568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,
01309     1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543., 1672.8, 1813.4,
01310     1966.1, 2131.4, 2309.5, 2499.3, 2705., 2925.7, 3161.6, 3411.3,
01311     3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,
01312     6719.6, 7385., 8145., 8977.7, 9831.9, 10827., 11934., 13063.,
01313     14434., 15878., 17591., 19435., 21510., 23835., 26835., 29740.,
01314     32878., 36305., 39830., 43273., 46931., 50499., 49586., 51598.,
01315     53429., 54619., 55081., 55102., 54485., 53487., 52042., 42689.,
01316     42607., 44020., 47994., 54169., 53916., 55808., 56642., 46049.,
01317     44243., 32929., 30658., 21963., 20835., 15962., 13679., 17652.,
01318     19680., 22388., 25625., 29184., 32520., 35720., 38414., 40523.,
01319     49228., 48173., 45678., 41768., 37600., 41313., 42654., 44465.,
01320     55736., 56630., 65409., 63308., 66572., 61845., 60379., 56777.,
01321     51920., 46601., 41367., 36529., 32219., 28470., 25192., 22362.,
01322     19907., 17772., 15907., 14273., 12835., 11567., 10445., 9450.2,
01323     8565.1, 7776., 7070.8, 6439.2, 5872.3, 5362.4, 4903., 4488.3,
01324     4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296.,
01325     2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236.,
01326     1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,
01327     643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,
01328     371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,
01329     219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,
01330     131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,
01331     80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,
01332     49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,
01333     30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,
01334     19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,
01335     12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,
01336     7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,
01337     5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,
01338     3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,
01339     2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,
01340     1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,
01341     .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,
```

```
01342     .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,
01343     .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,
01344     .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01345     .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01346     .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01347     .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01348     .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01349     .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01350     .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01351     .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01352     .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01353     .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01354     .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01355     .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01356     .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01357     .003388, .0032329, .0030852, .0029445, .0028105, .0026829,
01358     .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01359     .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
01360     .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01361     .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01362     9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01363     7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01364     6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
01365     6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01366     7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01367     9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01368     .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01369     .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01370     .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01371     .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01372     .0053809, .0056699, .0059325, .0055488, .005634, .0056392,
01373     .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01374     .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01375     .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01376     .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01377     .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01378     .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01379     .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01380     .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01381     .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01382     .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01383     .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01384     .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01385     .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01386     .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01387     .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01388     .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01389     .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01390     .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01391     1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01392     1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01393     2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01394     4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01395     8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01396     15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01397     27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01398     46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01399     87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01400     176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01401     366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01402     600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01403     478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01404     545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01405     314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01406     696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01407     954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01408     1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01409     1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01410     1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01411     361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01412     578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01413     659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01414     1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01415     489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01416     199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01417     92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01418     46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01419     24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01420     13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01421     7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01422     4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01423     2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01424     1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01425     1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01426     .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01427     .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01428     .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
```

```
01429     .16469
01430   };
01431
01432   static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01433     3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01434     4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01435     6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01436     8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01437     1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01438     1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01439     1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01440     2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01441     3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01442     4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01443     6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01444     9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
01445     .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01446     .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
01447     .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01448     .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01449     .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01450     .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01451     .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
01452     .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01453     .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01454     .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01455     .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01456     .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01457     .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01458     .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01459     .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,
01460     .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01461     .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01462     .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01463     1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01464     1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01465     2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01466     4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01467     7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01468     13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01469     22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01470     40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01471     74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01472     137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01473     263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01474     525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01475     1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01476     2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01477     5457.9, 5946.2, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01478     11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01479     1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01480     722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01481     2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01482     7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01483     4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01484     1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01485     783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01486     387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01487     197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01488     105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01489     57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01490     31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01491     17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01492     10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01493     6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01494     3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01495     2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01496     1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01497     .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01498     .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01499     .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01500     .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01501     .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01502     .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01503     .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01504     .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01505     .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01506     .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01507     .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01508     .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01509     .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01510     .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01511     .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01512     .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01513     .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01514     .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01515     .022311, .023742, .0249, .025599, .026981, .026462, .025143,
```

```
01516     .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01517     .024163, .023728, .02199, .019506, .018591, .015576, .012784,
01518     .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01519     .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01520     .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01521     .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01522     .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01523     .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01524     .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01525     .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01526     .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01527     .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01528     .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01529     .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01530     .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01531     .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
01532     .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,
01533     .0031747, .0028864, .0026284, .002397, .002189, .0020017,
01534     .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,
01535     .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,
01536     6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,
01537     4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,
01538     2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,
01539     1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,
01540     1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,
01541     9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,
01542     7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,
01543     6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,
01544     7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,
01545     8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,
01546     9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,
01547     1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,
01548     1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,
01549     2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,
01550     3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,
01551     4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,
01552     5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,
01553     7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,
01554     .0010922, .001154, .0012195, .0012889, .0013626, .0014407,
01555     .0015235, .0016114, .0017048, .0018038, .001909, .0020207,
01556     .0021395, .0022657, .0023998, .0025426, .0026944, .002856,
01557     .0030281, .0032114, .0034068, .003615, .0038371, .004074,
01558     .004327, .0045971, .0048857, .0051942, .0055239, .0058766,
01559     .0062538, .0066573, .0070891, .007551, .0080455, .0085747,
01560     .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,
01561     .014415, .01541, .016475, .017621, .018857, .020175, .02162,
01562     .023185, .024876, .02672, .028732, .030916, .033319, .035939,
01563     .038736, .041847, .04524, .048715, .052678, .056977, .061203,
01564     .066184, .07164, .076952, .083477, .090674, .098049, .10697,
01565     .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,
01566     .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,
01567     .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,
01568     .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,
01569     .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,
01570     .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,
01571     .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,
01572     1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,
01573     1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,
01574     1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,
01575     3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,
01576     6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,
01577     8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,
01578     7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,
01579     11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,
01580     21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,
01581     35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,
01582     51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,
01583     83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,
01584     142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,
01585     249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,
01586     449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,
01587     833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350., 1466.3,
01588     1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,
01589     3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,
01590     5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203.,
01591     11223., 12358., 13493., 14916., 16416., 18236., 20222., 22501.,
01592     25102., 28358., 31707., 35404., 39538., 43911., 48391., 53193.,
01593     58028., 58082., 61276., 64193., 66294., 67480., 67921., 67423.,
01594     66254., 64341., 51737., 51420., 53072., 58145., 66195., 65358.,
01595     67377., 67869., 53509., 50553., 35737., 32425., 21704., 19974.,
01596     14457., 12142., 16798., 19489., 23049., 27270., 31910., 36457.,
01597     40877., 44748., 47876., 59793., 58626., 55454., 50337., 44893.,
01598     50228., 52216., 54747., 69541., 70455., 81014., 77694., 80533.,
01599     73953., 70927., 65539., 59002., 52281., 45953., 40292., 35360.,
01600     31124., 27478., 24346., 21647., 19308., 17271., 15491., 13927.,
01601     12550., 11331., 10250., 9288.8, 8431.4, 7664.9, 6978.3, 6361.8,
01602     5807.4, 5307.7, 4856.8, 4449., 4079.8, 3744.9, 3440.8, 3164.2,
```

```
01603     2912.3, 2682.7, 2473., 2281.4, 2106., 1945.3, 1797.9, 1662.5,
01604     1538.1, 1423.6, 1318.1, 1221., 1131.5, 1049., 972.99, 902.87,
01605     838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,
01606     462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,
01607     258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,
01608     147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,
01609     86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,
01610     51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,
01611     30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,
01612     18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,
01613     11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,
01614     7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,
01615     4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,
01616     2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,
01617     1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,
01618     1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,
01619     .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,
01620     .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,
01621     .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,
01622     .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,
01623     .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,
01624     .092236, .087844, .083443, .07938, .075452, .071564, .067931,
01625     .064389, .061078, .057901, .054921, .052061, .049364, .046789,
01626     .04435, .042044, .039866, .037808, .035863, .034023, .032282,
01627     .030634, .029073, .027595, .026194, .024866, .023608, .022415,
01628     .021283, .02021, .019193, .018228, .017312, .016443, .015619,
01629     .014837, .014094, .01339, .012721, .012086, .011483, .010911,
01630     .010368, .009852, .0093623, .0088972, .0084556, .0080362,
01631     .0076379, .0072596, .0069003, .006559, .0062349, .0059269,
01632     .0056344, .0053565, .0050925, .0048417, .0046034, .004377,
01633     .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,
01634     .0030785, .002928, .0027851, .0026492, .0025201, .0023975,
01635     .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,
01636     .0016938, .0016122, .0015348, .0014612, .0013913, .001325,
01637     .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,
01638     9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,
01639     7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,
01640     5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,
01641     4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,
01642     3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,
01643     3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,
01644     3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,
01645     4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,
01646     6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,
01647     8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,
01648     .001282, .0014016, .0015355, .0016764, .0018272, .0020055,
01649     .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,
01650     .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,
01651     .0052989, .0056148, .0052452, .0053357, .005333, .0045069,
01652     .0043851, .004253, .003738, .0038084, .0039013, .0041505,
01653     .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,
01654     .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,
01655     .0076269, .0063758, .006254, .0067749, .0067909, .0068231,
01656     .0072143, .0072762, .0072954, .007679, .0075107, .0073658,
01657     .0072441, .0071074, .0070378, .007176, .0072472, .0075844,
01658     .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,
01659     .013895, .015567, .016011, .016392, .016737, .017043, .017731,
01660     .018031, .018419, .018877, .019474, .019868, .020604, .021538,
01661     .022653, .023869, .025288, .026879, .028547, .030524, .03274,
01662     .035132, .03769, .040567, .043793, .047188, .049962, .053542,
01663     .057205, .060776, .061489, .064419, .067124, .065945, .068487,
01664     .071209, .074783, .077039, .082444, .08902, .09692, .10617,
01665     .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,
01666     .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,
01667     .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,
01668     .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,
01669     .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,
01670     .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,
01671     1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,
01672     2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,
01673     4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,
01674     8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,
01675     15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,
01676     28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,
01677     46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,
01678     86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,
01679     179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,
01680     397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,
01681     719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,
01682     580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,
01683     615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,
01684     316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,
01685     813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,
01686     1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,
01687     1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,
01688     1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,
01689     1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,
```

```
01690     338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.85,
01691     662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,
01692     803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01693     1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01694     523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01695     211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01696     97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01697     49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01698     26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01699     14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01700     7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01701     4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01702     2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01703     1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01704     .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01705     .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
01706     .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01707     .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
01708     .12584
01709   };
01710
01711   double xw, dw, ew, cw296, cw260, cw230, dt230, dt260, dt296, ctw, ctmpth;
01712
01713   int iw;
01714
01715   /* Get CO2 continuum absorption... */
01716   xw = nu / 2 + 1;
01717   if (xw >= 1 && xw < 2001) {
01718     iw = (int) xw;
01719     dw = xw - iw;
01720     ew = 1 - dw;
01721     cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01722     cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01723     cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01724     dt230 = t - 230;
01725     dt260 = t - 260;
01726     dt296 = t - 296;
01727     ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01728       * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01729     ctmpth = u / NA / 1000 * p / P0 * ctw;
01730   } else
01731     ctmpth = 0;
01732   return ctmpth;
01733 }
01734
01735 /*****************************************************************************/
01736
01737 double ctmh2o(
01738   double nu,
01739   double p,
01740   double t,
01741   double q,
01742   double u) {
01743
01744   static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01745     .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01746     .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01747     .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01748     .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01749     .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01750     .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01751     6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01752     3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01753     1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01754     1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01755     6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01756     4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01757     3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01758     2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01759     1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01760     1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01761     1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01762     1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01763     1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01764     2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01765     4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01766     1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01767     2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01768     3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01769     3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01770     3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,
01771     2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,
01772     1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,
01773     4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,
01774     2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,
01775     1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,
01776     5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,
```

```
01777     2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,
01778     1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,
01779     7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,
01780     4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,
01781     2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,
01782     1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,
01783     1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,
01784     9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,
01785     7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,
01786     6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,
01787     5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,
01788     5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,
01789     5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,
01790     7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,
01791     1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,
01792     3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,
01793     6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,
01794     1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,
01795     1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,
01796     1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,
01797     1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,
01798     1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,
01799     3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,
01800     7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,
01801     1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,
01802     3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,
01803     7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,
01804     1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,
01805     1.387e-4, 1.363e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,
01806     1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,
01807     5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,
01808     2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,
01809     1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,
01810     5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,
01811     2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,
01812     1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,
01813     6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,
01814     3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,
01815     1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,
01816     1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,
01817     6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,
01818     4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,
01819     3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,
01820     2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,
01821     2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,
01822     2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,
01823     4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,
01824     8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,
01825     1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,
01826     3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,
01827     8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,
01828     2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,
01829     4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,
01830     8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,
01831     1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,
01832     9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,
01833     4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,
01834     1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,
01835     9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,
01836     4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,
01837     2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,
01838     1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,
01839     5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,
01840     2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,
01841     1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,
01842     8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,
01843     5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,
01844     3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,
01845     2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,
01846     2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,
01847     1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,
01848     2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,
01849     3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,
01850     5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,
01851     1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,
01852     2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,
01853     5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,
01854     1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,
01855     2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,
01856     3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,
01857     4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,
01858     5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01859     7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01860     1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01861     2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01862     4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01863     5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
```

```
01864     4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01865     2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01866     9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01867     4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01868     2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01869     1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01870     5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01871     2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,
01872     1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,
01873     7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,
01874     4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,
01875     3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,
01876     2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,
01877     2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,
01878     2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,
01879     4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,
01880     6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,
01881     9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,
01882     1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,
01883     1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,
01884     1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,
01885     2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,
01886     6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,
01887     1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,
01888     2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,
01889     2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,
01890     2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,
01891     1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,
01892     4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,
01893     2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,
01894     1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,
01895     5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,
01896     2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,
01897     1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,
01898     7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,
01899     4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,
01900     2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,
01901     1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,
01902     1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,
01903     1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,
01904     1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,
01905     2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,
01906     2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,
01907     3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,
01908     3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,
01909     6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,
01910     1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,
01911     3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,
01912     6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,
01913     1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,
01914     2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,
01915     2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,
01916     3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,
01917     7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,
01918     1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,
01919     1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,
01920     1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,
01921     5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,
01922     2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,
01923     1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,
01924     1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,
01925     1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,
01926     1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,
01927     1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,
01928     5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,
01929     2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,
01930     1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,
01931     5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,
01932     3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,
01933     1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,
01934     9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,
01935     7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,
01936     8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,
01937     1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,
01938     2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,
01939     3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,
01940     6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,
01941     1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,
01942     2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,
01943     4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,
01944     7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,
01945     7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,
01946     5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,
01947     2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,
01948     1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,
01949     8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,
01950     7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,
```

```
01951     7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,
01952     6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,
01953     3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,
01954     1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,
01955     7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,
01956     3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,
01957     1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,
01958     1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,
01959     6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,
01960     4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,
01961     4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,
01962     7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,
01963     1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,
01964     3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,
01965     7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,
01966     1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,
01967     3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,
01968     7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,
01969     1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,
01970     2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,
01971     4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,
01972     5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,
01973     5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,
01974     3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,
01975     1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,
01976     8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,
01977     5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,
01978     5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,
01979     8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,
01980     1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,
01981     1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,
01982     7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,
01983     3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,
01984     1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,
01985     7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,
01986     4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,
01987     2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,
01988     1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,
01989     6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,
01990     6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,
01991     8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,
01992     1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,
01993     3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,
01994     6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,
01995     1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,
01996     2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,
01997     3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,
01998     4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,
01999     3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,
02000     1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,
02001     7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,
02002     4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,
02003     3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,
02004     4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,
02005     7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,
02006     8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,
02007     6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,
02008     3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,
02009     1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,
02010     6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,
02011     3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,
02012     1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,
02013     1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,
02014     1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,
02015     2.534e-12, 2.858e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,
02016     5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,
02017     1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,
02018     2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,
02019     5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,
02020     1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,
02021     2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,
02022     4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,
02023     6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,
02024     5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,
02025     3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,
02026     1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,
02027     7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,
02028     4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,
02029     2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,
02030     2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,
02031     4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,
02032     6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,
02033     8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,
02034     8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,
02035     5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,
02036     2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,
02037     1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,
```

```
02038     5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,
02039     2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,
02040     1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,
02041     1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,
02042     1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,
02043     2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,
02044     5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,
02045     1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,
02046     2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,
02047     3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,
02048     4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,
02049     3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,
02050     2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,
02051     1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,
02052     5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,
02053     2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,
02054     1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,
02055     1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,
02056     1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,
02057     2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,
02058     3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,
02059     3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,
02060     2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,
02061     1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,
02062     5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,
02063     4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,
02064     5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,
02065     9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,
02066     2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,
02067     4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,
02068     1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,
02069     2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,
02070     4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,
02071     7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,
02072     7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,
02073     5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,
02074     2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02075     1.093e-11, 9.558e-12
02076   };
02077
02078   static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02079     .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02080     .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02081     .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02082     .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02083     .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02084     .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02085     .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02086     5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02087     3.366e-4, 3.085e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02088     2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02089     1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02090     9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02091     6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02092     4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02093     3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02094     2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02095     2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02096     2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02097     2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02098     3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02099     7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02100     1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02101     3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02102     5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02103     6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02104     5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02105     2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02106     1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02107     6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02108     2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02109     1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02110     7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02111     4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02112     2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02113     1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02114     9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02115     6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02116     4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02117     3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02118     2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
02119     1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,
02120     1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,
02121     1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,
02122     1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,
02123     1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,
02124     1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,
```

```
02125     2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,
02126     5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,
02127     1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,
02128     1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,
02129     2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,
02130     2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,
02131     2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,
02132     2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,
02133     5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,
02134     1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,
02135     2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,
02136     5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,
02137     1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,
02138     1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,
02139     2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,
02140     1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,
02141     7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,
02142     2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,
02143     1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,
02144     7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,
02145     3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,
02146     1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,
02147     1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,
02148     5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,
02149     3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,
02150     2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,
02151     1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,
02152     9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,
02153     7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,
02154     5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,
02155     5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,
02156     6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,
02157     9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,
02158     1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,
02159     3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,
02160     6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,
02161     1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,
02162     2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,
02163     6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,
02164     1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,
02165     1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,
02166     1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,
02167     7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,
02168     3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,
02169     1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,
02170     7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,
02171     3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,
02172     1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,
02173     9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,
02174     5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,
02175     3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,
02176     1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,
02177     1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,
02178     9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,
02179     6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,
02180     4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,
02181     4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,
02182     4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,
02183     5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,
02184     9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,
02185     1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,
02186     3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,
02187     7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,
02188     1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,
02189     3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,
02190     5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,
02191     7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,
02192     7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,
02193     8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,
02194     1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,
02195     3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,
02196     7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,
02197     8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,
02198     6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,
02199     2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,
02200     1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,
02201     6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,
02202     3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,
02203     1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,
02204     8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,
02205     4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,
02206     2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,
02207     1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,
02208     9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,
02209     6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,
02210     5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,
02211     4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,
```

```
02212     4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,
02213     6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,
02214     1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,
02215     1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,
02216     1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,
02217     1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,
02218     2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,
02219     3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,
02220     8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,
02221     2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,
02222     3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,
02223     4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,
02224     3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,
02225     1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,
02226     6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,
02227     3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,
02228     1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,
02229     9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,
02230     4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,
02231     2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,
02232     1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,
02233     7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,
02234     5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,
02235     3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,
02236     2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,
02237     2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,
02238     2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,
02239     3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,
02240     4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,
02241     5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,
02242     6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,
02243     1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,
02244     2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,
02245     4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,
02246     9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,
02247     2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,
02248     3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,
02249     3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,
02250     5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,
02251     1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,
02252     1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,
02253     2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,
02254     1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,
02255     6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,
02256     3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,
02257     2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,
02258     2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,
02259     2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,
02260     2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,
02261     1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,
02262     6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,
02263     3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,
02264     1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,
02265     8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,
02266     4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,
02267     2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,
02268     1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,
02269     1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,
02270     1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,
02271     2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,
02272     4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,
02273     6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,
02274     9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,
02275     1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,
02276     3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,
02277     7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,
02278     1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,
02279     1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,
02280     7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,
02281     2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,
02282     1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,
02283     1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,
02284     1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,
02285     1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,
02286     9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,
02287     5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,
02288     2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,
02289     1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,
02290     5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,
02291     3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,
02292     1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,
02293     1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,
02294     8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,
02295     9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,
02296     1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,
02297     3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,
02298     5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,
```

```
02299     1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,
02300     2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,
02301     5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,
02302     1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,
02303     2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,
02304     3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,
02305     7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,
02306     9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,
02307     8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,
02308     4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,
02309     2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,
02310     1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,
02311     8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,
02312     9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,
02313     1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,
02314     1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,
02315     1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,
02316     9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,
02317     4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,
02318     2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,
02319     1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,
02320     5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,
02321     3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,
02322     1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,
02323     1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,
02324     1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,
02325     1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,
02326     2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,
02327     5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,
02328     8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,
02329     1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,
02330     4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,
02331     6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,
02332     7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,
02333     5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,
02334     2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,
02335     1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,
02336     6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,
02337     5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,
02338     8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,
02339     1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,
02340     1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,
02341     8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,
02342     3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,
02343     1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,
02344     9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,
02345     5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,
02346     2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,
02347     2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,
02348     2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,
02349     4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,
02350     9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,
02351     1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,
02352     3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,
02353     8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,
02354     2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,
02355     4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,
02356     7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,
02357     9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,
02358     8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,
02359     5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,
02360     2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,
02361     1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,
02362     5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,
02363     3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,
02364     4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,
02365     6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,
02366     1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,
02367     1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,
02368     1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,
02369     7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,
02370     3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,
02371     1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,
02372     8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,
02373     4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,
02374     2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,
02375     2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,
02376     2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,
02377     5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,
02378     1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,
02379     2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,
02380     4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02381     7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02382     9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02383     8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02384     5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02385     2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
```

```
02386     9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02387     5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02388     2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02389     2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02390     3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02391     6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02392     9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02393     9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
02394     5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02395     2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02396     1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02397     1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
02398     1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02399     3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02400     7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02401     1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02402     3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
02403     8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02404     1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02405     2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02406     2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02407     1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02408     7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02409     3.511e-11
02410   };
02411
02412   static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02413     .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02414     .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02415     .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02416     4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02417     1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02418     3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02419     1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02420     4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02421     1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,
02422     7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,
02423     2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,
02424     1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,
02425     5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,
02426     2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,
02427     1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,
02428     5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,
02429     2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,
02430     3.413e-9, 3.738e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,
02431     8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,
02432     3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,
02433     1.889e-7, 2.589e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,
02434     1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,
02435     7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,
02436     2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,
02437     4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,
02438     2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,
02439     3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,
02440     2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,
02441     8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,
02442     2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,
02443     7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,
02444     2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,
02445     7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,
02446     2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,
02447     8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,
02448     3.157e-9, 2.725e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,
02449     1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,
02450     5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,
02451     2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,
02452     1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,
02453     5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,
02454     1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,
02455     9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,
02456     1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,
02457     1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,
02458     1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,
02459     6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,
02460     1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,
02461     1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,
02462     7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,
02463     1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,
02464     7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,
02465     1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,
02466     7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,
02467     4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,
02468     5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,
02469     1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,
02470     4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,
02471     1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,
02472     7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,
```

```
02473     1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,
02474     1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,
02475     1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,
02476     6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,
02477     8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,
02478     1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,
02479     4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,
02480     1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,
02481     6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,
02482     2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,
02483     9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,
02484     3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,
02485     1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,
02486     9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,
02487     2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,
02488     3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,
02489     2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,
02490     2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,
02491     1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,
02492     1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,
02493     1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,
02494     4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,
02495     1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,
02496     7.611e-9, 9.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,
02497     4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,
02498     3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,
02499     7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,
02500     8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,
02501     1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,
02502     7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,
02503     1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,
02504     2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,
02505     6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,
02506     2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,
02507     8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,
02508     3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,
02509     1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,
02510     5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,
02511     2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,
02512     2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,
02513     1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,
02514     6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,
02515     6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,
02516     6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,
02517     3.311e-12, 3.346e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,
02518     4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,
02519     1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,
02520     4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,
02521     2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,
02522     1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,
02523     4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,
02524     1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,
02525     2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,
02526     3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,
02527     4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,
02528     3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,
02529     7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,
02530     2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,
02531     4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,
02532     4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,
02533     5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,
02534     2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,
02535     3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,
02536     7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,
02537     3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,
02538     1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,
02539     4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,
02540     1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,
02541     5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,
02542     2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,
02543     9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,
02544     4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,
02545     3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,
02546     1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,
02547     9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,
02548     2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,
02549     4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,
02550     6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,
02551     6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,
02552     9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,
02553     5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,
02554     4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02555     2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02556     1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02557     1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02558     2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02559     2.859e-8, 2.839e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
```

```
02560     1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02561     2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02562     8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02563     3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02564     1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02565     6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02566     1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02567     6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
02568     2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,
02569     7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,
02570     2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,
02571     1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,
02572     4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,
02573     4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,
02574     1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,
02575     1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,
02576     2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,
02577     1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,
02578     1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,
02579     1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,
02580     5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,
02581     3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,
02582     9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,
02583     1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,
02584     1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,
02585     2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,
02586     5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,
02587     9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,
02588     1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,
02589     1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,
02590     6.403e-9, 4.643e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,
02591     9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,
02592     4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,
02593     9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,
02594     1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,
02595     1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,
02596     1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,
02597     6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,
02598     7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,
02599     1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,
02600     5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,
02601     1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,
02602     5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,
02603     2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,
02604     1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,
02605     8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,
02606     5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,
02607     1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,
02608     1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,
02609     2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,
02610     3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,
02611     1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,
02612     4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,
02613     5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,
02614     7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,
02615     7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,
02616     2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,
02617     5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,
02618     3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,
02619     5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,
02620     5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,
02621     5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,
02622     5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,
02623     1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,
02624     1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,
02625     3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,
02626     1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,
02627     3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,
02628     1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,
02629     6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,
02630     3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,
02631     2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,
02632     1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,
02633     3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,
02634     5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,
02635     1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,
02636     1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,
02637     4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,
02638     6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,
02639     1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,
02640     4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,
02641     5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,
02642     5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,
02643     5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,
02644     1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,
02645     2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,
02646     2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,
```

```
02647     3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,
02648     6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,
02649     9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,
02650     8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,
02651     1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,
02652     2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,
02653     4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,
02654     2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,
02655     1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,
02656     4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,
02657     5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,
02658     1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,
02659     9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,
02660     5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,
02661     4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,
02662     1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,
02663     1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,
02664     7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,
02665     2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,
02666     3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,
02667     3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,
02668     4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,
02669     1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,
02670     4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,
02671     1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,
02672     1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,
02673     3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,
02674     5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,
02675     6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,
02676     8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,
02677     4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,
02678     4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,
02679     5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,
02680     1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,
02681     5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,
02682     2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,
02683     1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,
02684     8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,
02685     2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,
02686     1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,
02687     1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,
02688     1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,
02689     6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,
02690     2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,
02691     4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,
02692     5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,
02693     5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,
02694     5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,
02695     1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,
02696     2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,
02697     6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,
02698     5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,
02699     9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,
02700     2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,
02701     5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,
02702     7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,
02703     7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,
02704     8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,
02705     1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,
02706     2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,
02707     9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,
02708     2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,
02709     4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,
02710     2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,
02711     2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,
02712     3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,
02713     1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,
02714     6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,
02715     1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,
02716     3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,
02717     3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,
02718     3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,
02719     3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,
02720     1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,
02721     1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,
02722     3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,
02723     9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,
02724     1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,
02725     8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,
02726     2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,
02727     3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,
02728     3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02729     3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02730     5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02731     4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02732     1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02733     6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
```

```
02734     9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02735     1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02736     1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02737     3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02738     1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02739     4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02740     6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02741     6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
02742     7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02743     2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02744     4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02745   };
02746
02747   static double xfcrev[15] =
02748     { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02749     1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02750   };
02751
02752   double a1, a2, a3, dw, ew, dx, xw, xx, vf2, vf6, cw260, cw296,
02753     sfac, fscal, cwfrn, ctmpth, ctwfrn, ctwslf;
02754
02755   int iw, ix;
02756
02757   /* Get H2O continuum absorption... */
02758   xw = nu / 10 + 1;
02759   if (xw >= 1 && xw < 2001) {
02760     iw = (int) xw;
02761     dw = xw - iw;
02762     ew = 1 - dw;
02763     cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02764     cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02765     cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02766     if (nu <= 820 || nu >= 960) {
02767       sfac = 1;
02768     } else {
02769       xx = (nu - 820) / 10;
02770       ix = (int) xx;
02771       dx = xx - ix;
02772       sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];
02773     }
02774     ctwslf = sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02775     vf2 = POW2(nu - 370);
02776     vf6 = POW3(vf2);
02777     fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02778     ctwfrn = cwfrn * fscal;
02779     a1 = nu * u * tanh(.7193876 / t * nu);
02780     a2 = 296 / t;
02781     a3 = p / P0 * (q * ctwslf + (1 - q) * ctwfrn) * 1e-20;
02782     ctmpth = a1 * a2 * a3;
02783   } else
02784     ctmpth = 0;
02785   return ctmpth;
02786 }
02787
02788 /*****************************************************************************/
02789
02790 double ctmn2(
02791   double nu,
02792   double p,
02793   double t) {
02794
02795   static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02796     1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02797     2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02798     5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02799     7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02800     9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02801     1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02802     1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02803     1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02804     1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02805     7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,
02806     3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02807     1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02808     7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02809   };
02810
02811   static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02812     511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02813     233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02814     -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02815     -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02816     -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02817     -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02818     121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02819     133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,
02820     372., 449., 514., 569., 609., 642., 673., 673.
```

```
02821   };
02822
02823   static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02824     2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
02825     2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02826     2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02827     2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02828     2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02829     2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
02830     2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02831     2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02832     2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02833     2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02834   };
02835
02836   double b, beta, q_n2 = 0.79, t0 = 273, tr = 296;
02837
02838   int idx;
02839
02840   /* Check wavenumber range... */
02841   if (nu < nua[0] || nu > nua[97])
02842     return 0;
02843
02844   /* Interpolate B and beta... */
02845   idx = locate_reg(nua, 98, nu);
02846   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02847   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02848
02849   /* Compute absorption coefficient... */
02850   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02851     * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02852 }
02853
02854 /*****************************************************************************/
02855
02856 double ctmo2(
02857   double nu,
02858   double p,
02859   double t) {
02860
02861   static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,
02862     .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,
02863     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02864     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02865     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02866     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02867     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02868     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02869     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02870     .071, .064, 0.
02871   };
02872
02873   static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02874     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02875     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02876     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,
02877     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02878     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02879     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02880     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02881     346., 322., 291., 290., 350., 371., 504., 504.
02882   };
02883
02884   static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02885     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02886     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02887     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02888     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
02889     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02890     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02891     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02892     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02893     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02894     1800., 1805.
02895   };
02896
02897   double b, beta, q_o2 = 0.21, t0 = 273, tr = 296;
02898
02899   int idx;
02900
02901   /* Check wavenumber range... */
02902   if (nu < nua[0] || nu > nua[89])
02903     return 0;
02904
02905   /* Interpolate B and beta... */
02906   idx = locate_reg(nua, 90, nu);
02907   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
```

```
02908   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02909
02910   /* Compute absorption coefficient... */
02911   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02912     b;
02913 }
02914
02915 /*****************************************************************************/
02916
02917 void copy_atm(
02918   ctl_t * ctl,
02919   atm_t * atm_dest,
02920   atm_t * atm_src,
02921   int init) {
02922
02923   int ig, ip, iw;
02924
02925   size_t s;
02926
02927   /* Data size... */
02928   s = (size_t) atm_src->np * sizeof(double);
02929
02930   /* Copy data... */
02931   atm_dest->np = atm_src->np;
02932   memcpy(atm_dest->time, atm_src->time, s);
02933   memcpy(atm_dest->z, atm_src->z, s);
02934   memcpy(atm_dest->lon, atm_src->lon, s);
02935   memcpy(atm_dest->lat, atm_src->lat, s);
02936   memcpy(atm_dest->p, atm_src->p, s);
02937   memcpy(atm_dest->t, atm_src->t, s);
02938   for (ig = 0; ig < ctl->ng; ig++)
02939     memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02940   for (iw = 0; iw < ctl->nw; iw++)
02941     memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02942
02943   /* Initialize... */
02944   if (init)
02945     for (ip = 0; ip < atm_dest->np; ip++) {
02946       atm_dest->p[ip] = 0;
02947       atm_dest->t[ip] = 0;
02948       for (ig = 0; ig < ctl->ng; ig++)
02949         atm_dest->q[ig][ip] = 0;
02950       for (iw = 0; iw < ctl->nw; iw++)
02951         atm_dest->k[iw][ip] = 0;
02952     }
02953 }
02954
02955 /*****************************************************************************/
02956
02957 void copy_obs(
02958   ctl_t * ctl,
02959   obs_t * obs_dest,
02960   obs_t * obs_src,
02961   int init) {
02962
02963   int id, ir;
02964
02965   size_t s;
02966
02967   /* Data size... */
02968   s = (size_t) obs_src->nr * sizeof(double);
02969
02970   /* Copy data... */
02971   obs_dest->nr = obs_src->nr;
02972   memcpy(obs_dest->time, obs_src->time, s);
02973   memcpy(obs_dest->obsz, obs_src->obsz, s);
02974   memcpy(obs_dest->obslon, obs_src->obslon, s);
02975   memcpy(obs_dest->obslat, obs_src->obslat, s);
02976   memcpy(obs_dest->vpz, obs_src->vpz, s);
02977   memcpy(obs_dest->vplon, obs_src->vplon, s);
02978   memcpy(obs_dest->vplat, obs_src->vplat, s);
02979   memcpy(obs_dest->tpz, obs_src->tpz, s);
02980   memcpy(obs_dest->tplon, obs_src->tplon, s);
02981   memcpy(obs_dest->tplat, obs_src->tplat, s);
02982   for (id = 0; id < ctl->nd; id++)
02983     memcpy(obs_dest->rad[id], obs_src->rad[id], s);
02984   for (id = 0; id < ctl->nd; id++)
02985     memcpy(obs_dest->tau[id], obs_src->tau[id], s);
02986
02987   /* Initialize... */
02988   if (init)
02989     for (id = 0; id < ctl->nd; id++)
02990       for (ir = 0; ir < obs_dest->nr; ir++)
02991         if (gsl_finite(obs_dest->rad[id][ir])) {
02992           obs_dest->rad[id][ir] = 0;
02993           obs_dest->tau[id][ir] = 0;
02994         }
```

```
02995 }
02996
02997 /*****************************************************************************/
02998
02999 int find_emitter(
03000   ctl_t * ctl,
03001   const char *emitter) {
03002
03003   int ig;
03004
03005   for (ig = 0; ig < ctl->ng; ig++)
03006     if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03007       return ig;
03008
03009   return -1;
03010 }
03011
03012 /*****************************************************************************/
03013
03014 void formod(
03015   ctl_t * ctl,
03016   atm_t * atm,
03017   obs_t * obs) {
03018
03019   int id, ir, *mask;
03020
03021   /* Allocate... */
03022   ALLOC(mask, int,
03023         ND * NR);
03024
03025   /* Save observation mask... */
03026   for (id = 0; id < ctl->nd; id++)
03027     for (ir = 0; ir < obs->nr; ir++)
03028       mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03029
03030   /* Hydrostatic equilibrium... */
03031   hydrostatic(ctl, atm);
03032
03033   /* Calculate pencil beams... */
03034   for (ir = 0; ir < obs->nr; ir++)
03035     formod_pencil(ctl, atm, obs, ir);
03036
03037   /* Apply field-of-view convolution... */
03038   formod_fov(ctl, obs);
03039
03040   /* Convert radiance to brightness temperature... */
03041   if (ctl->write_bbt)
03042     for (id = 0; id < ctl->nd; id++)
03043       for (ir = 0; ir < obs->nr; ir++)
03044         obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03045
03046   /* Apply observation mask... */
03047   for (id = 0; id < ctl->nd; id++)
03048     for (ir = 0; ir < obs->nr; ir++)
03049       if (mask[id * NR + ir])
03050         obs->rad[id][ir] = GSL_NAN;
03051
03052   /* Free... */
03053   free(mask);
03054 }
03055
03056 /*****************************************************************************/
03057
03058 void formod_continua(
03059   ctl_t * ctl,
03060   los_t * los,
03061   int ip,
03062   double *beta) {
03063
03064   static int ig_co2 = -999, ig_h2o = -999;
03065
03066   int id;
03067
03068   /* Extinction... */
03069   for (id = 0; id < ctl->nd; id++)
03070     beta[id] = los->k[ctl->window[id]][ip];
03071
03072   /* CO2 continuum... */
03073   if (ctl->ctm_co2) {
03074     if (ig_co2 == -999)
03075       ig_co2 = find_emitter(ctl, "CO2");
03076     if (ig_co2 >= 0)
03077       for (id = 0; id < ctl->nd; id++)
03078         beta[id] += ctmco2(ctl->nu[id], los->p[ip], los->t[ip],
03079                            los->u[ig_co2][ip]) / los->ds[ip];
03080   }
03081
```

```
03082   /* H2O continuum... */
03083   if (ctl->ctm_h2o) {
03084     if (ig_h2o == -999)
03085       ig_h2o = find_emitter(ctl, "H2O");
03086     if (ig_h2o >= 0)
03087       for (id = 0; id < ctl->nd; id++)
03088         beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03089                            los->q[ig_h2o][ip],
03090                            los->u[ig_h2o][ip]) / los->ds[ip];
03091   }
03092
03093   /* N2 continuum... */
03094   if (ctl->ctm_n2)
03095     for (id = 0; id < ctl->nd; id++)
03096       beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03097
03098   /* O2 continuum... */
03099   if (ctl->ctm_o2)
03100     for (id = 0; id < ctl->nd; id++)
03101       beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03102 }
03103
03104 /*****************************************************************************/
03105
03106 void formod_fov(
03107   ctl_t * ctl,
03108   obs_t * obs) {
03109
03110   static double dz[NSHAPE], w[NSHAPE];
03111
03112   static int init = 0, n;
03113
03114   obs_t *obs2;
03115
03116   double rad[ND][NR], tau[ND][NR], wsum, z[NR], zfov;
03117
03118   int i, id, idx, ir, ir2, nz;
03119
03120   /* Do not take into account FOV... */
03121   if (ctl->fov[0] == '-')
03122     return;
03123
03124   /* Initialize FOV data... */
03125   if (!init) {
03126     init = 1;
03127     read_shape(ctl->fov, dz, w, &n);
03128   }
03129
03130   /* Allocate... */
03131   ALLOC(obs2, obs_t, 1);
03132
03133   /* Copy observation data... */
03134   copy_obs(ctl, obs2, obs, 0);
03135
03136   /* Loop over ray paths... */
03137   for (ir = 0; ir < obs->nr; ir++) {
03138
03139     /* Get radiance and transmittance profiles... */
03140     nz = 0;
03141     for (ir2 = GSL_MAX(ir - NFOV, 0); ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr);
03142          ir2++)
03143       if (obs->time[ir2] == obs->time[ir]) {
03144         z[nz] = obs2->vpz[ir2];
03145         for (id = 0; id < ctl->nd; id++) {
03146           rad[id][nz] = obs2->rad[id][ir2];
03147           tau[id][nz] = obs2->tau[id][ir2];
03148         }
03149         nz++;
03150       }
03151     if (nz < 2)
03152       ERRMSG("Cannot apply FOV convolution!");
03153
03154     /* Convolute profiles with FOV... */
03155     wsum = 0;
03156     for (id = 0; id < ctl->nd; id++) {
03157       obs->rad[id][ir] = 0;
03158       obs->tau[id][ir] = 0;
03159     }
03160     for (i = 0; i < n; i++) {
03161       zfov = obs->vpz[ir] + dz[i];
03162       idx = locate_irr(z, nz, zfov);
03163       for (id = 0; id < ctl->nd; id++) {
03164         obs->rad[id][ir] += w[i]
03165           * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
03166         obs->tau[id][ir] += w[i]
03167           * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
03168       }
```

```
03169       wsum += w[i];
03170     }
03171     for (id = 0; id < ctl->nd; id++) {
03172       obs->rad[id][ir] /= wsum;
03173       obs->tau[id][ir] /= wsum;
03174     }
03175   }
03176
03177   /* Free... */
03178   free(obs2);
03179 }
03180
03181 /*****************************************************************************/
03182
03183 void formod_pencil(
03184   ctl_t * ctl,
03185   atm_t * atm,
03186   obs_t * obs,
03187   int ir) {
03188
03189   static tbl_t *tbl;
03190
03191   static int init = 0;
03192
03193   los_t *los;
03194
03195   double beta_ctm[ND], eps, src_planck[ND], tau_path[NG][ND], tau_gas[ND];
03196
03197   int id, ip;
03198
03199   /* Initialize look-up tables... */
03200   if (!init) {
03201     init = 1;
03202     ALLOC(tbl, tbl_t, 1);
03203     init_tbl(ctl, tbl);
03204   }
03205
03206   /* Allocate... */
03207   ALLOC(los, los_t, 1);
03208
03209   /* Initialize... */
03210   for (id = 0; id < ctl->nd; id++) {
03211     obs->rad[id][ir] = 0;
03212     obs->tau[id][ir] = 1;
03213   }
03214
03215   /* Raytracing... */
03216   raytrace(ctl, atm, obs, los, ir);
03217
03218   /* Loop over LOS points... */
03219   for (ip = 0; ip < los->np; ip++) {
03220
03221     /* Get trace gas transmittance... */
03222     intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03223
03224     /* Get continuum absorption... */
03225     formod_continua(ctl, los, ip, beta_ctm);
03226
03227     /* Compute Planck function... */
03228     formod_srcfunc(ctl, tbl, los->t[ip], src_planck);
03229
03230     /* Loop over channels... */
03231     for (id = 0; id < ctl->nd; id++)
03232       if (tau_gas[id] > 0) {
03233
03234         /* Get segment emissivity... */
03235         eps = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03236
03237         /* Compute radiance... */
03238         obs->rad[id][ir] += src_planck[id] * eps * obs->tau[id][ir];
03239
03240         /* Compute path transmittance... */
03241         obs->tau[id][ir] *= (1 - eps);
03242       }
03243   }
03244
03245   /* Add surface... */
03246   if (los->tsurf > 0) {
03247     formod_srcfunc(ctl, tbl, los->tsurf, src_planck);
03248     for (id = 0; id < ctl->nd; id++)
03249       obs->rad[id][ir] += src_planck[id] * obs->tau[id][ir];
03250   }
03251
03252   /* Free... */
03253   free(los);
03254 }
03255
```

```
03256 /*****************************************************************************/
03257
03258 void formod_srcfunc(
03259   ctl_t * ctl,
03260   tbl_t * tbl,
03261   double t,
03262   double *src) {
03263
03264   int id, it;
03265
03266   /* Determine index in temperature array... */
03267   it = locate_reg(tbl->st, TBLNS, t);
03268
03269   /* Interpolate Planck function value... */
03270   for (id = 0; id < ctl->nd; id++)
03271     src[id] = LIN(tbl->st[it], tbl->sr[id][it],
03272                   tbl->st[it + 1], tbl->sr[id][it + 1], t);
03273 }
03274
03275 /*****************************************************************************/
03276
03277 void geo2cart(
03278   double z,
03279   double lon,
03280   double lat,
03281   double *x) {
03282
03283   double radius;
03284
03285   radius = z + RE;
03286   x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);
03287   x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);
03288   x[2] = radius * sin(lat / 180 * M_PI);
03289 }
03290
03291 /*****************************************************************************/
03292
03293 void hydrostatic(
03294   ctl_t * ctl,
03295   atm_t * atm) {
03296
03297   static int ig_h2o = -999;
03298
03299   double dzmin = 1e99, e = 0, mean, mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03300
03301   int i, ip, ipref = 0, ipts = 20;
03302
03303   /* Check reference height... */
03304   if (ctl->hydz < 0)
03305     return;
03306
03307   /* Determine emitter index of H2O... */
03308   if (ig_h2o == -999)
03309     ig_h2o = find_emitter(ctl, "H2O");
03310
03311   /* Find air parcel next to reference height... */
03312   for (ip = 0; ip < atm->np; ip++)
03313     if (fabs(atm->z[ip] - ctl->hydz) < dzmin) {
03314       dzmin = fabs(atm->z[ip] - ctl->hydz);
03315       ipref = ip;
03316     }
03317
03318   /* Upper part of profile... */
03319   for (ip = ipref + 1; ip < atm->np; ip++) {
03320     mean = 0;
03321     for (i = 0; i < ipts; i++) {
03322       if (ig_h2o >= 0)
03323         e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03324                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03325       mean += (e * mmh2o + (1 - e) * mmair)
03326         * G0 / RI
03327         / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03328     }
03329
03330     /* Compute p(z,T)... */
03331     atm->p[ip] =
03332       exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03333   }
03334
03335   /* Lower part of profile... */
03336   for (ip = ipref - 1; ip >= 0; ip--) {
03337     mean = 0;
03338     for (i = 0; i < ipts; i++) {
03339       if (ig_h2o >= 0)
03340         e = LIN(0.0, atm->q[ig_h2o][ip + 1],
03341                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03342       mean += (e * mmh2o + (1 - e) * mmair)
```

```
03343         * G0 / RI
03344         / LIN(0.0, atm->t[ip + 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03345     }
03346
03347     /* Compute p(z,T)... */
03348     atm->p[ip] =
03349       exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03350   }
03351 }
03352
03353 /*****************************************************************************/
03354
03355 void idx2name(
03356   ctl_t * ctl,
03357   int idx,
03358   char *quantity) {
03359
03360   int ig, iw;
03361
03362   if (idx == IDXP)
03363     sprintf(quantity, "PRESSURE");
03364
03365   if (idx == IDXT)
03366     sprintf(quantity, "TEMPERATURE");
03367
03368   for (ig = 0; ig < ctl->ng; ig++)
03369     if (idx == IDXQ(ig))
03370       sprintf(quantity, "%s", ctl->emitter[ig]);
03371
03372   for (iw = 0; iw < ctl->nw; iw++)
03373     if (idx == IDXK(iw))
03374       sprintf(quantity, "EXTINCT_WINDOW%d", iw);
03375 }
03376
03377 /*****************************************************************************/
03378
03379 void init_tbl(
03380   ctl_t * ctl,
03381   tbl_t * tbl) {
03382
03383   FILE *in;
03384
03385   char filename[2 * LEN], line[LEN];
03386
03387   double eps, eps_old, press, press_old, temp, temp_old, u, u_old,
03388     f[NSHAPE], fsum, nu[NSHAPE];
03389
03390   int i, id, ig, ip, it, n;
03391
03392   /* Loop over trace gases and channels... */
03393   for (ig = 0; ig < ctl->ng; ig++)
03394 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(in,filename,line,eps,eps_old,press,
     press_old,temp,temp_old,u,u_old,id,ip,it)
03395     for (id = 0; id < ctl->nd; id++) {
03396
03397       /* Initialize... */
03398       tbl->np[ig][id] = -1;
03399       eps_old = -999;
03400       press_old = -999;
03401       temp_old = -999;
03402       u_old = -999;
03403
03404       /* Try to open file... */
03405       sprintf(filename, "%s_%.4f_%s.tab",
03406               ctl->tblbase, ctl->nu[id], ctl->emitter[ig]);
03407       if (!(in = fopen(filename, "r"))) {
03408         printf("Missing emissivity table: %s\n", filename);
03409         continue;
03410       }
03411       printf("Read emissivity table: %s\n", filename);
03412
03413       /* Read data... */
03414       while (fgets(line, LEN, in)) {
03415
03416         /* Parse line... */
03417         if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
03418           continue;
03419
03420         /* Determine pressure index... */
03421         if (press != press_old) {
03422           press_old = press;
03423           if ((++tbl->np[ig][id]) >= TBLNP)
03424             ERRMSG("Too many pressure levels!");
03425           tbl->nt[ig][id][tbl->np[ig][id]] = -1;
03426         }
03427
03428         /* Determine temperature index... */
```

```
03429         if (temp != temp_old) {
03430           temp_old = temp;
03431           if ((++tbl->nt[ig][id][tbl->np[ig][id]]) >= TBLNT)
03432             ERRMSG("Too many temperatures!");
03433           tbl->nu[ig][id][tbl->np[ig][id]]
03434             [tbl->nt[ig][id][tbl->np[ig][id]]] = -1;
03435         }
03436
03437         /* Determine column density index... */
03438         if ((eps > eps_old && u > u_old) || tbl->nu[ig][id][tbl->np[ig][id]]
03439              [tbl->nt[ig][id][tbl->np[ig][id]]] < 0) {
03440           eps_old = eps;
03441           u_old = u;
03442           if ((++tbl->nu[ig][id][tbl->np[ig][id]]
03443                [tbl->nt[ig][id][tbl->np[ig][id]]]) >= TBLNU) {
03444             tbl->nu[ig][id][tbl->np[ig][id]]
03445               [tbl->nt[ig][id][tbl->np[ig][id]]]--;
03446             continue;
03447           }
03448         }
03449
03450         /* Store data... */
03451         tbl->p[ig][id][tbl->np[ig][id]] = press;
03452         tbl->t[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03453           = temp;
03454         tbl->u[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03455           [tbl->nu[ig][id][tbl->np[ig][id]]
03456            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) u;
03457         tbl->eps[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03458           [tbl->nu[ig][id][tbl->np[ig][id]]
03459            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) eps;
03460       }
03461
03462       /* Increment counters... */
03463       tbl->np[ig][id]++;
03464       for (ip = 0; ip < tbl->np[ig][id]; ip++) {
03465         tbl->nt[ig][id][ip]++;
03466         for (it = 0; it < tbl->nt[ig][id][ip]; it++)
03467           tbl->nu[ig][id][ip][it]++;
03468       }
03469
03470       /* Close file... */
03471       fclose(in);
03472     }
03473
03474   /* Write info... */
03475   printf("Initialize source function table...\n");
03476
03477   /* Loop over channels... */
03478 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(filename,it,i,n,f,fsum,nu)
03479   for (id = 0; id < ctl->nd; id++) {
03480
03481     /* Read filter function... */
03482     sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03483     read_shape(filename, nu, f, &n);
03484
03485     /* Compute source function table... */
03486     for (it = 0; it < TBLNS; it++) {
03487
03488       /* Set temperature... */
03489       tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03490
03491       /* Integrate Planck function... */
03492       fsum = 0;
03493       tbl->sr[id][it] = 0;
03494       for (i = 0; i < n; i++) {
03495         fsum += f[i];
03496         tbl->sr[id][it] += f[i] * planck(tbl->st[it], nu[i]);
03497       }
03498       tbl->sr[id][it] /= fsum;
03499     }
03500   }
03501 }
03502
03503 /*****************************************************************************/
03504
03505 void intpol_atm(
03506   ctl_t * ctl,
03507   atm_t * atm,
03508   double z,
03509   double *p,
03510   double *t,
03511   double *q,
03512   double *k) {
03513
03514   int ig, ip, iw;
03515
```

```
03516   /* Get array index... */
03517   ip = locate_irr(atm->z, atm->np, z);
03518
03519   /* Interpolate... */
03520   *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);
03521   *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03522   for (ig = 0; ig < ctl->ng; ig++)
03523     q[ig] =
03524       LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03525   for (iw = 0; iw < ctl->nw; iw++)
03526     k[iw] =
03527       LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03528 }
03529
03530 /*****************************************************************************/
03531
03532 void intpol_tbl(
03533   ctl_t * ctl,
03534   tbl_t * tbl,
03535   los_t * los,
03536   int ip,
03537   double tau_path[NG][ND],
03538   double tau_seg[ND]) {
03539
03540   double eps, eps00, eps01, eps10, eps11, u;
03541
03542   int id, ig, ipr, it0, it1;
03543
03544   /* Initialize... */
03545   if (ip <= 0)
03546     for (ig = 0; ig < ctl->ng; ig++)
03547       for (id = 0; id < ctl->nd; id++)
03548         tau_path[ig][id] = 1;
03549
03550   /* Loop over channels... */
03551   for (id = 0; id < ctl->nd; id++) {
03552
03553     /* Initialize... */
03554     tau_seg[id] = 1;
03555
03556     /* Loop over emitters.... */
03557     for (ig = 0; ig < ctl->ng; ig++) {
03558
03559       /* Check size of table (pressure)... */
03560       if (tbl->np[ig][id] < 2)
03561         eps = 0;
03562
03563       /* Check transmittance... */
03564       else if (tau_path[ig][id] < 1e-9)
03565         eps = 1;
03566
03567       /* Interpolate... */
03568       else {
03569
03570         /* Determine pressure and temperature indices... */
03571         ipr = locate_irr(tbl->p[ig][id], tbl->np[ig][id], los->p[ip]);
03572         it0 =
03573           locate_irr(tbl->t[ig][id][ipr], tbl->nt[ig][id][ipr], los->
    t[ip]);
03574         it1 =
03575           locate_reg(tbl->t[ig][id][ipr + 1], tbl->nt[ig][id][ipr + 1],
03576                      los->t[ip]);
03577
03578         /* Check size of table (temperature and column density)... */
03579         if (tbl->nt[ig][id][ipr] < 2 || tbl->nt[ig][id][ipr + 1] < 2
03580             || tbl->nu[ig][id][ipr][it0] < 2
03581             || tbl->nu[ig][id][ipr][it0 + 1] < 2
03582             || tbl->nu[ig][id][ipr + 1][it1] < 2
03583             || tbl->nu[ig][id][ipr + 1][it1 + 1] < 2)
03584           eps = 0;
03585
03586         else {
03587
03588           /* Get emissivities of extended path... */
03589           u = intpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[ig][id]);
03590           eps00 = intpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ig][ip]);
03591
03592           u = intpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[ig][id]);
03593           eps01 =
03594             intpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ig][ip]);
03595
03596           u = intpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[ig][id]);
03597           eps10 =
03598             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ig][ip]);
03599
03600           u =
03601             intpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[ig][id]);
```

```
03602           eps11 =
03603             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->
      u[ig][ip]);
03604
03605           /* Interpolate with respect to temperature... */
03606           eps00 = LIN(tbl->t[ig][id][ipr][it0], eps00,
03607                       tbl->t[ig][id][ipr][it0 + 1], eps01, los->t[ip]);
03608           eps11 = LIN(tbl->t[ig][id][ipr + 1][it1], eps10,
03609                       tbl->t[ig][id][ipr + 1][it1 + 1], eps11, los->t[ip]);
03610
03611           /* Interpolate with respect to pressure... */
03612           eps00 = LIN(tbl->p[ig][id][ipr], eps00,
03613                       tbl->p[ig][id][ipr + 1], eps11, los->p[ip]);
03614
03615           /* Check emssivity range... */
03616           eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03617
03618           /* Determine segment emissivity... */
03619           eps = 1 - (1 - eps00) / tau_path[ig][id];
03620         }
03621       }
03622
03623       /* Get transmittance of extended path... */
03624       tau_path[ig][id] *= (1 - eps);
03625
03626       /* Get segment transmittance... */
03627       tau_seg[id] *= (1 - eps);
03628     }
03629   }
03630 }
03631
03632 /*****************************************************************************/
03633
03634 double intpol_tbl_eps(
03635   tbl_t * tbl,
03636   int ig,
03637   int id,
03638   int ip,
03639   int it,
03640   double u) {
03641
03642   int idx;
03643
03644   /* Lower boundary... */
03645   if (u < tbl->u[ig][id][ip][it][0])
03646     return LIN(0, 0, tbl->u[ig][id][ip][it][0], tbl->eps[ig][id][ip][it][0],
03647                u);
03648
03649   /* Upper boundary... */
03650   else if (u > tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
03651     return LIN(tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03652                tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03653                1e30, 1, u);
03654
03655   /* Interpolation... */
03656   else {
03657
03658     /* Get index... */
03659     idx = locate_tbl(tbl->u[ig][id][ip][it], tbl->nu[ig][id][ip][it], u);
03660
03661     /* Interpolate... */
03662     return
03663       LIN(tbl->u[ig][id][ip][it][idx], tbl->eps[ig][id][ip][it][idx],
03664           tbl->u[ig][id][ip][it][idx + 1], tbl->eps[ig][id][ip][it][idx + 1],
03665           u);
03666   }
03667 }
03668
03669 /*****************************************************************************/
03670
03671 double intpol_tbl_u(
03672   tbl_t * tbl,
03673   int ig,
03674   int id,
03675   int ip,
03676   int it,
03677   double eps) {
03678
03679   int idx;
03680
03681   /* Lower boundary... */
03682   if (eps < tbl->eps[ig][id][ip][it][0])
03683     return LIN(0, 0, tbl->eps[ig][id][ip][it][0], tbl->u[ig][id][ip][it][0],
03684                eps);
03685
03686   /* Upper boundary... */
03687   else if (eps > tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
```

```
03688     return LIN(tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03689                tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03690                1, 1e30, eps);
03691
03692   /* Interpolation... */
03693   else {
03694
03695     /* Get index... */
03696     idx = locate_tbl(tbl->eps[ig][id][ip][it], tbl->nu[ig][id][ip][it], eps);
03697
03698     /* Interpolate... */
03699     return
03700       LIN(tbl->eps[ig][id][ip][it][idx], tbl->u[ig][id][ip][it][idx],
03701           tbl->eps[ig][id][ip][it][idx + 1], tbl->u[ig][id][ip][it][idx + 1],
03702           eps);
03703   }
03704 }
03705
03706 /*****************************************************************************/
03707
03708 void jsec2time(
03709   double jsec,
03710   int *year,
03711   int *mon,
03712   int *day,
03713   int *hour,
03714   int *min,
03715   int *sec,
03716   double *remain) {
03717
03718   struct tm t0, *t1;
03719
03720   time_t jsec0;
03721
03722   t0.tm_year = 100;
03723   t0.tm_mon = 0;
03724   t0.tm_mday = 1;
03725   t0.tm_hour = 0;
03726   t0.tm_min = 0;
03727   t0.tm_sec = 0;
03728
03729   jsec0 = (time_t) jsec + timegm(&t0);
03730   t1 = gmtime(&jsec0);
03731
03732   *year = t1->tm_year + 1900;
03733   *mon = t1->tm_mon + 1;
03734   *day = t1->tm_mday;
03735   *hour = t1->tm_hour;
03736   *min = t1->tm_min;
03737   *sec = t1->tm_sec;
03738   *remain = jsec - floor(jsec);
03739 }
03740
03741 /*****************************************************************************/
03742
03743 void kernel(
03744   ctl_t * ctl,
03745   atm_t * atm,
03746   obs_t * obs,
03747   gsl_matrix * k) {
03748
03749   atm_t *atm1;
03750   obs_t *obs1;
03751
03752   gsl_vector *x0, *x1, *yy0, *yy1;
03753
03754   int *iqa, j;
03755
03756   double h;
03757
03758   size_t i, n, m;
03759
03760   /* Get sizes... */
03761   m = k->size1;
03762   n = k->size2;
03763
03764   /* Allocate... */
03765   x0 = gsl_vector_alloc(n);
03766   yy0 = gsl_vector_alloc(m);
03767   ALLOC(iqa, int,
03768         N);
03769
03770   /* Compute radiance for undisturbed atmospheric data... */
03771   formod(ctl, atm, obs);
03772
03773   /* Compose vectors... */
03774   atm2x(ctl, atm, x0, iqa, NULL);
```

```
03775   obs2y(ctl, obs, yy0, NULL, NULL);
03776
03777   /* Initialize kernel matrix... */
03778   gsl_matrix_set_zero(k);
03779
03780   /* Loop over state vector elements... */
03781 #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(i, j, h, x1, yy1, atm1,
       obs1)
03782   for (j = 0; j < (int) n; j++) {
03783
03784     /* Allocate... */
03785     x1 = gsl_vector_alloc(n);
03786     yy1 = gsl_vector_alloc(m);
03787     ALLOC(atm1, atm_t, 1);
03788     ALLOC(obs1, obs_t, 1);
03789
03790     /* Set perturbation size... */
03791     if (iqa[j] == IDXP)
03792       h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-7);
03793     else if (iqa[j] == IDXT)
03794       h = 1;
03795     else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03796       h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-15);
03797     else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03798       h = 1e-4;
03799     else
03800       ERRMSG("Cannot set perturbation size!");
03801
03802     /* Disturb state vector element... */
03803     gsl_vector_memcpy(x1, x0);
03804     gsl_vector_set(x1, (size_t) j, gsl_vector_get(x1, (size_t) j) + h);
03805     copy_atm(ctl, atm1, atm, 0);
03806     copy_obs(ctl, obs1, obs, 0);
03807     x2atm(ctl, x1, atm1);
03808
03809     /* Compute radiance for disturbed atmospheric data... */
03810     formod(ctl, atm1, obs1);
03811
03812     /* Compose measurement vector for disturbed radiance data... */
03813     obs2y(ctl, obs1, yy1, NULL, NULL);
03814
03815     /* Compute derivatives... */
03816     for (i = 0; i < m; i++)
03817       gsl_matrix_set(k, i, (size_t) j,
03818                      (gsl_vector_get(yy1, i) - gsl_vector_get(yy0, i)) / h);
03819
03820     /* Free... */
03821     gsl_vector_free(x1);
03822     gsl_vector_free(yy1);
03823     free(atm1);
03824     free(obs1);
03825   }
03826
03827   /* Free... */
03828   gsl_vector_free(x0);
03829   gsl_vector_free(yy0);
03830   free(iqa);
03831 }
03832
03833 /*****************************************************************************/
03834
03835 int locate_irr(
03836   double *xx,
03837   int n,
03838   double x) {
03839
03840   int i, ilo, ihi;
03841
03842   ilo = 0;
03843   ihi = n - 1;
03844   i = (ihi + ilo) >> 1;
03845
03846   if (xx[i] < xx[i + 1])
03847     while (ihi > ilo + 1) {
03848       i = (ihi + ilo) >> 1;
03849       if (xx[i] > x)
03850         ihi = i;
03851       else
03852         ilo = i;
03853   } else
03854     while (ihi > ilo + 1) {
03855       i = (ihi + ilo) >> 1;
03856       if (xx[i] <= x)
03857         ihi = i;
03858       else
03859         ilo = i;
03860     }
```

```
03861
03862   return ilo;
03863 }
03864
03865 /*****************************************************************************/
03866
03867 int locate_reg(
03868   double *xx,
03869   int n,
03870   double x) {
03871
03872   int i;
03873
03874   /* Calculate index... */
03875   i = (int) ((x - xx[0]) / (xx[1] - xx[0]));
03876
03877   /* Check range... */
03878   if (i < 0)
03879     i = 0;
03880   else if (i >= n - 2)
03881     i = n - 2;
03882
03883   return i;
03884 }
03885
03886 /*****************************************************************************/
03887
03888 int locate_tbl(
03889   float *xx,
03890   int n,
03891   double x) {
03892
03893   int i, ilo, ihi;
03894
03895   ilo = 0;
03896   ihi = n - 1;
03897   i = (ihi + ilo) >> 1;
03898
03899   while (ihi > ilo + 1) {
03900     i = (ihi + ilo) >> 1;
03901     if (xx[i] > x)
03902       ihi = i;
03903     else
03904       ilo = i;
03905   }
03906
03907   return ilo;
03908 }
03909
03910 /*****************************************************************************/
03911
03912 size_t obs2y(
03913   ctl_t * ctl,
03914   obs_t * obs,
03915   gsl_vector * y,
03916   int *ida,
03917   int *ira) {
03918
03919   int id, ir;
03920
03921   size_t m = 0;
03922
03923   /* Determine measurement vector... */
03924   for (ir = 0; ir < obs->nr; ir++)
03925     for (id = 0; id < ctl->nd; id++)
03926       if (gsl_finite(obs->rad[id][ir])) {
03927         if (y != NULL)
03928           gsl_vector_set(y, m, obs->rad[id][ir]);
03929         if (ida != NULL)
03930           ida[m] = id;
03931         if (ira != NULL)
03932           ira[m] = ir;
03933         m++;
03934       }
03935
03936   return m;
03937 }
03938
03939 /*****************************************************************************/
03940
03941 double planck(
03942   double t,
03943   double nu) {
03944
03945   return C1 * POW3(nu) / gsl_expm1(C2 * nu / t);
03946 }
03947
```

```
03948 /*****************************************************************************/
03949
03950 void raytrace(
03951   ctl_t * ctl,
03952   atm_t * atm,
03953   obs_t * obs,
03954   los_t * los,
03955   int ir) {
03956
03957   double cosa, d, dmax, dmin = 0, ds, ex0[3], ex1[3], frac, h = 0.02, k[NW],
03958     lat, lon, n, naux, ng[3], norm, p, q[NG], t, x[3], xh[3],
03959     xobs[3], xvp[3], z = 1e99, zmax, zmin, zrefrac = 60;
03960
03961   int i, ig, ip, iw, stop = 0;
03962
03963   /* Initialize... */
03964   los->np = 0;
03965   los->tsurf = -999;
03966   obs->tpz[ir] = obs->vpz[ir];
03967   obs->tplon[ir] = obs->vplon[ir];
03968   obs->tplat[ir] = obs->vplat[ir];
03969
03970   /* Get altitude range of atmospheric data... */
03971   gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03972
03973   /* Check observer altitude... */
03974   if (obs->obsz[ir] < zmin)
03975     ERRMSG("Observer below surface!");
03976
03977   /* Check view point altitude... */
03978   if (obs->vpz[ir] > zmax)
03979     return;
03980
03981   /* Determine Cartesian coordinates for observer and view point... */
03982   geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
03983   geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
03984
03985   /* Determine initial tangent vector... */
03986   for (i = 0; i < 3; i++)
03987     ex0[i] = xvp[i] - xobs[i];
03988   norm = NORM(ex0);
03989   for (i = 0; i < 3; i++)
03990     ex0[i] /= norm;
03991
03992   /* Observer within atmosphere... */
03993   for (i = 0; i < 3; i++)
03994     x[i] = xobs[i];
03995
03996   /* Observer above atmosphere (search entry point)... */
03997   if (obs->obsz[ir] > zmax) {
03998     dmax = norm;
03999     while (fabs(dmin - dmax) > 0.001) {
04000       d = (dmax + dmin) / 2;
04001       for (i = 0; i < 3; i++)
04002         x[i] = xobs[i] + d * ex0[i];
04003       cart2geo(x, &z, &lon, &lat);
04004       if (z <= zmax && z > zmax - 0.001)
04005         break;
04006       if (z < zmax - 0.0005)
04007         dmax = d;
04008       else
04009         dmin = d;
04010     }
04011   }
04012
04013   /* Ray-tracing... */
04014   while (1) {
04015
04016     /* Set step length... */
04017     ds = ctl->rayds;
04018     if (ctl->raydz > 0) {
04019       norm = NORM(x);
04020       for (i = 0; i < 3; i++)
04021         xh[i] = x[i] / norm;
04022       cosa = fabs(DOTP(ex0, xh));
04023       if (cosa != 0)
04024         ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04025     }
04026
04027     /* Determine geolocation... */
04028     cart2geo(x, &z, &lon, &lat);
04029
04030     /* Check if LOS hits the ground or has left atmosphere... */
04031     if (z < zmin || z > zmax) {
04032       stop = (z < zmin ? 2 : 1);
04033       frac =
04034         ((z <
```

```
04035           zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04036                                                                       1]);
04037       geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04038                los->lat[los->np - 1], xh);
04039       for (i = 0; i < 3; i++)
04040         x[i] = xh[i] + frac * (x[i] - xh[i]);
04041       cart2geo(x, &z, &lon, &lat);
04042       los->ds[los->np - 1] = ds * frac;
04043       ds = 0;
04044     }
04045
04046     /* Interpolate atmospheric data... */
04047     intpol_atm(ctl, atm, z, &p, &t, q, k);
04048
04049     /* Save data... */
04050     los->lon[los->np] = lon;
04051     los->lat[los->np] = lat;
04052     los->z[los->np] = z;
04053     los->p[los->np] = p;
04054     los->t[los->np] = t;
04055     for (ig = 0; ig < ctl->ng; ig++)
04056       los->q[ig][los->np] = q[ig];
04057     for (iw = 0; iw < ctl->nw; iw++)
04058       los->k[iw][los->np] = k[iw];
04059     los->ds[los->np] = ds;
04060
04061     /* Increment and check number of LOS points... */
04062     if ((++los->np) > NLOS)
04063       ERRMSG("Too many LOS points!");
04064
04065     /* Check stop flag... */
04066     if (stop) {
04067       los->tsurf = (stop == 2 ? t : -999);
04068       break;
04069     }
04070
04071     /* Determine refractivity... */
04072     if (ctl->refrac && z <= zrefrac)
04073       n = 1 + refractivity(p, t);
04074     else
04075       n = 1;
04076
04077     /* Construct new tangent vector (first term)... */
04078     for (i = 0; i < 3; i++)
04079       ex1[i] = ex0[i] * n;
04080
04081     /* Compute gradient of refractivity... */
04082     if (ctl->refrac && z <= zrefrac) {
04083       for (i = 0; i < 3; i++)
04084         xh[i] = x[i] + 0.5 * ds * ex0[i];
04085       cart2geo(xh, &z, &lon, &lat);
04086       intpol_atm(ctl, atm, z, &p, &t, q, k);
04087       n = refractivity(p, t);
04088       for (i = 0; i < 3; i++) {
04089         xh[i] += h;
04090         cart2geo(xh, &z, &lon, &lat);
04091         intpol_atm(ctl, atm, z, &p, &t, q, k);
04092         naux = refractivity(p, t);
04093         ng[i] = (naux - n) / h;
04094         xh[i] -= h;
04095       }
04096     } else
04097       for (i = 0; i < 3; i++)
04098         ng[i] = 0;
04099
04100     /* Construct new tangent vector (second term)... */
04101     for (i = 0; i < 3; i++)
04102       ex1[i] += ds * ng[i];
04103
04104     /* Normalize new tangent vector... */
04105     norm = NORM(ex1);
04106     for (i = 0; i < 3; i++)
04107       ex1[i] /= norm;
04108
04109     /* Determine next point of LOS... */
04110     for (i = 0; i < 3; i++)
04111       x[i] += 0.5 * ds * (ex0[i] + ex1[i]);
04112
04113     /* Copy tangent vector... */
04114     for (i = 0; i < 3; i++)
04115       ex0[i] = ex1[i];
04116   }
04117
04118   /* Get tangent point (to be done before changing segment lengths!)... */
04119   tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->
      tplat[ir]);
04120
```

```
04121   /* Change segment lengths according to trapezoid rule... */
04122   for (ip = los->np - 1; ip >= 1; ip--)
04123     los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04124   los->ds[0] *= 0.5;
04125
04126   /* Compute column density... */
04127   for (ip = 0; ip < los->np; ip++)
04128     for (ig = 0; ig < ctl->ng; ig++)
04129       los->u[ig][ip] = 10 * los->q[ig][ip] * los->p[ip]
04130         / (KB * los->t[ip]) * los->ds[ip];
04131 }
04132
04133 /*****************************************************************************/
04134
04135 void read_atm(
04136   const char *dirname,
04137   const char *filename,
04138   ctl_t * ctl,
04139   atm_t * atm) {
04140
04141   FILE *in;
04142
04143   char file[LEN], line[LEN], *tok;
04144
04145   int ig, iw;
04146
04147   /* Init... */
04148   atm->np = 0;
04149
04150   /* Set filename... */
04151   if (dirname != NULL)
04152     sprintf(file, "%s/%s", dirname, filename);
04153   else
04154     sprintf(file, "%s", filename);
04155
04156   /* Write info... */
04157   printf("Read atmospheric data: %s\n", file);
04158
04159   /* Open file... */
04160   if (!(in = fopen(file, "r")))
04161     ERRMSG("Cannot open file!");
04162
04163   /* Read line... */
04164   while (fgets(line, LEN, in)) {
04165
04166     /* Read data... */
04167     TOK(line, tok, "%lg", atm->time[atm->np]);
04168     TOK(NULL, tok, "%lg", atm->z[atm->np]);
04169     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04170     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04171     TOK(NULL, tok, "%lg", atm->p[atm->np]);
04172     TOK(NULL, tok, "%lg", atm->t[atm->np]);
04173     for (ig = 0; ig < ctl->ng; ig++)
04174       TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04175     for (iw = 0; iw < ctl->nw; iw++)
04176       TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04177
04178     /* Increment data point counter... */
04179     if ((++atm->np) > NP)
04180       ERRMSG("Too many data points!");
04181   }
04182
04183   /* Close file... */
04184   fclose(in);
04185
04186   /* Check number of points... */
04187   if (atm->np < 1)
04188     ERRMSG("Could not read any data!");
04189 }
04190
04191 /*****************************************************************************/
04192
04193 void read_ctl(
04194   int argc,
04195   char *argv[],
04196   ctl_t * ctl) {
04197
04198   int id, ig, iw;
04199
04200   /* Write info... */
04201   printf("\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04202          "(executable: %s | compiled: %s, %s)\n\n",
04203          argv[0], __DATE__, __TIME__);
04204
04205   /* Emitters... */
04206   ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04207   if (ctl->ng < 0 || ctl->ng > NG)
```

```
04208     ERRMSG("Set 0 <= NG <= MAX!");
04209   for (ig = 0; ig < ctl->ng; ig++)
04210     scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04211
04212   /* Radiance channels... */
04213   ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04214   if (ctl->nd < 0 || ctl->nd > ND)
04215     ERRMSG("Set 0 <= ND <= MAX!");
04216   for (id = 0; id < ctl->nd; id++)
04217     ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04218
04219   /* Spectral windows... */
04220   ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04221   if (ctl->nw < 0 || ctl->nw > NW)
04222     ERRMSG("Set 0 <= NW <= MAX!");
04223   for (id = 0; id < ctl->nd; id++)
04224     ctl->window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04225
04226   /* Emissivity look-up tables... */
04227   scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04228
04229   /* Hydrostatic equilibrium... */
04230   ctl->hydz = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04231
04232   /* Continua... */
04233   ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04234   ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04235   ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04236   ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04237
04238   /* Ray-tracing... */
04239   ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04240   ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04241   ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.5", NULL);
04242
04243   /* Field of view... */
04244   scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04245
04246   /* Retrieval interface... */
04247   ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04248   ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04249   ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04250   ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04251   for (ig = 0; ig < ctl->ng; ig++) {
04252     ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETQ_ZMIN", ig, "-999", NULL);
04253     ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETQ_ZMAX", ig, "-999", NULL);
04254   }
04255   for (iw = 0; iw < ctl->nw; iw++) {
04256     ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04257     ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04258   }
04259
04260   /* Output flags... */
04261   ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04262   ctl->write_matrix =
04263     (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04264 }
04265
04266 /*****************************************************************************/
04267
04268 void read_matrix(
04269   const char *dirname,
04270   const char *filename,
04271   gsl_matrix * matrix) {
04272
04273   FILE *in;
04274
04275   char dum[LEN], file[LEN], line[LEN];
04276
04277   double value;
04278
04279   int i, j;
04280
04281   /* Set filename... */
04282   if (dirname != NULL)
04283     sprintf(file, "%s/%s", dirname, filename);
04284   else
04285     sprintf(file, "%s", filename);
04286
04287   /* Write info... */
04288   printf("Read matrix: %s\n", file);
04289
04290   /* Open file... */
04291   if (!(in = fopen(file, "r")))
04292     ERRMSG("Cannot open file!");
04293
04294   /* Read data... */
```

```
04295   gsl_matrix_set_zero(matrix);
04296   while (fgets(line, LEN, in))
04297     if (sscanf(line, "%d %s %s %s %s %s %d %s %s %s %s %s %lg",
04298                &i, dum, dum, dum, dum, dum,
04299                &j, dum, dum, dum, dum, dum, &value) == 13)
04300       gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04301
04302   /* Close file... */
04303   fclose(in);
04304 }
04305
04306 /*****************************************************************************/
04307
04308 void read_obs(
04309   const char *dirname,
04310   const char *filename,
04311   ctl_t * ctl,
04312   obs_t * obs) {
04313
04314   FILE *in;
04315
04316   char file[LEN], line[LEN], *tok;
04317
04318   int id;
04319
04320   /* Init... */
04321   obs->nr = 0;
04322
04323   /* Set filename... */
04324   if (dirname != NULL)
04325     sprintf(file, "%s/%s", dirname, filename);
04326   else
04327     sprintf(file, "%s", filename);
04328
04329   /* Write info... */
04330   printf("Read observation data: %s\n", file);
04331
04332   /* Open file... */
04333   if (!(in = fopen(file, "r")))
04334     ERRMSG("Cannot open file!");
04335
04336   /* Read line... */
04337   while (fgets(line, LEN, in)) {
04338
04339     /* Read data... */
04340     TOK(line, tok, "%lg", obs->time[obs->nr]);
04341     TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04342     TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04343     TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04344     TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04345     TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04346     TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04347     TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04348     TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04349     TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04350     for (id = 0; id < ctl->nd; id++)
04351       TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04352     for (id = 0; id < ctl->nd; id++)
04353       TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04354
04355     /* Increment counter... */
04356     if ((++obs->nr) > NR)
04357       ERRMSG("Too many rays!");
04358   }
04359
04360   /* Close file... */
04361   fclose(in);
04362
04363   /* Check number of points... */
04364   if (obs->nr < 1)
04365     ERRMSG("Could not read any data!");
04366 }
04367
04368 /*****************************************************************************/
04369
04370 void read_shape(
04371   const char *filename,
04372   double *x,
04373   double *y,
04374   int *n) {
04375
04376   FILE *in;
04377
04378   char line[LEN];
04379
04380   /* Write info... */
04381   printf("Read shape function: %s\n", filename);
```

```
04382
04383   /* Open file... */
04384   if (!(in = fopen(filename, "r")))
04385     ERRMSG("Cannot open file!");
04386
04387   /* Read data... */
04388   *n = 0;
04389   while (fgets(line, LEN, in))
04390     if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04391       if ((++(*n)) > NSHAPE)
04392         ERRMSG("Too many data points!");
04393
04394   /* Check number of points... */
04395   if (*n < 1)
04396     ERRMSG("Could not read any data!");
04397
04398   /* Close file... */
04399   fclose(in);
04400 }
04401
04402 /*****************************************************************************/
04403
04404 double refractivity(
04405   double p,
04406   double t) {
04407
04408   /* Refractivity of air at 4 to 15 micron... */
04409   return 7.753e-05 * p / t;
04410 }
04411
04412 /*****************************************************************************/
04413
04414 double scan_ctl(
04415   int argc,
04416   char *argv[],
04417   const char *varname,
04418   int arridx,
04419   const char *defvalue,
04420   char *value) {
04421
04422   FILE *in = NULL;
04423
04424   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04425     msg[2 * LEN], rvarname[LEN], rval[LEN];
04426
04427   int contain = 0, i;
04428
04429   /* Open file... */
04430   if (argv[1][0] != '-')
04431     if (!(in = fopen(argv[1], "r")))
04432       ERRMSG("Cannot open file!");
04433
04434   /* Set full variable name... */
04435   if (arridx >= 0) {
04436     sprintf(fullname1, "%s[%d]", varname, arridx);
04437     sprintf(fullname2, "%s[*]", varname);
04438   } else {
04439     sprintf(fullname1, "%s", varname);
04440     sprintf(fullname2, "%s", varname);
04441   }
04442
04443   /* Read data... */
04444   if (in != NULL)
04445     while (fgets(line, LEN, in))
04446       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04447         if (strcasecmp(rvarname, fullname1) == 0 ||
04448             strcasecmp(rvarname, fullname2) == 0) {
04449           contain = 1;
04450           break;
04451         }
04452   for (i = 1; i < argc - 1; i++)
04453     if (strcasecmp(argv[i], fullname1) == 0 ||
04454         strcasecmp(argv[i], fullname2) == 0) {
04455       sprintf(rval, "%s", argv[i + 1]);
04456       contain = 1;
04457       break;
04458     }
04459
04460   /* Close file... */
04461   if (in != NULL)
04462     fclose(in);
04463
04464   /* Check for missing variables... */
04465   if (!contain) {
04466     if (strlen(defvalue) > 0)
04467       sprintf(rval, "%s", defvalue);
04468     else {
```

```
04469       sprintf(msg, "Missing variable %s!\n", fullname1);
04470       ERRMSG(msg);
04471     }
04472   }
04473
04474   /* Write info... */
04475   printf("%s = %s\n", fullname1, rval);
04476
04477   /* Return values... */
04478   if (value != NULL)
04479     sprintf(value, "%s", rval);
04480   return atof(rval);
04481 }
04482
04483 /*****************************************************************************/
04484
04485 void tangent_point(
04486   los_t * los,
04487   double *tpz,
04488   double *tplon,
04489   double *tplat) {
04490
04491   double a, b, c, dummy, v[3], v0[3], v2[3], x, x1, x2, yy0, yy1, yy2;
04492
04493   size_t i, ip;
04494
04495   /* Find minimum altitude... */
04496   ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
04497
04498   /* Nadir or zenith... */
04499   if (ip <= 0 || ip >= (size_t) los->np - 1) {
04500     *tpz = los->z[los->np - 1];
04501     *tplon = los->lon[los->np - 1];
04502     *tplat = los->lat[los->np - 1];
04503   }
04504
04505   /* Limb... */
04506   else {
04507
04508     /* Determine interpolating polynomial y=a*x^2+b*x+c... */
04509     yy0 = los->z[ip - 1];
04510     yy1 = los->z[ip];
04511     yy2 = los->z[ip + 1];
04512     x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
04513     x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
04514     a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);
04515     b = -(yy0 - yy1) / x1 - a * x1;
04516     c = yy0;
04517
04518     /* Get tangent point location... */
04519     x = -b / (2 * a);
04520     *tpz = a * x * x + b * x + c;
04521     geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
04522     geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);
04523     for (i = 0; i < 3; i++)
04524       v[i] = LIN(0.0, v0[i], x2, v2[i], x);
04525     cart2geo(v, &dummy, tplon, tplat);
04526   }
04527 }
04528
04529 /*****************************************************************************/
04530
04531 void time2jsec(
04532   int year,
04533   int mon,
04534   int day,
04535   int hour,
04536   int min,
04537   int sec,
04538   double remain,
04539   double *jsec) {
04540
04541   struct tm t0, t1;
04542
04543   t0.tm_year = 100;
04544   t0.tm_mon = 0;
04545   t0.tm_mday = 1;
04546   t0.tm_hour = 0;
04547   t0.tm_min = 0;
04548   t0.tm_sec = 0;
04549
04550   t1.tm_year = year - 1900;
04551   t1.tm_mon = mon - 1;
04552   t1.tm_mday = day;
04553   t1.tm_hour = hour;
04554   t1.tm_min = min;
04555   t1.tm_sec = sec;
```

```
04556
04557   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
04558 }
04559
04560 /*****************************************************************************/
04561
04562 void timer(
04563   const char *name,
04564   const char *file,
04565   const char *func,
04566   int line,
04567   int mode) {
04568
04569   static double w0[10];
04570
04571   static int l0[10], nt;
04572
04573   /* Start new timer... */
04574   if (mode == 1) {
04575     w0[nt] = omp_get_wtime();
04576     l0[nt] = line;
04577     if ((++nt) >= 10)
04578       ERRMSG("Too many timers!");
04579   }
04580
04581   /* Write elapsed time... */
04582   else {
04583
04584     /* Check timer index... */
04585     if (nt - 1 < 0)
04586       ERRMSG("Coding error!");
04587
04588     /* Write elapsed time... */
04589     printf("Timer '%s' (%s, %s, l%d-%d): %.3f sec\n",
04590            name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
04591   }
04592
04593   /* Stop timer... */
04594   if (mode == 3)
04595     nt--;
04596 }
04597
04598 /*****************************************************************************/
04599
04600 void write_atm(
04601   const char *dirname,
04602   const char *filename,
04603   ctl_t * ctl,
04604   atm_t * atm) {
04605
04606   FILE *out;
04607
04608   char file[LEN];
04609
04610   int ig, ip, iw, n = 6;
04611
04612   /* Set filename... */
04613   if (dirname != NULL)
04614     sprintf(file, "%s/%s", dirname, filename);
04615   else
04616     sprintf(file, "%s", filename);
04617
04618   /* Write info... */
04619   printf("Write atmospheric data: %s\n", file);
04620
04621   /* Create file... */
04622   if (!(out = fopen(file, "w")))
04623     ERRMSG("Cannot create file!");
04624
04625   /* Write header... */
04626   fprintf(out,
04627           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04628           "# $2 = altitude [km]\n"
04629           "# $3 = longitude [deg]\n"
04630           "# $4 = latitude [deg]\n"
04631           "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
04632   for (ig = 0; ig < ctl->ng; ig++)
04633     fprintf(out, "# $%d = %s volume mixing ratio\n", ++n, ctl->emitter[ig]);
04634   for (iw = 0; iw < ctl->nw; iw++)
04635     fprintf(out, "# $%d = window %d: extinction [1/km]\n", ++n, iw);
04636
04637   /* Write data... */
04638   for (ip = 0; ip < atm->np; ip++) {
04639     if (ip == 0 || atm->lat[ip] != atm->lat[ip - 1]
04640         || atm->lon[ip] != atm->lon[ip - 1])
04641       fprintf(out, "\n");
04642     fprintf(out, "%.2f %g %g %g %g %g", atm->time[ip], atm->z[ip],
```

```
04643             atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
04644     for (ig = 0; ig < ctl->ng; ig++)
04645       fprintf(out, " %g", atm->q[ig][ip]);
04646     for (iw = 0; iw < ctl->nw; iw++)
04647       fprintf(out, " %g", atm->k[iw][ip]);
04648     fprintf(out, "\n");
04649   }
04650
04651   /* Close file... */
04652   fclose(out);
04653 }
04654
04655 /*****************************************************************************/
04656
04657 void write_matrix(
04658   const char *dirname,
04659   const char *filename,
04660   ctl_t * ctl,
04661   gsl_matrix * matrix,
04662   atm_t * atm,
04663   obs_t * obs,
04664   const char *rowspace,
04665   const char *colspace,
04666   const char *sort) {
04667
04668   FILE *out;
04669
04670   char file[LEN], quantity[LEN];
04671
04672   int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
04673
04674   size_t i, j, nc, nr;
04675
04676   /* Check output flag... */
04677   if (!ctl->write_matrix)
04678     return;
04679
04680   /* Allocate... */
04681   ALLOC(cida, int, M);
04682   ALLOC(ciqa, int,
04683         N);
04684   ALLOC(cipa, int,
04685         N);
04686   ALLOC(cira, int,
04687         M);
04688   ALLOC(rida, int,
04689         M);
04690   ALLOC(riqa, int,
04691         N);
04692   ALLOC(ripa, int,
04693         N);
04694   ALLOC(rira, int,
04695         M);
04696
04697   /* Set filename... */
04698   if (dirname != NULL)
04699     sprintf(file, "%s/%s", dirname, filename);
04700   else
04701     sprintf(file, "%s", filename);
04702
04703   /* Write info... */
04704   printf("Write matrix: %s\n", file);
04705
04706   /* Create file... */
04707   if (!(out = fopen(file, "w")))
04708     ERRMSG("Cannot create file!");
04709
04710   /* Write header (row space)... */
04711   if (rowspace[0] == 'y') {
04712
04713     fprintf(out,
04714             "# $1 = Row: index (measurement space)\n"
04715             "# $2 = Row: channel wavenumber [cm^-1]\n"
04716             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
04717             "# $4 = Row: view point altitude [km]\n"
04718             "# $5 = Row: view point longitude [deg]\n"
04719             "# $6 = Row: view point latitude [deg]\n");
04720
04721     /* Get number of rows... */
04722     nr = obs2y(ctl, obs, NULL, rida, rira);
04723
04724   } else {
04725
04726     fprintf(out,
04727             "# $1 = Row: index (state space)\n"
04728             "# $2 = Row: name of quantity\n"
04729             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
```

```
04730             "# $4 = Row: altitude [km]\n"
04731             "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
04732
04733     /* Get number of rows... */
04734     nr = atm2x(ctl, atm, NULL, riqa, ripa);
04735   }
04736
04737   /* Write header (column space)... */
04738   if (colspace[0] == 'y') {
04739
04740     fprintf(out,
04741             "# $7 = Col: index (measurement space)\n"
04742             "# $8 = Col: channel wavenumber [cm^-1]\n"
04743             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04744             "# $10 = Col: view point altitude [km]\n"
04745             "# $11 = Col: view point longitude [deg]\n"
04746             "# $12 = Col: view point latitude [deg]\n");
04747
04748     /* Get number of columns... */
04749     nc = obs2y(ctl, obs, NULL, cida, cira);
04750
04751   } else {
04752
04753     fprintf(out,
04754             "# $7 = Col: index (state space)\n"
04755             "# $8 = Col: name of quantity\n"
04756             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04757             "# $10 = Col: altitude [km]\n"
04758             "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
04759
04760     /* Get number of columns... */
04761     nc = atm2x(ctl, atm, NULL, ciqa, cipa);
04762   }
04763
04764   /* Write header entry... */
04765   fprintf(out, "# $13 = Matrix element\n\n");
04766
04767   /* Write matrix data... */
04768   i = j = 0;
04769   while (i < nr && j < nc) {
04770
04771     /* Write info about the row... */
04772     if (rowspace[0] == 'y')
04773       fprintf(out, "%d %g %.2f %g %g %g",
04774               (int) i, ctl->nu[rida[i]],
04775               obs->time[rira[i]], obs->vpz[rira[i]],
04776               obs->vplon[rira[i]], obs->vplat[rira[i]]);
04777     else {
04778       idx2name(ctl, riqa[i], quantity);
04779       fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
04780               atm->time[ripa[i]], atm->z[ripa[i]],
04781               atm->lon[ripa[i]], atm->lat[ripa[i]]);
04782     }
04783
04784     /* Write info about the column... */
04785     if (colspace[0] == 'y')
04786       fprintf(out, " %d %g %.2f %g %g %g",
04787               (int) j, ctl->nu[cida[j]],
04788               obs->time[cira[j]], obs->vpz[cira[j]],
04789               obs->vplon[cira[j]], obs->vplat[cira[j]]);
04790     else {
04791       idx2name(ctl, ciqa[j], quantity);
04792       fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
04793               atm->time[cipa[j]], atm->z[cipa[j]],
04794               atm->lon[cipa[j]], atm->lat[cipa[j]]);
04795     }
04796
04797     /* Write matrix entry... */
04798     fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
04799
04800     /* Set matrix indices... */
04801     if (sort[0] == 'r') {
04802       j++;
04803       if (j >= nc) {
04804         j = 0;
04805         i++;
04806         fprintf(out, "\n");
04807       }
04808     } else {
04809       i++;
04810       if (i >= nr) {
04811         i = 0;
04812         j++;
04813         fprintf(out, "\n");
04814       }
04815     }
04816   }
```

```
04817
04818   /* Close file... */
04819   fclose(out);
04820
04821   /* Free... */
04822   free(cida);
04823   free(ciqa);
04824   free(cipa);
04825   free(cira);
04826   free(rida);
04827   free(riqa);
04828   free(ripa);
04829   free(rira);
04830 }
04831
04832 /*****************************************************************************/
04833
04834 void write_obs(
04835   const char *dirname,
04836   const char *filename,
04837   ctl_t * ctl,
04838   obs_t * obs) {
04839
04840   FILE *out;
04841
04842   char file[LEN];
04843
04844   int id, ir, n = 10;
04845
04846   /* Set filename... */
04847   if (dirname != NULL)
04848     sprintf(file, "%s/%s", dirname, filename);
04849   else
04850     sprintf(file, "%s", filename);
04851
04852   /* Write info... */
04853   printf("Write observation data: %s\n", file);
04854
04855   /* Create file... */
04856   if (!(out = fopen(file, "w")))
04857     ERRMSG("Cannot create file!");
04858
04859   /* Write header... */
04860   fprintf(out,
04861           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04862           "# $2 = observer altitude [km]\n"
04863           "# $3 = observer longitude [deg]\n"
04864           "# $4 = observer latitude [deg]\n"
04865           "# $5 = view point altitude [km]\n"
04866           "# $6 = view point longitude [deg]\n"
04867           "# $7 = view point latitude [deg]\n"
04868           "# $8 = tangent point altitude [km]\n"
04869           "# $9 = tangent point longitude [deg]\n"
04870           "# $10 = tangent point latitude [deg]\n");
04871   for (id = 0; id < ctl->nd; id++)
04872     fprintf(out, "# $%d = channel %g: radiance [W/(m^2 sr cm^-1)]\n",
04873             ++n, ctl->nu[id]);
04874   for (id = 0; id < ctl->nd; id++)
04875     fprintf(out, "# $%d = channel %g: transmittance\n", ++n, ctl->nu[id]);
04876
04877   /* Write data... */
04878   for (ir = 0; ir < obs->nr; ir++) {
04879     if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
04880       fprintf(out, "\n");
04881     fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
04882             obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
04883             obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
04884             obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
04885     for (id = 0; id < ctl->nd; id++)
04886       fprintf(out, " %g", obs->rad[id][ir]);
04887     for (id = 0; id < ctl->nd; id++)
04888       fprintf(out, " %g", obs->tau[id][ir]);
04889     fprintf(out, "\n");
04890   }
04891
04892   /* Close file... */
04893   fclose(out);
04894 }
04895
04896 /*****************************************************************************/
04897
04898 void x2atm(
04899   ctl_t * ctl,
04900   gsl_vector * x,
04901   atm_t * atm) {
04902
04903   int ig, iw;
```

```
04904
04905   size_t n = 0;
04906
04907   /* Set pressure... */
04908   x2atm_help(atm, ctl->retp_zmin, ctl->retp_zmax, atm->
      p, x, &n);
04909
04910   /* Set temperature... */
04911   x2atm_help(atm, ctl->rett_zmin, ctl->rett_zmax, atm->
      t, x, &n);
04912
04913   /* Set volume mixing ratio... */
04914   for (ig = 0; ig < ctl->ng; ig++)
04915     x2atm_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
04916                atm->q[ig], x, &n);
04917
04918   /* Set extinction... */
04919   for (iw = 0; iw < ctl->nw; iw++)
04920     x2atm_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
04921                atm->k[iw], x, &n);
04922 }
04923
04924 /*****************************************************************************/
04925
04926 void x2atm_help(
04927   atm_t * atm,
04928   double zmin,
04929   double zmax,
04930   double *value,
04931   gsl_vector * x,
04932   size_t * n) {
04933
04934   int ip;
04935
04936   /* Extract state vector elements... */
04937   for (ip = 0; ip < atm->np; ip++)
04938     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
04939       value[ip] = gsl_vector_get(x, *n);
04940       (*n)++;
04941     }
04942 }
04943
04944 /*****************************************************************************/
04945
04946 void y2obs(
04947   ctl_t * ctl,
04948   gsl_vector * y,
04949   obs_t * obs) {
04950
04951   int id, ir;
04952
04953   size_t m = 0;
04954
04955   /* Decompose measurement vector... */
04956   for (ir = 0; ir < obs->nr; ir++)
04957     for (id = 0; id < ctl->nd; id++)
04958       if (gsl_finite(obs->rad[id][ir])) {
04959         obs->rad[id][ir] = gsl_vector_get(y, m);
04960         m++;
04961       }
04962 }
```

5.7 jurassic.h File Reference

JURASSIC library declarations.

Data Structures

- struct atm_t

  Atmospheric data.
- struct ctl_t

  Forward model control parameters.
- struct los_t

  Line-of-sight data.

- struct obs_t

  Observation geometry and radiance data.
- struct tbl_t

  Emissivity look-up tables.

Functions

- size_t atm2x (ctl_t *ctl, atm_t *atm, gsl_vector *x, int *iqa, int *ipa)

  Compose state vector or parameter vector.
- void atm2x_help (atm_t *atm, double zmin, double zmax, double *value, int val_iqa, gsl_vector *x, int *iqa, int *ipa, size_t *n)

  Add elements to state vector.
- double brightness (double rad, double nu)

  Compute brightness temperature.
- void cart2geo (double *x, double *z, double *lon, double *lat)

  Convert Cartesian coordinates to geolocation.
- void climatology (ctl_t *ctl, atm_t *atm_mean)

  Interpolate climatological data.
- double ctmco2 (double nu, double p, double t, double u)

  Compute carbon dioxide continuum (optical depth).
- double ctmh2o (double nu, double p, double t, double q, double u)

  Compute water vapor continuum (optical depth).
- double ctmn2 (double nu, double p, double t)

  Compute nitrogen continuum (absorption coefficient).
- double ctmo2 (double nu, double p, double t)

  Compute oxygen continuum (absorption coefficient).
- void copy_atm (ctl_t *ctl, atm_t *atm_dest, atm_t *atm_src, int init)

  Copy and initialize atmospheric data.
- void copy_obs (ctl_t *ctl, obs_t *obs_dest, obs_t *obs_src, int init)

  Copy and initialize observation data.
- int find_emitter (ctl_t *ctl, const char *emitter)

  Find index of an emitter.
- void formod (ctl_t *ctl, atm_t *atm, obs_t *obs)

  Determine ray paths and compute radiative transfer.
- void formod_continua (ctl_t *ctl, los_t *los, int ip, double *beta)

  Compute absorption coefficient of continua.
- void formod_fov (ctl_t *ctl, obs_t *obs)

  Apply field of view convolution.
- void formod_pencil (ctl_t *ctl, atm_t *atm, obs_t *obs, int ir)

  Compute radiative transfer for a pencil beam.
- void formod_srcfunc (ctl_t *ctl, tbl_t *tbl, double t, double *src)

  Compute Planck source function.
- void geo2cart (double z, double lon, double lat, double *x)

  Convert geolocation to Cartesian coordinates.
- void hydrostatic (ctl_t *ctl, atm_t *atm)

  Set hydrostatic equilibrium.
- void idx2name (ctl_t *ctl, int idx, char *quantity)

  Determine name of state vector quantity for given index.
- void init_tbl (ctl_t *ctl, tbl_t *tbl)

  Initialize look-up tables.

- void intpol_atm (ctl_t *ctl, atm_t *atm, double z, double *p, double *t, double *q, double *k)

  Interpolate atmospheric data.
- void intpol_tbl (ctl_t *ctl, tbl_t *tbl, los_t *los, int ip, double tau_path[NG][ND], double tau_seg[ND])

  Get transmittance from look-up tables.
- double intpol_tbl_eps (tbl_t *tbl, int ig, int id, int ip, int it, double u)

  Interpolate emissivity from look-up tables.
- double intpol_tbl_u (tbl_t *tbl, int ig, int id, int ip, int it, double eps)

  Interpolate column density from look-up tables.
- void jsec2time (double jsec, int *year, int *mon, int *day, int *hour, int *min, int *sec, double *remain)

  Convert seconds to date.
- void kernel (ctl_t *ctl, atm_t *atm, obs_t *obs, gsl_matrix *k)

  Compute Jacobians.
- int locate_irr (double *xx, int n, double x)

  Find array index for irregular grid.
- int locate_reg (double *xx, int n, double x)

  Find array index for regular grid.
- int locate_tbl (float *xx, int n, double x)

  Find array index in float array.
- size_t obs2y (ctl_t *ctl, obs_t *obs, gsl_vector *y, int *ida, int *ira)

  Compose measurement vector.
- double planck (double t, double nu)

  Compute Planck function.
- void raytrace (ctl_t *ctl, atm_t *atm, obs_t *obs, los_t *los, int ir)

  Do ray-tracing to determine LOS.
- void read_atm (const char *dirname, const char *filename, ctl_t *ctl, atm_t *atm)

  Read atmospheric data.
- void read_ctl (int argc, char *argv[ ], ctl_t *ctl)

  Read forward model control parameters.
- void read_matrix (const char *dirname, const char *filename, gsl_matrix *matrix)

  Read matrix.
- void read_obs (const char *dirname, const char *filename, ctl_t *ctl, obs_t *obs)

  Read observation data.
- void read_shape (const char *filename, double *x, double *y, int *n)

  Read shape function.
- double refractivity (double p, double t)

  Compute refractivity (return value is n - 1).
- double scan_ctl (int argc, char *argv[ ], const char *varname, int arridx, const char *defvalue, char *value)

  Search control parameter file for variable entry.
- void tangent_point (los_t *los, double *tpz, double *tplon, double *tplat)

  Find tangent point of a given LOS.
- void time2jsec (int year, int mon, int day, int hour, int min, int sec, double remain, double *jsec)

  Convert date to seconds.
- void timer (const char *name, const char *file, const char *func, int line, int mode)

  Measure wall-clock time.
- void write_atm (const char *dirname, const char *filename, ctl_t *ctl, atm_t *atm)

  Write atmospheric data.
- void write_matrix (const char *dirname, const char *filename, ctl_t *ctl, gsl_matrix *matrix, atm_t *atm, obs_t *obs, const char *rowspace, const char *colspace, const char *sort)

  Write matrix.
- void write_obs (const char *dirname, const char *filename, ctl_t *ctl, obs_t *obs)

  Write observation data.

- void x2atm (ctl_t *ctl, gsl_vector *x, atm_t *atm)

  Decompose parameter vector or state vector.
- void x2atm_help (atm_t *atm, double zmin, double zmax, double *value, gsl_vector *x, size_t *n)

  Extract elements from state vector.
- void y2obs (ctl_t *ctl, gsl_vector *y, obs_t *obs)

  Decompose measurement vector.

5.7.1 Detailed Description

JURASSIC library declarations.

Definition in file jurassic.h.

5.7.2 Function Documentation

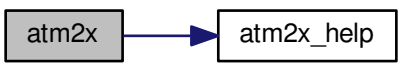
5.7.2.1 size_t atm2x (ctl_t * *ctl*, atm_t * *atm*, gsl_vector * *x*, int * *iqa*, int * *ipa*)

Compose state vector or parameter vector.

Definition at line 29 of file jurassic.c.

```
00034            {
00035
00036   int ig, iw;
00037
00038   size_t n = 0;
00039
00040   /* Add pressure... */
00041   atm2x_help(atm, ctl->retp_zmin, ctl->retp_zmax,
00042              atm->p, IDXP, x, iqa, ipa, &n);
00043
00044   /* Add temperature... */
00045   atm2x_help(atm, ctl->rett_zmin, ctl->rett_zmax,
00046              atm->t, IDXT, x, iqa, ipa, &n);
00047
00048   /* Add volume mixing ratios... */
00049   for (ig = 0; ig < ctl->ng; ig++)
00050     atm2x_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
00051                atm->q[ig], IDXQ(ig), x, iqa, ipa, &n);
00052
00053   /* Add extinction... */
00054   for (iw = 0; iw < ctl->nw; iw++)
00055     atm2x_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
00056                atm->k[iw], IDXK(iw), x, iqa, ipa, &n);
00057
00058   return n;
00059 }
```

Here is the call graph for this function:



5.7.2.2 void atm2x_help (atm_t * *atm*, double *zmin*, double *zmax*, double * *value*, int *val_iqa*, gsl_vector * *x*, int * *iqa*, int * *ipa*, size_t * *n*)

Add elements to state vector.

Definition at line 63 of file jurassic.c.

```
00072               {
00073
00074   int ip;
00075
00076   /* Add elements to state vector... */
00077   for (ip = 0; ip < atm->np; ip++)
00078     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
00079       if (x != NULL)
00080         gsl_vector_set(x, *n, value[ip]);
00081       if (iqa != NULL)
00082         iqa[*n] = val_iqa;
00083       if (ipa != NULL)
00084         ipa[*n] = ip;
00085       (*n)++;
00086     }
00087 }
```

5.7.2.3 double brightness (double *rad*, double *nu*)

Compute brightness temperature.

Definition at line 91 of file jurassic.c.

```
00093               {
00094
00095   return C2 * nu / gsl_log1p(C1 * POW3(nu) / rad);
00096 }
```

5.7.2.4 void cart2geo (double * *x*, double * *z*, double * *lon*, double * *lat*)

Convert Cartesian coordinates to geolocation.

Definition at line 101 of file jurassic.c.

```
00105                {
00106
00107   double radius;
00108
00109   radius = NORM(x);
00110   *lat = asin(x[2] / radius) * 180 / M_PI;
00111   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00112   *z = radius - RE;
00113 }
```

5.7.2.5 void climatology (ctl_t * *ctl*, atm_t * *atm_mean*)

Interpolate climatological data.

Definition at line 117 of file jurassic.c.

```
00119               {
00120
00121   static double z[121] = {
00122     0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00123     20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
00124     38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00125     56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00126     74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00127     92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00128     108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00129   };
00130
00131   static double pre[121] = {
00132     1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00133     357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00134     104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00135     29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00136     10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00137     3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00138     1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00139     0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00140     0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00141     0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00142     0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00143     0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00144     0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00145     0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00146     0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00147     0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00148     9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00149     4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00150     2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00151   };
00152
00153   static double tem[121] = {
00154     285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00155     229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00156     215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00157     222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00158     241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00159     262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00160     258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,
00161     237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00162     220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00163     207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00164     190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00165     178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00166     201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00167     272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00168   };
00169
00170   static double c2h2[121] = {
00171     1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00172     2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
00173     5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00174     2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00175     9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00176     1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00177     1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00178     1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00179     2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00180     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00181     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00182     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00183   };
00184
00185   static double c2h6[121] = {
00186     2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00187     1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00188     5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00189     2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00190     2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00191     1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00192     5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00193     2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00194     1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00195     7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,
```

```
00196     3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00197     1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00198     4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00199     1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00200     3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00201     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00202     0, 0, 0, 0, 0, 0, 0, 0, 0
00203   };
00204
00205   static double ccl4[121] = {
00206     1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00207     1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00208     8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00209     3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00210     3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
00211     4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00212     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00213     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00214     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00215     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00216     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00217     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00218     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00219     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00220     1e-14, 1e-14, 1e-14
00221   };
00222
00223   static double ch4[121] = {
00224     1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00225     1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00226     1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00227     1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00228     1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00229     8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00230     6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00231     4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00232     3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00233     2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00234     1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00235     1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00236     1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00237     9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00238     7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00239     5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00240     4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00241     3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00242     2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00243     2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00244     1.782e-08
00245   };
00246
00247   static double clo[121] = {
00248     7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00249     6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00250     8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00251     2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00252     1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00253     2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00254     4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00255     5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,
00256     3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00257     1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00258     6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00259     2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
00260     8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00261     3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00262     1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00263     3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00264     1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00265     3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00266     1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00267     5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00268     3.148e-15
00269   };
00270
00271   static double clono2[121] = {
00272     1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00273     1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00274     2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00275     2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00276     8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00277     6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00278     1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00279     1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00280     1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00281     1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00282     9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
```

```
00283     6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00284     3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00285     1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00286     8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00287     3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00288     9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00289     3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00290     2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00291     2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00292     4.041e-27
00293   };
00294
00295   static double co[121] = {
00296     1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00297     9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
00298     5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00299     2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00300     1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00301     2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00302     3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00303     3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00304     6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00305     2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00306     8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00307     2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00308     3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00309     6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00310     1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00311     1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00312     3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00313     5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00314     6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00315     7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05
00316   };
00317
00318   static double cof2[121] = {
00319     7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00320     6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00321     7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00322     4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00323     1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00324     1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00325     1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00326     8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00327     5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00328     2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00329     7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00330     1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00331     4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00332     1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00333     2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00334     4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
00335     7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00336     1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00337     3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00338     1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00339     4.662e-18
00340   };
00341
00342   static double f11[121] = {
00343     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00344     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00345     2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00346     1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
00347     7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00348     5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00349     1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00350     3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00351     6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00352     1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00353     1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00354     2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00355     2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00356     2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00357     2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00358     1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00359     1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00360     1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00361     2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00362     4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00363   };
00364
00365   static double f12[121] = {
00366     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00367     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00368     5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00369     4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
```

```
00370     2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00371     5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00372     2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00373     8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00374     3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00375     1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00376     8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00377     4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00378     2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00379     9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00380     4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00381     1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00382     7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00383     3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00384     1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
00385     1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00386   };
00387
00388   static double f14[121] = {
00389     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00390     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00391     8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00392     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00393     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00394     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00395     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00396     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00397     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00398     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00399     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00400     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00401     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00402     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00403     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00404     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00405     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00406   };
00407
00408   static double f22[121] = {
00409     1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00410     1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00411     1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00412     7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00413     4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00414     3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00415     1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00416     1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00417     8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00418     5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00419     4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00420     3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00421     3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
00422     2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00423     2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00424     2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00425     1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00426     1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00427     1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00428     1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00429   };
00430
00431   static double h2o[121] = {
00432     0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00433     0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
00434     6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00435     4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00436     4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00437     5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00438     5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00439     6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00440     6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00441     6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00442     5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00443     4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00444     3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00445     2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00446     1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00447     5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00448     1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00449     7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00450     3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00451     1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00452   };
00453
00454   static double h2o2[121] = {
00455     1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00456     4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
```

```
00457     3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00458     1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00459     8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00460     1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00461     1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00462     6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00463     5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00464     4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00465     3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00466     2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00467     1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00468     1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00469     9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00470     7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00471     4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
00472     3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00473     2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00474     2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00475     1.775e-12
00476   };
00477
00478   static double hcn[121] = {
00479     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00480     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00481     5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00482     1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00483     1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00484     1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00485     1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00486     1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00487     1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00488     9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00489     8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00490     7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00491     6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00492     6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00493     6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00494     6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00495     5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00496     5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00497     5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00498     5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00499   };
00500
00501   static double hno3[121] = {
00502     1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00503     2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00504     5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00505     3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00506     8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00507     3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00508     8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
00509     1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00510     6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00511     3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00512     1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00513     9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00514     5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00515     2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00516     1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
00517     5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00518     2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00519     1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00520     5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
00521     3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00522     2.332e-14
00523   };
00524
00525   static double hno4[121] = {
00526     6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00527     1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00528     3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00529     1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00530     2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00531     1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00532     3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00533     2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00534     1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00535     2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00536     1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00537     5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00538     2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00539     8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00540     3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00541     1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00542     3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00543     1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
```

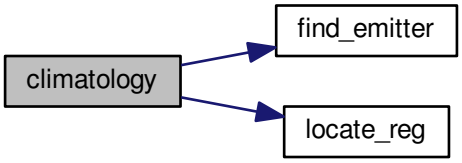
```
00544     5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00545     2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00546     1.64e-18
00547   };
00548
00549   static double hocl[121] = {
00550     1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00551     2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00552     5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00553     1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00554     4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00555     7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00556     1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00557     6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00558     1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
00559     3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00560     6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00561     1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00562     1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00563     2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00564     3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00565     4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00566     5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00567     6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00568     9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00569     2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00570     7.881e-21
00571   };
00572
00573   static double n2o[121] = {
00574     3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00575     3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00576     2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00577     2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00578     1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00579     7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00580     2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00581     1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00582     5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00583     2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00584     1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00585     1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00586     9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00587     7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00588     5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00589     4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00590     3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00591     2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00592     2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00593     2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00594   };
00595
00596   static double n2o5[121] = {
00597     1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00598     1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00599     4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00600     7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00601     3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00602     2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00603     2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
00604     6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00605     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00606     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00607     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00608     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00609     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00610     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00611     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00612     1e-16, 1e-16
00613   };
00614
00615   static double nh3[121] = {
00616     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00617     1e-10, 1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00618     4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00619     5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00620     6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00621     1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00622     1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00623     1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00624     2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00625     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00626     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00627     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00628     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00629     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00630     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
```

```
00631     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00632     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00633     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00634     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00635     1.914e-17
00636   };
00637
00638   static double no[121] = {
00639     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00640     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00641     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00642     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00643     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00644     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00645     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
00646     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00647     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00648     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00649     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00650     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00651     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00652     1.077e-08, 1.367e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00653     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00654     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00655     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00656     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00657     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00658     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00659     0.0001133
00660   };
00661
00662   static double no2[121] = {
00663     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00664     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00665     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00666     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00667     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00668     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00669     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00670     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00671     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00672     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00673     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00674     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00675     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00676     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00677     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00678     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00679     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00680     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00681   };
00682
00683   static double o3[121] = {
00684     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00685     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00686     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00687     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00688     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00689     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00690     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
00691     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00692     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00693     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00694     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
00695     4.509e-07, 3.922e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00696     2.476e-07, 2.284e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00697     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00698     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00699     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00700     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00701     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00702     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00703     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00704     3.665e-10
00705   };
00706
00707   static double ocs[121] = {
00708     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00709     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00710     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00711     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00712     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00713     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00714     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00715     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00716     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00717     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
```

```
00718     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00719     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00720     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00721     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00722     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00723     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00724     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00725     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00726     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00727     1.091e-14, 1.091e-14, 1.091e-14
00728   };
00729
00730   static double sf6[121] = {
00731     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00732     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
00733     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00734     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00735     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00736     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00737     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00738     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00739     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00740     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00741     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00742     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00743     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00744     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00745     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00746     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00747     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00748     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00749     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00750   };
00751
00752   static double so2[121] = {
00753     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00754     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00755     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00756     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00757     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00758     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00759     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00760     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00761     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00762     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00763     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00764     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00765     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00766     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00767     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00768   };
00769
00770   static int ig_co2 = -999;
00771
00772   double co2, *q[NG] = { NULL };
00773
00774   int ig, ip, iw, iz;
00775
00776   /* Find emitter index of CO2... */
00777   if (ig_co2 == -999)
00778     ig_co2 = find_emitter(ctl, "CO2");
00779
00780   /* Identify variable... */
00781   for (ig = 0; ig < ctl->ng; ig++) {
00782     q[ig] = NULL;
00783     if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00784       q[ig] = c2h2;
00785     if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00786       q[ig] = c2h6;
00787     if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00788       q[ig] = ccl4;
00789     if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00790       q[ig] = ch4;
00791     if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00792       q[ig] = clo;
00793     if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00794       q[ig] = clono2;
00795     if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00796       q[ig] = co;
00797     if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00798       q[ig] = cof2;
00799     if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00800       q[ig] = f11;
00801     if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00802       q[ig] = f12;
00803     if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00804       q[ig] = f14;
```

```
00805     if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00806       q[ig] = f22;
00807     if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00808       q[ig] = h2o;
00809     if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00810       q[ig] = h2o2;
00811     if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00812       q[ig] = hcn;
00813     if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00814       q[ig] = hno3;
00815     if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00816       q[ig] = hno4;
00817     if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00818       q[ig] = hocl;
00819     if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
00820       q[ig] = n2o;
00821     if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00822       q[ig] = n2o5;
00823     if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00824       q[ig] = nh3;
00825     if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00826       q[ig] = no;
00827     if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00828       q[ig] = no2;
00829     if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00830       q[ig] = o3;
00831     if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00832       q[ig] = ocs;
00833     if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00834       q[ig] = sf6;
00835     if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00836       q[ig] = so2;
00837   }
00838
00839   /* Loop over atmospheric data points... */
00840   for (ip = 0; ip < atm->np; ip++) {
00841
00842     /* Get altitude index... */
00843     iz = locate_reg(z, 121, atm->z[ip]);
00844
00845     /* Interpolate pressure... */
00846     atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00847
00848     /* Interpolate temperature... */
00849     atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00850
00851     /* Interpolate trace gases... */
00852     for (ig = 0; ig < ctl->ng; ig++)
00853       if (q[ig] != NULL)
00854         atm->q[ig][ip] =
00855           LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00856       else
00857         atm->q[ig][ip] = 0;
00858
00859     /* Set CO2... */
00860     if (ig_co2 >= 0) {
00861       co2 =
00862         371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00863       atm->q[ig_co2][ip] = co2;
00864     }
00865
00866     /* Set extinction to zero... */
00867     for (iw = 0; iw < ctl->nw; iw++)
00868       atm->k[iw][ip] = 0;
00869   }
00870 }
```

Here is the call graph for this function:



5.7.2.6 double ctmco2 (double *nu*, double *p*, double *t*, double *u*)

Compute carbon dioxide continuum (optical depth).

Definition at line 874 of file jurassic.c.

```
00878             {
00879
00880   static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00881     1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00882     1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00883     1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00884     2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00885     3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00886     4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00887     5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00888     7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00889     .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00890     .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00891     .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00892     .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00893     .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00894     .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00895     .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00896     .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00897     .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00898     .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00899     .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00900     .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00901     .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00902     .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00903     .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00904     .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00905     .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00906     .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
00907     .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00908     .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,
00909     .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,
00910     1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,
00911     2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,
00912     3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,
00913     4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,
00914     7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,
00915     12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,
00916     21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,
00917     35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,
00918     60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,
00919     107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,
00920     199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,
00921     386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,
00922     756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,
00923     1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,
00924     3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,
00925     5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,
00926     2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,
00927     820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,
00928     1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,
00929     6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,
```

```
00930     4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,
00931     1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,
00932     999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,
00933     476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,
00934     251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,
00935     133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,
00936     73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,
00937     42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,
00938     25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,
00939     15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,
00940     9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,
00941     5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,
00942     3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,
00943     2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,
00944     1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,
00945     1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,
00946     .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,
00947     .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,
00948     .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,
00949     .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,
00950     .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,
00951     .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,
00952     .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,
00953     .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,
00954     .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,
00955     .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,
00956     .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,
00957     .099745, .091118, .083404, .076494, .070292, .064716, .059697,
00958     .055173, .051093, .047411, .044089, .041092, .038392, .035965,
00959     .033789, .031846, .030122, .028607, .02729, .026169, .025209,
00960     .024405, .023766, .023288, .022925, .022716, .022681, .022685,
00961     .022768, .023133, .023325, .023486, .024004, .024126, .024083,
00962     .023785, .024023, .023029, .021649, .021108, .019454, .017809,
00963     .017292, .016635, .017037, .018068, .018977, .018756, .017847,
00964     .016557, .016142, .014459, .012869, .012381, .010875, .0098701,
00965     .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,
00966     .014362, .015017, .016507, .017741, .01768, .017784, .0171,
00967     .016357, .016172, .017257, .018978, .020935, .021741, .023567,
00968     .025183, .025589, .026732, .027648, .028278, .028215, .02856,
00969     .029015, .029062, .028851, .028497, .027825, .027801, .026523,
00970     .02487, .022967, .022168, .020194, .018605, .017903, .018439,
00971     .019697, .020311, .020855, .020057, .018608, .016738, .015963,
00972     .013844, .011801, .011134, .0097573, .0086007, .0086226,
00973     .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,
00974     .014657, .015771, .016351, .016079, .014829, .013431, .013185,
00975     .013207, .01448, .016176, .017971, .018265, .019526, .020455,
00976     .019797, .019802, .0194, .018176, .017505, .016197, .015339,
00977     .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,
00978     .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,
00979     .0041501, .003752, .0033996, .0030865, .0028077, .0025586,
00980     .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,
00981     .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,
00982     8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,
00983     5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,
00984     3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,
00985     2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,
00986     2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,
00987     1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,
00988     1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,
00989     1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,
00990     1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,
00991     2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,
00992     2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,
00993     3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,
00994     4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,
00995     6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,
00996     8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
00997     .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
00998     .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
00999     .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01000     .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01001     .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01002     .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01003     .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01004     .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01005     .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01006     .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01007     .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01008     .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01009     .056326, .0602, .064093, .068521, .073278, .077734, .083064,
01010     .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,
01011     .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,
01012     .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,
01013     .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,
01014     .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,
01015     .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,
01016     .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,
```

```
01017     1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,
01018     1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,
01019     1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,
01020     2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,
01021     4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,
01022     7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,
01023     9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,
01024     10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,
01025     16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,
01026     27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,
01027     43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,
01028     63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,
01029     98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,
01030     159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,
01031     264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,
01032     449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,
01033     789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,
01034     1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,
01035     2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,
01036     4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,
01037     9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,
01038     20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,
01039     40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,
01040     43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,
01041     44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,
01042     21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,
01043     28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,
01044     31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,
01045     54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,
01046     29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,
01047     11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,
01048     5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,
01049     2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,
01050     1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,
01051     845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,
01052     489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,
01053     291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,
01054     176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,
01055     108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,
01056     67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,
01057     42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,
01058     27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,
01059     17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,
01060     11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,
01061     7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,
01062     4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,
01063     3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,
01064     2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,
01065     1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,
01066     .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,
01067     .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,
01068     .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,
01069     .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,
01070     .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,
01071     .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,
01072     .093203, .088815, .084641, .080671, .076892, .073296, .069873,
01073     .066613, .06351, .060555, .05774, .055058, .052504, .050071,
01074     .047752, .045543, .043438, .041432, .039521, .037699, .035962,
01075     .034307, .032729, .031225, .029791, .028423, .02712, .025877,
01076     .024692, .023563, .022485, .021458, .020478, .019543, .018652,
01077     .017802, .016992, .016219, .015481, .014778, .014107, .013467,
01078     .012856, .012274, .011718, .011188, .010682, .0102, .0097393,
01079     .0093001, .008881, .0084812, .0080997, .0077358, .0073885,
01080     .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,
01081     .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,
01082     .0040884, .0039088, .0037373, .0035739, .003418, .0032693,
01083     .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,
01084     .0024074, .0023066, .0022108, .0021196, .002033, .0019507,
01085     .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,
01086     .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,
01087     .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,
01088     .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,
01089     9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,
01090     .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,
01091     .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,
01092     .0016144, .0016312, .001705, .0017892, .0018499, .0019715,
01093     .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,
01094     .0031193, .003346, .0034552, .0036906, .0037584, .0040084,
01095     .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,
01096     .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,
01097     .0060972, .0055539, .0055653, .0055772, .005331, .0054953,
01098     .0055919, .0058684, .006183, .0066675, .0069808, .0075142,
01099     .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,
01100     .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,
01101     .0105, .010617, .010706, .01078, .011177, .011212, .011304,
01102     .011446, .011603, .011816, .012165, .012545, .013069, .013539,
01103     .01411, .014776, .016103, .017016, .017994, .018978, .01998,
```

```
01104     .021799, .022745, .023681, .024627, .025562, .026992, .027958,
01105     .029013, .030154, .031402, .03228, .033651, .035272, .037088,
01106     .039021, .041213, .043597, .045977, .04877, .051809, .054943,
01107     .058064, .061528, .06537, .069309, .071928, .075752, .079589,
01108     .083352, .084096, .087497, .090817, .091198, .094966, .099045,
01109     .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,
01110     .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,
01111     .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,
01112     .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01113     .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01114     .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01115     1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01116     2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01117     3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01118     5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01119     10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01120     18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01121     30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01122     57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01123     107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01124     205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01125     394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01126     501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01127     455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01128     376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01129     367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01130     588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01131     962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01132     1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01133     751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01134     777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01135     322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01136     654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01137     679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01138     818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01139     364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01140     155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01141     74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01142     38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01143     21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01144     12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01145     7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01146     4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01147     2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01148     1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01149     1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01150     .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01151     .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01152     .26732, .25337, .24017, .22774, .21601, .20479, .19426
01153   };
01154
01155   static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01156     6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01157     9.2421e-5, 9.6846e-5, 1.0149e-4, 1.0635e-4, 1.1145e-4, 1.1679e-4,
01158     1.224e-4, 1.2828e-4, 1.3444e-4, 1.409e-4, 1.4768e-4, 1.5479e-4,
01159     1.6224e-4, 1.7006e-4, 1.7826e-4, 1.8685e-4, 1.9587e-4, 2.0532e-4,
01160     2.1524e-4, 2.2565e-4, 2.3656e-4, 2.48e-4, 2.6001e-4, 2.7261e-4,
01161     2.8582e-4, 2.9968e-4, 3.1422e-4, 3.2948e-4, 3.4548e-4, 3.6228e-4,
01162     3.799e-4, 3.9838e-4, 4.1778e-4, 4.3814e-4, 4.595e-4, 4.8191e-4,
01163     5.0543e-4, 5.3012e-4, 5.5603e-4, 5.8321e-4, 6.1175e-4, 6.417e-4,
01164     6.7314e-4, 7.0614e-4, 7.4078e-4, 7.7714e-4, 8.1531e-4, 8.5538e-4,
01165     8.9745e-4, 9.4162e-4, 9.8798e-4, .0010367, .0010878, .0011415,
01166     .0011978, .001257, .0013191, .0013844, .001453, .0015249,
01167     .0016006, .00168, .0017634, .001851, .001943, .0020397, .0021412,
01168     .0022479, .00236, .0024778, .0026015, .0027316, .0028682,
01169     .0030117, .0031626, .0033211, .0034877, .0036628, .0038469,
01170     .0040403, .0042436, .0044574, .004682, .0049182, .0051665,
01171     .0054276, .0057021, .0059907, .0062942, .0066133, .0069489,
01172     .0073018, .0076729, .0080632, .0084738, .0089056, .0093599,
01173     .0098377, .01034, .010869, .011426, .012011, .012627, .013276,
01174     .013958, .014676, .015431, .016226, .017063, .017944, .018872,
01175     .019848, .020876, .021958, .023098, .024298, .025561, .026892,
01176     .028293, .029769, .031323, .032961, .034686, .036503, .038418,
01177     .040435, .042561, .044801, .047161, .049649, .052271, .055035,
01178     .057948, .061019, .064256, .06767, .07127, .075066, .079069,
01179     .083291, .087744, .092441, .097396, .10262, .10814, .11396,
01180     .1201, .12658, .13342, .14064, .14826, .1563, .1648, .17376,
01181     .18323, .19324, .2038, .21496, .22674, .23919, .25234, .26624,
01182     .28093, .29646, .31287, .33021, .34855, .36794, .38844, .41012,
01183     .43305, .45731, .48297, .51011, .53884, .56924, .60141, .63547,
01184     .67152, .70969, .75012, .79292, .83826, .8863, .93718, .99111,
01185     1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,
01186     1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,
01187     2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,
01188     4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,
01189     6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,
01190     11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,
```

```
01191     18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,
01192     31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,
01193     54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,
01194     97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,
01195     182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,
01196     356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,
01197     716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,
01198     1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,
01199     3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,
01200     6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,
01201     2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,
01202     843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,
01203     1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381.,
01204     9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,
01205     4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,
01206     2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,
01207     1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,
01208     525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,
01209     268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,
01210     139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,
01211     76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,
01212     43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,
01213     25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,
01214     14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,
01215     9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,
01216     5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,
01217     3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,
01218     2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,
01219     1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,
01220     .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,
01221     .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,
01222     .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,
01223     .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,
01224     .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,
01225     .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,
01226     .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,
01227     .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,
01228     .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,
01229     .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,
01230     .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,
01231     .1152, .10304, .092437, .083163, .075031, .067878, .061564,
01232     .055976, .051018, .046609, .042679, .03917, .036032, .033223,
01233     .030706, .02845, .026428, .024617, .022998, .021554, .02027,
01234     .019136, .018141, .017278, .016541, .015926, .015432, .015058,
01235     .014807, .014666, .014635, .014728, .014947, .01527, .015728,
01236     .016345, .017026, .017798, .018839, .019752, .020636, .021886,
01237     .022695, .02327, .023478, .024292, .023544, .022222, .021932,
01238     .020052, .018143, .017722, .017031, .017782, .01938, .020734,
01239     .020476, .019255, .017477, .016878, .014617, .012489, .011765,
01240     .0099077, .0086446, .0079446, .0078644, .0079763, .008671,
01241     .01001, .0108, .012933, .015349, .016341, .018484, .020254,
01242     .020254, .020478, .019591, .018595, .018385, .019913, .022254,
01243     .024847, .025809, .028053, .029924, .030212, .031367, .03222,
01244     .032739, .032537, .03286, .033344, .033507, .033499, .033339,
01245     .032809, .033041, .031723, .029837, .027511, .026603, .024032,
01246     .021914, .020948, .021701, .023425, .024259, .024987, .023818,
01247     .021768, .019223, .018144, .015282, .012604, .01163, .0097907,
01248     .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,
01249     .014378, .016761, .01726, .018997, .019998, .019809, .01819,
01250     .016358, .016099, .01617, .017939, .020223, .022521, .02277,
01251     .024279, .025247, .024222, .023989, .023224, .021493, .020362,
01252     .018596, .017309, .015975, .014466, .013171, .011921, .01078,
01253     .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,
01254     .0051459, .0046273, .0041712, .0037686, .0034119, .003095,
01255     .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,
01256     .001636, .0015017, .00138, .0012697, .0011694, .0010782,
01257     9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,
01258     6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,
01259     4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,
01260     2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,
01261     1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,
01262     1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,
01263     1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,
01264     1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,
01265     1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,
01266     1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,
01267     1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,
01268     2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,
01269     2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,
01270     3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,
01271     4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,
01272     6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,
01273     8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,
01274     .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,
01275     .0014953, .001572, .0016529, .0017381, .0018279, .0019226,
01276     .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,
01277     .002747, .0028921, .0030453, .0032071, .003378, .0035586,
```

```
01278     .0037494, .003951, .0041642, .0043897, .0046282, .0048805,
01279     .0051476, .0054304, .00573, .0060473, .0063837, .0067404,
01280     .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,
01281     .0099366, .010516, .011134, .011792, .012494, .013244, .014046,
01282     .014898, .015808, .016781, .017822, .018929, .020108, .02138,
01283     .022729, .02419, .02576, .027412, .029233, .031198, .033301,
01284     .035594, .038092, .040767, .04372, .046918, .050246, .053974,
01285     .058009, .061976, .066586, .071537, .076209, .081856, .087998,
01286     .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,
01287     .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,
01288     .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,
01289     .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,
01290     .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,
01291     .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,
01292     .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,
01293     .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,
01294     1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,
01295     1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,
01296     2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,
01297     5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,
01298     7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,
01299     10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,
01300     11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,
01301     18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,
01302     29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,
01303     46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,
01304     70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,
01305     114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,
01306     190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,
01307     324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,
01308     568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,
01309     1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543., 1672.8, 1813.4,
01310     1966.1, 2131.4, 2309.5, 2499.3, 2705., 2925.7, 3161.6, 3411.3,
01311     3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,
01312     6719.6, 7385., 8145., 8977.7, 9831.9, 10827., 11934., 13063.,
01313     14434., 15878., 17591., 19435., 21510., 23835., 26835., 29740.,
01314     32878., 36305., 39830., 43273., 46931., 50499., 49586., 51598.,
01315     53429., 54619., 55081., 55102., 54485., 53487., 52042., 42689.,
01316     42607., 44020., 47994., 54169., 53916., 55808., 56642., 46049.,
01317     44243., 32929., 30658., 21963., 20835., 15962., 13679., 17652.,
01318     19680., 22388., 25625., 29184., 32520., 35720., 38414., 40523.,
01319     49228., 48173., 45678., 41768., 37600., 41313., 42654., 44465.,
01320     55736., 56630., 65409., 63308., 66572., 61845., 60379., 56777.,
01321     51920., 46601., 41367., 36529., 32219., 28470., 25192., 22362.,
01322     19907., 17772., 15907., 14273., 12835., 11567., 10445., 9450.2,
01323     8565.1, 7776., 7070.8, 6439.2, 5872.3, 5362.4, 4903., 4488.3,
01324     4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296.,
01325     2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236.,
01326     1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,
01327     643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,
01328     371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,
01329     219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,
01330     131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,
01331     80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,
01332     49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,
01333     30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,
01334     19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,
01335     12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,
01336     7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,
01337     5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,
01338     3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,
01339     2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,
01340     1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,
01341     .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,
01342     .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,
01343     .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,
01344     .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01345     .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01346     .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01347     .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01348     .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01349     .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01350     .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01351     .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01352     .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01353     .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01354     .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01355     .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01356     .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01357     .003388, .0032329, .0030852, .0029445, .0028105, .0026829,
01358     .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01359     .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
01360     .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01361     .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01362     9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01363     7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01364     6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
```

```
01365     6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01366     7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01367     9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01368     .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01369     .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01370     .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01371     .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01372     .0053809, .0056699, .0059325, .0055488, .005634, .0056392,
01373     .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01374     .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01375     .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01376     .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01377     .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01378     .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01379     .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01380     .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01381     .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01382     .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01383     .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01384     .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01385     .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01386     .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01387     .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01388     .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01389     .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01390     .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01391     1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01392     1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01393     2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01394     4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01395     8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01396     15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01397     27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01398     46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01399     87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01400     176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01401     366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01402     600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01403     478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01404     545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01405     314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01406     696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01407     954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01408     1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01409     1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01410     1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01411     361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01412     578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01413     659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01414     1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01415     489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01416     199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01417     92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01418     46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01419     24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01420     13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01421     7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01422     4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01423     2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01424     1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01425     1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01426     .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01427     .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01428     .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
01429     .16469
01430   };
01431
01432   static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01433     3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01434     4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01435     6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01436     8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01437     1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01438     1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01439     1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01440     2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01441     3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01442     4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01443     6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01444     9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
01445     .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01446     .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
01447     .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01448     .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01449     .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01450     .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01451     .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
```

```
01452     .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01453     .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01454     .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01455     .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01456     .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01457     .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01458     .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01459     .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,
01460     .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01461     .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01462     .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01463     1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01464     1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01465     2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01466     4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01467     7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01468     13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01469     22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01470     40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01471     74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01472     137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01473     263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01474     525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01475     1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01476     2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01477     5457.9, 5946.2, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01478     11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01479     1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01480     722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01481     2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01482     7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01483     4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01484     1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01485     783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01486     387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01487     197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01488     105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01489     57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01490     31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01491     17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01492     10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01493     6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01494     3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01495     2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01496     1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01497     .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01498     .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01499     .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01500     .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01501     .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01502     .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01503     .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01504     .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01505     .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01506     .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01507     .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01508     .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01509     .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01510     .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01511     .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01512     .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01513     .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01514     .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01515     .022311, .023742, .0249, .025599, .026981, .026462, .025143,
01516     .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01517     .024163, .023728, .02199, .019506, .018591, .015576, .012784,
01518     .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01519     .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01520     .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01521     .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01522     .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01523     .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01524     .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01525     .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01526     .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01527     .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01528     .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01529     .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01530     .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01531     .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
01532     .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,
01533     .0031747, .0028864, .0026284, .002397, .002189, .0020017,
01534     .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,
01535     .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,
01536     6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,
01537     4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,
01538     2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,
```

```
01539     1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,
01540     1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,
01541     9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,
01542     7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,
01543     6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,
01544     7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,
01545     8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,
01546     9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,
01547     1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,
01548     1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,
01549     2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,
01550     3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,
01551     4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,
01552     5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,
01553     7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,
01554     .0010922, .001154, .0012195, .0012889, .0013626, .0014407,
01555     .0015235, .0016114, .0017048, .0018038, .001909, .0020207,
01556     .0021395, .0022657, .0023998, .0025426, .0026944, .002856,
01557     .0030281, .0032114, .0034068, .003615, .0038371, .004074,
01558     .004327, .0045971, .0048857, .0051942, .0055239, .0058766,
01559     .0062538, .0066573, .0070891, .007551, .0080455, .0085747,
01560     .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,
01561     .014415, .01541, .016475, .017621, .018857, .020175, .02162,
01562     .023185, .024876, .02672, .028732, .030916, .033319, .035939,
01563     .038736, .041847, .04524, .048715, .052678, .056977, .061203,
01564     .066184, .07164, .076952, .083477, .090674, .098049, .10697,
01565     .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,
01566     .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,
01567     .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,
01568     .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,
01569     .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,
01570     .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,
01571     .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,
01572     1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,
01573     1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,
01574     1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,
01575     3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,
01576     6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,
01577     8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,
01578     7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,
01579     11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,
01580     21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,
01581     35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,
01582     51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,
01583     83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,
01584     142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,
01585     249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,
01586     449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,
01587     833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350., 1466.3,
01588     1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,
01589     3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,
01590     5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203.,
01591     11223., 12358., 13493., 14916., 16416., 18236., 20222., 22501.,
01592     25102., 28358., 31707., 35404., 39538., 43911., 48391., 53193.,
01593     58028., 58082., 61276., 64193., 66294., 67480., 67921., 67423.,
01594     66254., 64341., 51737., 51420., 53072., 58145., 66195., 65358.,
01595     67377., 67869., 53509., 50553., 35737., 32425., 21704., 19974.,
01596     14457., 12142., 16798., 19489., 23049., 27270., 31910., 36457.,
01597     40877., 44748., 47876., 59793., 58626., 55454., 50337., 44893.,
01598     50228., 52216., 54747., 69541., 70455., 81014., 77694., 80533.,
01599     73953., 70927., 65539., 59002., 52281., 45953., 40292., 35360.,
01600     31124., 27478., 24346., 21647., 19308., 17271., 15491., 13927.,
01601     12550., 11331., 10250., 9288.8, 8431.4, 7664.9, 6978.3, 6361.8,
01602     5807.4, 5307.7, 4856.8, 4449., 4079.8, 3744.9, 3440.8, 3164.2,
01603     2912.3, 2682.7, 2473., 2281.4, 2106., 1945.3, 1797.9, 1662.5,
01604     1538.1, 1423.6, 1318.1, 1221., 1131.5, 1049., 972.99, 902.87,
01605     838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,
01606     462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,
01607     258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,
01608     147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,
01609     86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,
01610     51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,
01611     30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,
01612     18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,
01613     11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,
01614     7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,
01615     4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,
01616     2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,
01617     1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,
01618     1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,
01619     .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,
01620     .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,
01621     .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,
01622     .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,
01623     .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,
01624     .092236, .087844, .083443, .07938, .075452, .071564, .067931,
01625     .064389, .061078, .057901, .054921, .052061, .049364, .046789,
```

```
01626     .04435, .042044, .039866, .037808, .035863, .034023, .032282,
01627     .030634, .029073, .027595, .026194, .024866, .023608, .022415,
01628     .021283, .02021, .019193, .018228, .017312, .016443, .015619,
01629     .014837, .014094, .01339, .012721, .012086, .011483, .010911,
01630     .010368, .009852, .0093623, .0088972, .0084556, .0080362,
01631     .0076379, .0072596, .0069003, .006559, .0062349, .0059269,
01632     .0056344, .0053565, .0050925, .0048417, .0046034, .004377,
01633     .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,
01634     .0030785, .002928, .0027851, .0026492, .0025201, .0023975,
01635     .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,
01636     .0016938, .0016122, .0015348, .0014612, .0013913, .001325,
01637     .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,
01638     9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,
01639     7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,
01640     5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,
01641     4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,
01642     3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,
01643     3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,
01644     3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,
01645     4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,
01646     6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,
01647     8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,
01648     .001282, .0014016, .0015355, .0016764, .0018272, .0020055,
01649     .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,
01650     .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,
01651     .0052989, .0056148, .0052452, .0053357, .005333, .0045069,
01652     .0043851, .004253, .003738, .0038084, .0039013, .0041505,
01653     .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,
01654     .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,
01655     .0076269, .0063758, .006254, .0067749, .0067909, .0068231,
01656     .0072143, .0072762, .0072954, .007679, .0075107, .0073658,
01657     .0072441, .0071074, .0070378, .007176, .0072472, .0075844,
01658     .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,
01659     .013895, .015567, .016011, .016392, .016737, .017043, .017731,
01660     .018031, .018419, .018877, .019474, .019868, .020604, .021538,
01661     .022653, .023869, .025288, .026879, .028547, .030524, .03274,
01662     .035132, .03769, .040567, .043793, .047188, .049962, .053542,
01663     .057205, .060776, .061489, .064419, .067124, .065945, .068487,
01664     .071209, .074783, .077039, .082444, .08902, .09692, .10617,
01665     .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,
01666     .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,
01667     .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,
01668     .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,
01669     .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,
01670     .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,
01671     1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,
01672     2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,
01673     4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,
01674     8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,
01675     15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,
01676     28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,
01677     46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,
01678     86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,
01679     179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,
01680     397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,
01681     719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,
01682     580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,
01683     615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,
01684     316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,
01685     813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,
01686     1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,
01687     1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,
01688     1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,
01689     1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,
01690     338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.85,
01691     662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,
01692     803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01693     1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01694     523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01695     211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01696     97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01697     49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01698     26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01699     14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01700     7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01701     4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01702     2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01703     1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01704     .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01705     .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
01706     .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01707     .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
01708     .12584
01709   };
01710
01711   double xw, dw, ew, cw296, cw260, cw230, dt230, dt260, dt296, ctw, ctmpth;
01712
```

```
01713   int iw;
01714
01715   /* Get CO2 continuum absorption... */
01716   xw = nu / 2 + 1;
01717   if (xw >= 1 && xw < 2001) {
01718     iw = (int) xw;
01719     dw = xw - iw;
01720     ew = 1 - dw;
01721     cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01722     cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01723     cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01724     dt230 = t - 230;
01725     dt260 = t - 260;
01726     dt296 = t - 296;
01727     ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01728       * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01729     ctmpth = u / NA / 1000 * p / P0 * ctw;
01730   } else
01731     ctmpth = 0;
01732   return ctmpth;
01733 }
```

5.7.2.7 double ctmh2o (double *nu*, double *p*, double *t*, double *q*, double *u*)

Compute water vapor continuum (optical depth).

Definition at line 1737 of file jurassic.c.

```
01742             {
01743
01744   static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01745     .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01746     .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01747     .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01748     .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01749     .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01750     .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01751     6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01752     3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01753     1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01754     1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01755     6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01756     4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01757     3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01758     2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01759     1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01760     1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01761     1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01762     1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01763     1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01764     2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01765     4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01766     1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01767     2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01768     3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01769     3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01770     3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,
01771     2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,
01772     1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,
01773     4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,
01774     2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,
01775     1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,
01776     5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,
01777     2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,
01778     1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,
01779     7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,
01780     4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,
01781     2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,
01782     1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,
01783     1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,
01784     9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,
01785     7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,
01786     6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,
01787     5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,
01788     5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,
01789     5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,
01790     7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,
01791     1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,
01792     3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,
01793     6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,
```

```
01794     1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,
01795     1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,
01796     1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,
01797     1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,
01798     1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,
01799     3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,
01800     7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,
01801     1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,
01802     3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,
01803     7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,
01804     1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,
01805     1.387e-4, 1.363e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,
01806     1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,
01807     5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,
01808     2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,
01809     1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,
01810     5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,
01811     2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,
01812     1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,
01813     6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,
01814     3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,
01815     1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,
01816     1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,
01817     6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,
01818     4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,
01819     3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,
01820     2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,
01821     2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,
01822     2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,
01823     4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,
01824     8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,
01825     1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,
01826     3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,
01827     8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,
01828     2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,
01829     4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,
01830     8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,
01831     1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,
01832     9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,
01833     4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,
01834     1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,
01835     9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,
01836     4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,
01837     2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,
01838     1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,
01839     5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,
01840     2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,
01841     1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,
01842     8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,
01843     5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,
01844     3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,
01845     2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,
01846     2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,
01847     1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,
01848     2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,
01849     3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,
01850     5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,
01851     1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,
01852     2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,
01853     5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,
01854     1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,
01855     2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,
01856     3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,
01857     4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,
01858     5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01859     7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01860     1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01861     2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01862     4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01863     5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
01864     4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01865     2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01866     9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01867     4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01868     2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01869     1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01870     5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01871     2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,
01872     1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,
01873     7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,
01874     4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,
01875     3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,
01876     2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,
01877     2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,
01878     2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,
01879     4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,
01880     6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,
```

```
01881     9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,
01882     1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,
01883     1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,
01884     1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,
01885     2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,
01886     6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,
01887     1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,
01888     2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,
01889     2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,
01890     2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,
01891     1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,
01892     4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,
01893     2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,
01894     1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,
01895     5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,
01896     2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,
01897     1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,
01898     7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,
01899     4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,
01900     2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,
01901     1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,
01902     1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,
01903     1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,
01904     1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,
01905     2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,
01906     2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,
01907     3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,
01908     3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,
01909     6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,
01910     1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,
01911     3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,
01912     6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,
01913     1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,
01914     2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,
01915     2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,
01916     3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,
01917     7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,
01918     1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,
01919     1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,
01920     1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,
01921     5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,
01922     2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,
01923     1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,
01924     1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,
01925     1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,
01926     1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,
01927     1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,
01928     5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,
01929     2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,
01930     1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,
01931     5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,
01932     3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,
01933     1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,
01934     9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,
01935     7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,
01936     8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,
01937     1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,
01938     2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,
01939     3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,
01940     6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,
01941     1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,
01942     2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,
01943     4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,
01944     7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,
01945     7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,
01946     5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,
01947     2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,
01948     1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,
01949     8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,
01950     7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,
01951     7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,
01952     6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,
01953     3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,
01954     1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,
01955     7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,
01956     3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,
01957     1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,
01958     1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,
01959     6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,
01960     4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,
01961     4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,
01962     7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,
01963     1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,
01964     3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,
01965     7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,
01966     1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,
01967     3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,
```

```
01968     7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,
01969     1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,
01970     2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,
01971     4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,
01972     5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,
01973     5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,
01974     3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,
01975     1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,
01976     8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,
01977     5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,
01978     5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,
01979     8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,
01980     1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,
01981     1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,
01982     7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,
01983     3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,
01984     1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,
01985     7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,
01986     4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,
01987     2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,
01988     1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,
01989     6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,
01990     6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,
01991     8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,
01992     1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,
01993     3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,
01994     6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,
01995     1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,
01996     2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,
01997     3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,
01998     4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,
01999     3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,
02000     1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,
02001     7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,
02002     4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,
02003     3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,
02004     4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,
02005     7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,
02006     8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,
02007     6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,
02008     3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,
02009     1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,
02010     6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,
02011     3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,
02012     1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,
02013     1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,
02014     1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,
02015     2.534e-12, 2.858e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,
02016     5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,
02017     1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,
02018     2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,
02019     5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,
02020     1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,
02021     2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,
02022     4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,
02023     6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,
02024     5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,
02025     3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,
02026     1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,
02027     7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,
02028     4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,
02029     2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,
02030     2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,
02031     4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,
02032     6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,
02033     8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,
02034     8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,
02035     5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,
02036     2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,
02037     1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,
02038     5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,
02039     2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,
02040     1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,
02041     1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,
02042     1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,
02043     2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,
02044     5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,
02045     1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,
02046     2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,
02047     3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,
02048     4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,
02049     3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,
02050     2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,
02051     1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,
02052     5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,
02053     2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,
02054     1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,
```

```
02055     1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,
02056     1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,
02057     2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,
02058     3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,
02059     3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,
02060     2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,
02061     1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,
02062     5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,
02063     4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,
02064     5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,
02065     9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,
02066     2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,
02067     4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,
02068     1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,
02069     2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,
02070     4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,
02071     7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,
02072     7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,
02073     5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,
02074     2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02075     1.093e-11, 9.558e-12
02076   };
02077
02078   static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02079     .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02080     .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02081     .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02082     .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02083     .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02084     .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02085     .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02086     5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02087     3.366e-4, 3.085e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02088     2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02089     1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02090     9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02091     6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02092     4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02093     3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02094     2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02095     2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02096     2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02097     2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02098     3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02099     7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02100     1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02101     3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02102     5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02103     6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02104     5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02105     2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02106     1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02107     6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02108     2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02109     1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02110     7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02111     4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02112     2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02113     1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02114     9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02115     6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02116     4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02117     3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02118     2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
02119     1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,
02120     1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,
02121     1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,
02122     1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,
02123     1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,
02124     1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,
02125     2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,
02126     5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,
02127     1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,
02128     1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,
02129     2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,
02130     2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,
02131     2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,
02132     2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,
02133     5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,
02134     1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,
02135     2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,
02136     5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,
02137     1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,
02138     1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,
02139     2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,
02140     1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,
02141     7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,
```

```
02142     2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,
02143     1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,
02144     7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,
02145     3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,
02146     1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,
02147     1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,
02148     5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,
02149     3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,
02150     2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,
02151     1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,
02152     9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,
02153     7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,
02154     5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,
02155     5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,
02156     6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,
02157     9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,
02158     1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,
02159     3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,
02160     6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,
02161     1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,
02162     2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,
02163     6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,
02164     1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,
02165     1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,
02166     1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,
02167     7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,
02168     3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,
02169     1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,
02170     7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,
02171     3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,
02172     1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,
02173     9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,
02174     5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,
02175     3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,
02176     1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,
02177     1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,
02178     9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,
02179     6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,
02180     4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,
02181     4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,
02182     4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,
02183     5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,
02184     9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,
02185     1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,
02186     3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,
02187     7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,
02188     1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,
02189     3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,
02190     5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,
02191     7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,
02192     7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,
02193     8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,
02194     1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,
02195     3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,
02196     7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,
02197     8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,
02198     6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,
02199     2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,
02200     1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,
02201     6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,
02202     3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,
02203     1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,
02204     8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,
02205     4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,
02206     2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,
02207     1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,
02208     9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,
02209     6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,
02210     5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,
02211     4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,
02212     4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,
02213     6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,
02214     1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,
02215     1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,
02216     1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,
02217     1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,
02218     2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,
02219     3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,
02220     8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,
02221     2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,
02222     3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,
02223     4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,
02224     3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,
02225     1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,
02226     6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,
02227     3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,
02228     1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,
```

```
02229     9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,
02230     4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,
02231     2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,
02232     1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,
02233     7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,
02234     5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,
02235     3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,
02236     2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,
02237     2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,
02238     2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,
02239     3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,
02240     4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,
02241     5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,
02242     6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,
02243     1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,
02244     2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,
02245     4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,
02246     9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,
02247     2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,
02248     3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,
02249     3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,
02250     5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,
02251     1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,
02252     1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,
02253     2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,
02254     1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,
02255     6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,
02256     3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,
02257     2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,
02258     2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,
02259     2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,
02260     2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,
02261     1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,
02262     6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,
02263     3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,
02264     1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,
02265     8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,
02266     4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,
02267     2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,
02268     1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,
02269     1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,
02270     1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,
02271     2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,
02272     4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,
02273     6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,
02274     9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,
02275     1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,
02276     3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,
02277     7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,
02278     1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,
02279     1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,
02280     7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,
02281     2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,
02282     1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,
02283     1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,
02284     1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,
02285     1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,
02286     9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,
02287     5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,
02288     2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,
02289     1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,
02290     5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,
02291     3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,
02292     1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,
02293     1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,
02294     8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,
02295     9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,
02296     1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,
02297     3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,
02298     5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,
02299     1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,
02300     2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,
02301     5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,
02302     1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,
02303     2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,
02304     3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,
02305     7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,
02306     9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,
02307     8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,
02308     4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,
02309     2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,
02310     1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,
02311     8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,
02312     9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,
02313     1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,
02314     1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,
02315     1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,
```

```
02316     9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,
02317     4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,
02318     2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,
02319     1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,
02320     5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,
02321     3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,
02322     1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,
02323     1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,
02324     1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,
02325     1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,
02326     2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,
02327     5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,
02328     8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,
02329     1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,
02330     4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,
02331     6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,
02332     7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,
02333     5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,
02334     2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,
02335     1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,
02336     6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,
02337     5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,
02338     8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,
02339     1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,
02340     1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,
02341     8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,
02342     3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,
02343     1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,
02344     9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,
02345     5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,
02346     2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,
02347     2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,
02348     2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,
02349     4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,
02350     9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,
02351     1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,
02352     3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,
02353     8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,
02354     2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,
02355     4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,
02356     7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,
02357     9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,
02358     8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,
02359     5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,
02360     2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,
02361     1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,
02362     5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,
02363     3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,
02364     4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,
02365     6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,
02366     1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,
02367     1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,
02368     1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,
02369     7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,
02370     3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,
02371     1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,
02372     8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,
02373     4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,
02374     2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,
02375     2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,
02376     2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,
02377     5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,
02378     1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,
02379     2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,
02380     4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02381     7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02382     9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02383     8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02384     5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02385     2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
02386     9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02387     5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02388     2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02389     2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02390     3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02391     6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02392     9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02393     9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
02394     5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02395     2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02396     1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02397     1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
02398     1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02399     3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02400     7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02401     1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02402     3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
```

```
02403     8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02404     1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02405     2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02406     2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02407     1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02408     7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02409     3.511e-11
02410   };
02411
02412   static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02413     .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02414     .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02415     .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02416     4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02417     1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02418     3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02419     1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02420     4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02421     1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,
02422     7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,
02423     2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,
02424     1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,
02425     5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,
02426     2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,
02427     1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,
02428     5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,
02429     2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,
02430     3.413e-9, 3.738e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,
02431     8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,
02432     3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,
02433     1.889e-7, 2.589e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,
02434     1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,
02435     7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,
02436     2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,
02437     4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,
02438     2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,
02439     3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,
02440     2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,
02441     8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,
02442     2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,
02443     7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,
02444     2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,
02445     7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,
02446     2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,
02447     8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,
02448     3.157e-9, 2.725e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,
02449     1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,
02450     5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,
02451     2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,
02452     1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,
02453     5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,
02454     1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,
02455     9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,
02456     1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,
02457     1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,
02458     1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,
02459     6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,
02460     1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,
02461     1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,
02462     7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,
02463     1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,
02464     7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,
02465     1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,
02466     7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,
02467     4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,
02468     5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,
02469     1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,
02470     4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,
02471     1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,
02472     7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,
02473     1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,
02474     1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,
02475     1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,
02476     6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,
02477     8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,
02478     1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,
02479     4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,
02480     1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,
02481     6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,
02482     2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,
02483     9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,
02484     3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,
02485     1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,
02486     9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,
02487     2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,
02488     3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,
02489     2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,
```

```
02490     2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,
02491     1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,
02492     1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,
02493     1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,
02494     4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,
02495     1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,
02496     7.611e-9, 9.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,
02497     4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,
02498     3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,
02499     7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,
02500     8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,
02501     1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,
02502     7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,
02503     1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,
02504     2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,
02505     6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,
02506     2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,
02507     8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,
02508     3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,
02509     1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,
02510     5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,
02511     2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,
02512     2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,
02513     1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,
02514     6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,
02515     6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,
02516     6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,
02517     3.311e-12, 3.346e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,
02518     4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,
02519     1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,
02520     4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,
02521     2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,
02522     1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,
02523     4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,
02524     1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,
02525     2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,
02526     3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,
02527     4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,
02528     3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,
02529     7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,
02530     2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,
02531     4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,
02532     4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,
02533     5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,
02534     2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,
02535     3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,
02536     7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,
02537     3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,
02538     1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,
02539     4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,
02540     1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,
02541     5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,
02542     2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,
02543     9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,
02544     4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,
02545     3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,
02546     1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,
02547     9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,
02548     2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,
02549     4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,
02550     6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,
02551     6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,
02552     9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,
02553     5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,
02554     4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02555     2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02556     1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02557     1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02558     2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02559     2.859e-8, 2.839e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
02560     1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02561     2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02562     8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02563     3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02564     1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02565     6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02566     1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02567     6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
02568     2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,
02569     7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,
02570     2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,
02571     1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,
02572     4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,
02573     4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,
02574     1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,
02575     1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,
02576     2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,
```

```
02577     1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,
02578     1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,
02579     1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,
02580     5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,
02581     3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,
02582     9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,
02583     1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,
02584     1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,
02585     2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,
02586     5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,
02587     9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,
02588     1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,
02589     1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,
02590     6.403e-9, 4.643e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,
02591     9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,
02592     4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,
02593     9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,
02594     1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,
02595     1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,
02596     1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,
02597     6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,
02598     7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,
02599     1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,
02600     5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,
02601     1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,
02602     5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,
02603     2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,
02604     1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,
02605     8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,
02606     5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,
02607     1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,
02608     1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,
02609     2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,
02610     3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,
02611     1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,
02612     4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,
02613     5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,
02614     7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,
02615     7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,
02616     2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,
02617     5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,
02618     3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,
02619     5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,
02620     5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,
02621     5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,
02622     5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,
02623     1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,
02624     1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,
02625     3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,
02626     1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,
02627     3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,
02628     1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,
02629     6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,
02630     3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,
02631     2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,
02632     1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,
02633     3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,
02634     5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,
02635     1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,
02636     1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,
02637     4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,
02638     6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,
02639     1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,
02640     4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,
02641     5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,
02642     5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,
02643     5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,
02644     1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,
02645     2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,
02646     2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,
02647     3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,
02648     6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,
02649     9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,
02650     8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,
02651     1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,
02652     2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,
02653     4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,
02654     2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,
02655     1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,
02656     4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,
02657     5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,
02658     1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,
02659     9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,
02660     5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,
02661     4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,
02662     1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,
02663     1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,
```

```
02664     7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,
02665     2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,
02666     3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,
02667     3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,
02668     4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,
02669     1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,
02670     4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,
02671     1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,
02672     1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,
02673     3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,
02674     5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,
02675     6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,
02676     8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,
02677     4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,
02678     4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,
02679     5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,
02680     1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,
02681     5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,
02682     2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,
02683     1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,
02684     8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,
02685     2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,
02686     1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,
02687     1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,
02688     1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,
02689     6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,
02690     2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,
02691     4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,
02692     5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,
02693     5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,
02694     5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,
02695     1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,
02696     2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,
02697     6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,
02698     5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,
02699     9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,
02700     2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,
02701     5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,
02702     7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,
02703     7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,
02704     8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,
02705     1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,
02706     2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,
02707     9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,
02708     2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,
02709     4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,
02710     2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,
02711     2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,
02712     3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,
02713     1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,
02714     6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,
02715     1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,
02716     3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,
02717     3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,
02718     3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,
02719     3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,
02720     1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,
02721     1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,
02722     3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,
02723     9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,
02724     1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,
02725     8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,
02726     2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,
02727     3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,
02728     3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02729     3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02730     5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02731     4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02732     1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02733     6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
02734     9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02735     1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02736     1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02737     3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02738     1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02739     4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02740     6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02741     6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
02742     7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02743     2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02744     4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02745   };
02746
02747   static double xfcrev[15] =
02748     { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02749     1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02750   };
```

```
02751
02752   double a1, a2, a3, dw, ew, dx, xw, xx, vf2, vf6, cw260, cw296,
02753     sfac, fscal, cwfrn, ctmpth, ctwfrn, ctwslf;
02754
02755   int iw, ix;
02756
02757   /* Get H2O continuum absorption... */
02758   xw = nu / 10 + 1;
02759   if (xw >= 1 && xw < 2001) {
02760     iw = (int) xw;
02761     dw = xw - iw;
02762     ew = 1 - dw;
02763     cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02764     cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02765     cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02766     if (nu <= 820 || nu >= 960) {
02767       sfac = 1;
02768     } else {
02769       xx = (nu - 820) / 10;
02770       ix = (int) xx;
02771       dx = xx - ix;
02772       sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];
02773     }
02774     ctwslf = sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02775     vf2 = POW2(nu - 370);
02776     vf6 = POW3(vf2);
02777     fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02778     ctwfrn = cwfrn * fscal;
02779     a1 = nu * u * tanh(.7193876 / t * nu);
02780     a2 = 296 / t;
02781     a3 = p / P0 * (q * ctwslf + (1 - q) * ctwfrn) * 1e-20;
02782     ctmpth = a1 * a2 * a3;
02783   } else
02784     ctmpth = 0;
02785   return ctmpth;
02786 }
```

5.7.2.8 double ctmn2 (double *nu*, double *p*, double *t*)

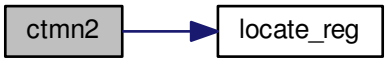
Compute nitrogen continuum (absorption coefficient).

Definition at line 2790 of file jurassic.c.

```
02793           {
02794
02795   static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02796     1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02797     2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02798     5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02799     7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02800     9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02801     1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02802     1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02803     1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02804     1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02805     7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,
02806     3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02807     1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02808     7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02809   };
02810
02811   static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02812     511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02813     233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02814     -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02815     -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02816     -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02817     -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02818     121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02819     133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,
02820     372., 449., 514., 569., 609., 642., 673., 673.
02821   };
02822
02823   static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02824     2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
02825     2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02826     2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02827     2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02828     2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02829     2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
```

```
02830     2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02831     2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02832     2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02833     2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02834   };
02835
02836   double b, beta, q_n2 = 0.79, t0 = 273, tr = 296;
02837
02838   int idx;
02839
02840   /* Check wavenumber range... */
02841   if (nu < nua[0] || nu > nua[97])
02842     return 0;
02843
02844   /* Interpolate B and beta... */
02845   idx = locate_reg(nua, 98, nu);
02846   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02847   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02848
02849   /* Compute absorption coefficient... */
02850   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02851     * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02852 }
```

Here is the call graph for this function:



5.7.2.9 double ctmo2 (double *nu*, double *p*, double *t*)

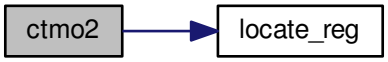
Compute oxygen continuum (absorption coefficient).

Definition at line 2856 of file jurassic.c.

```
02859             {
02860
02861   static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,
02862     .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,
02863     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02864     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02865     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02866     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02867     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02868     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02869     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02870     .071, .064, 0.
02871   };
02872
02873   static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02874     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02875     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02876     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,
02877     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02878     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02879     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02880     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02881     346., 322., 291., 290., 350., 371., 504., 504.
02882   };
02883
02884   static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02885     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02886     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02887     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02888     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
```

```
02889     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02890     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02891     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02892     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02893     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02894     1800., 1805.
02895   };
02896
02897   double b, beta, q_o2 = 0.21, t0 = 273, tr = 296;
02898
02899   int idx;
02900
02901   /* Check wavenumber range... */
02902   if (nu < nua[0] || nu > nua[89])
02903     return 0;
02904
02905   /* Interpolate B and beta... */
02906   idx = locate_reg(nua, 90, nu);
02907   b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02908   beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02909
02910   /* Compute absorption coefficient... */
02911   return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02912     b;
02913 }
```

Here is the call graph for this function:



5.7.2.10 void copy_atm (ctl_t * *ctl*, atm_t * *atm_dest*, atm_t * *atm_src*, int *init*)

Copy and initialize atmospheric data.

Definition at line 2917 of file jurassic.c.

```
02921             {
02922
02923   int ig, ip, iw;
02924
02925   size_t s;
02926
02927   /* Data size... */
02928   s = (size_t) atm_src->np * sizeof(double);
02929
02930   /* Copy data... */
02931   atm_dest->np = atm_src->np;
02932   memcpy(atm_dest->time, atm_src->time, s);
02933   memcpy(atm_dest->z, atm_src->z, s);
02934   memcpy(atm_dest->lon, atm_src->lon, s);
02935   memcpy(atm_dest->lat, atm_src->lat, s);
02936   memcpy(atm_dest->p, atm_src->p, s);
02937   memcpy(atm_dest->t, atm_src->t, s);
02938   for (ig = 0; ig < ctl->ng; ig++)
02939     memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02940   for (iw = 0; iw < ctl->nw; iw++)
02941     memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02942
02943   /* Initialize... */
02944   if (init)
02945     for (ip = 0; ip < atm_dest->np; ip++) {
02946       atm_dest->p[ip] = 0;
02947       atm_dest->t[ip] = 0;
02948       for (ig = 0; ig < ctl->ng; ig++)
02949         atm_dest->q[ig][ip] = 0;
02950       for (iw = 0; iw < ctl->nw; iw++)
02951         atm_dest->k[iw][ip] = 0;
02952     }
02953 }
```

5.7.2.11 void copy_obs (ctl_t * ctl, obs_t * obs_dest, obs_t * obs_src, int init)

Copy and initialize observation data.

Definition at line 2957 of file jurassic.c.

```
02961             {
02962
02963   int id, ir;
02964
02965   size_t s;
02966
02967   /* Data size... */
02968   s = (size_t) obs_src->nr * sizeof(double);
02969
02970   /* Copy data... */
02971   obs_dest->nr = obs_src->nr;
02972   memcpy(obs_dest->time, obs_src->time, s);
02973   memcpy(obs_dest->obsz, obs_src->obsz, s);
02974   memcpy(obs_dest->obslon, obs_src->obslon, s);
02975   memcpy(obs_dest->obslat, obs_src->obslat, s);
02976   memcpy(obs_dest->vpz, obs_src->vpz, s);
02977   memcpy(obs_dest->vplon, obs_src->vplon, s);
02978   memcpy(obs_dest->vplat, obs_src->vplat, s);
02979   memcpy(obs_dest->tpz, obs_src->tpz, s);
02980   memcpy(obs_dest->tplon, obs_src->tplon, s);
02981   memcpy(obs_dest->tplat, obs_src->tplat, s);
02982   for (id = 0; id < ctl->nd; id++)
02983     memcpy(obs_dest->rad[id], obs_src->rad[id], s);
02984   for (id = 0; id < ctl->nd; id++)
02985     memcpy(obs_dest->tau[id], obs_src->tau[id], s);
02986
02987   /* Initialize... */
02988   if (init)
02989     for (id = 0; id < ctl->nd; id++)
02990       for (ir = 0; ir < obs_dest->nr; ir++)
02991         if (gsl_finite(obs_dest->rad[id][ir])) {
02992           obs_dest->rad[id][ir] = 0;
02993           obs_dest->tau[id][ir] = 0;
02994         }
02995 }
```

5.7.2.12 int find_emitter (ctl_t * ctl, const char * emitter)

Find index of an emitter.

Definition at line 2999 of file jurassic.c.

```
03001                         {
03002
03003   int ig;
03004
03005   for (ig = 0; ig < ctl->ng; ig++)
03006     if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03007       return ig;
03008
03009   return -1;
03010 }
```

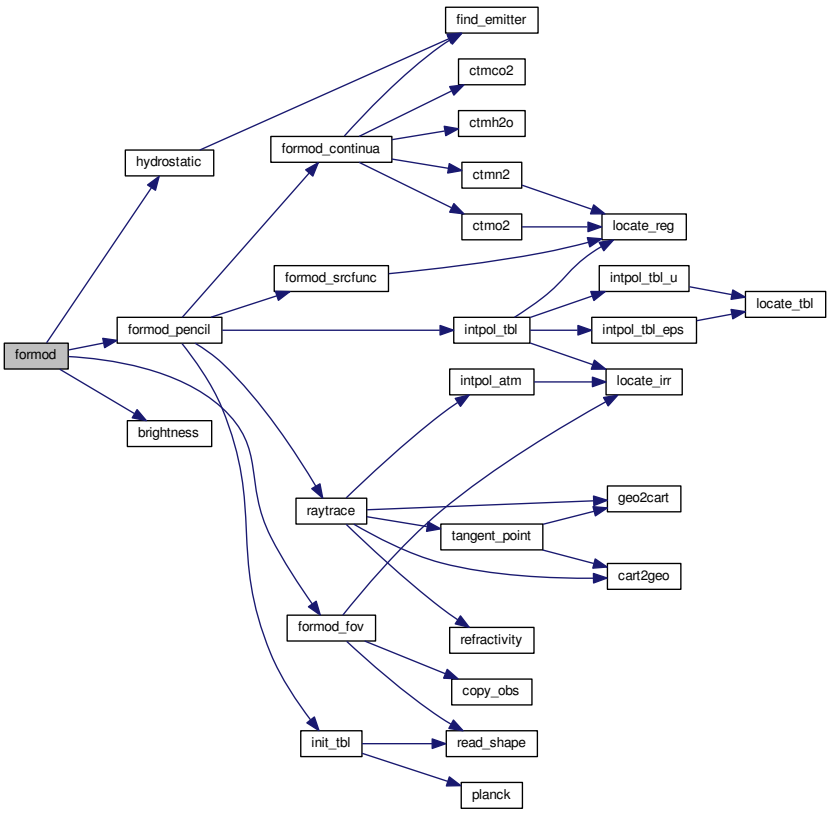
5.7.2.13 void formod (ctl_t * ctl, atm_t * atm, obs_t * obs)

Determine ray paths and compute radiative transfer.

Definition at line 3014 of file jurassic.c.

```
03017                {
03018
03019   int id, ir, *mask;
03020
03021   /* Allocate... */
03022   ALLOC(mask, int,
03023         ND * NR);
03024
03025   /* Save observation mask... */
03026   for (id = 0; id < ctl->nd; id++)
03027     for (ir = 0; ir < obs->nr; ir++)
03028       mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03029
03030   /* Hydrostatic equilibrium... */
03031   hydrostatic(ctl, atm);
03032
03033   /* Calculate pencil beams... */
03034   for (ir = 0; ir < obs->nr; ir++)
03035     formod_pencil(ctl, atm, obs, ir);
03036
03037   /* Apply field-of-view convolution... */
03038   formod_fov(ctl, obs);
03039
03040   /* Convert radiance to brightness temperature... */
03041   if (ctl->write_bbt)
03042     for (id = 0; id < ctl->nd; id++)
03043       for (ir = 0; ir < obs->nr; ir++)
03044         obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03045
03046   /* Apply observation mask... */
03047   for (id = 0; id < ctl->nd; id++)
03048     for (ir = 0; ir < obs->nr; ir++)
03049       if (mask[id * NR + ir])
03050         obs->rad[id][ir] = GSL_NAN;
03051
03052   /* Free... */
03053   free(mask);
03054 }
```

Here is the call graph for this function:



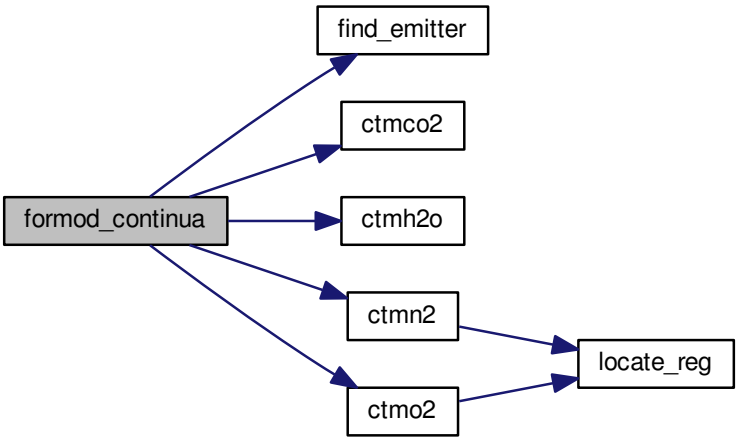
5.7.2.14 void formod_continua (ctl_t * *ctl*, los_t * *los*, int *ip*, double * *beta*)

Compute absorption coefficient of continua.

Definition at line 3058 of file jurassic.c.

```
03062                {
03063
03064   static int ig_co2 = -999, ig_h2o = -999;
03065
03066   int id;
03067
03068   /* Extinction... */
03069   for (id = 0; id < ctl->nd; id++)
03070     beta[id] = los->k[ctl->window[id]][ip];
03071
03072   /* CO2 continuum... */
03073   if (ctl->ctm_co2) {
03074     if (ig_co2 == -999)
03075       ig_co2 = find_emitter(ctl, "CO2");
03076     if (ig_co2 >= 0)
03077       for (id = 0; id < ctl->nd; id++)
03078         beta[id] += ctmco2(ctl->nu[id], los->p[ip], los->t[ip],
03079                            los->u[ig_co2][ip]) / los->ds[ip];
03080   }
03081
03082   /* H2O continuum... */
03083   if (ctl->ctm_h2o) {
03084     if (ig_h2o == -999)
03085       ig_h2o = find_emitter(ctl, "H2O");
03086     if (ig_h2o >= 0)
03087       for (id = 0; id < ctl->nd; id++)
03088         beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03089                            los->q[ig_h2o][ip],
03090                            los->u[ig_h2o][ip]) / los->ds[ip];
03091   }
03092
03093   /* N2 continuum... */
03094   if (ctl->ctm_n2)
03095     for (id = 0; id < ctl->nd; id++)
03096       beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03097
03098   /* O2 continuum... */
03099   if (ctl->ctm_o2)
03100     for (id = 0; id < ctl->nd; id++)
03101       beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03102 }
```

Here is the call graph for this function:



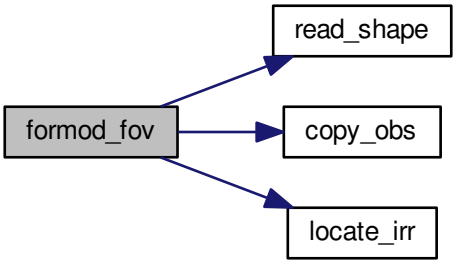
5.7.2.15 void formod_fov (ctl_t * *ctl*, obs_t * *obs*)

Apply field of view convolution.

Definition at line 3106 of file jurassic.c.

```
03108               {
03109
03110   static double dz[NSHAPE], w[NSHAPE];
03111
03112   static int init = 0, n;
03113
03114   obs_t *obs2;
03115
03116   double rad[ND][NR], tau[ND][NR], wsum, z[NR], zfov;
03117
03118   int i, id, idx, ir, ir2, nz;
03119
03120   /* Do not take into account FOV... */
03121   if (ctl->fov[0] == '-')
03122     return;
03123
03124   /* Initialize FOV data... */
03125   if (!init) {
03126     init = 1;
03127     read_shape(ctl->fov, dz, w, &n);
03128   }
03129
03130   /* Allocate... */
03131   ALLOC(obs2, obs_t, 1);
03132
03133   /* Copy observation data... */
03134   copy_obs(ctl, obs2, obs, 0);
03135
03136   /* Loop over ray paths... */
03137   for (ir = 0; ir < obs->nr; ir++) {
03138
03139     /* Get radiance and transmittance profiles... */
03140     nz = 0;
03141     for (ir2 = GSL_MAX(ir - NFOV, 0); ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr);
03142          ir2++)
03143       if (obs->time[ir2] == obs->time[ir]) {
03144         z[nz] = obs2->vpz[ir2];
03145         for (id = 0; id < ctl->nd; id++) {
03146           rad[id][nz] = obs2->rad[id][ir2];
03147           tau[id][nz] = obs2->tau[id][ir2];
03148         }
03149         nz++;
03150       }
03151     if (nz < 2)
03152       ERRMSG("Cannot apply FOV convolution!");
03153
03154     /* Convolute profiles with FOV... */
03155     wsum = 0;
03156     for (id = 0; id < ctl->nd; id++) {
03157       obs->rad[id][ir] = 0;
03158       obs->tau[id][ir] = 0;
03159     }
03160     for (i = 0; i < n; i++) {
03161       zfov = obs->vpz[ir] + dz[i];
03162       idx = locate_irr(z, nz, zfov);
03163       for (id = 0; id < ctl->nd; id++) {
03164         obs->rad[id][ir] += w[i]
03165           * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
03166         obs->tau[id][ir] += w[i]
03167           * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
03168       }
03169       wsum += w[i];
03170     }
03171     for (id = 0; id < ctl->nd; id++) {
03172       obs->rad[id][ir] /= wsum;
03173       obs->tau[id][ir] /= wsum;
03174     }
03175   }
03176
03177   /* Free... */
03178   free(obs2);
03179 }
```

Here is the call graph for this function:



5.7.2.16 void formod_pencil (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, int *ir*)

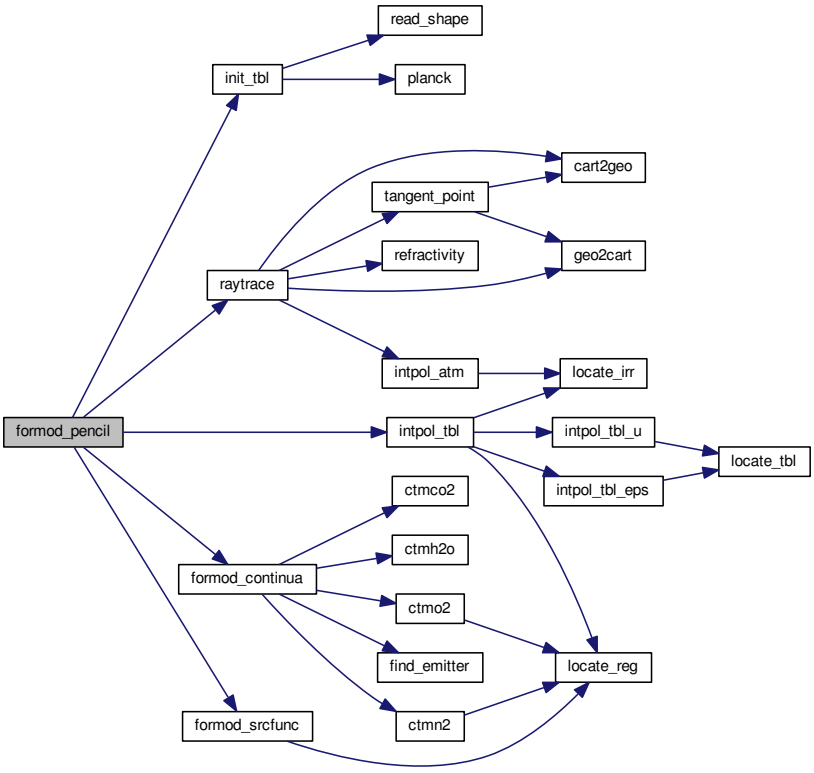
Compute radiative transfer for a pencil beam.

Definition at line 3183 of file jurassic.c.

```
03187           {
03188
03189     static tbl_t *tbl;
03190
03191     static int init = 0;
03192
03193     los_t *los;
03194
03195     double beta_ctm[ND], eps, src_planck[ND], tau_path[NG][ND], tau_gas[ND];
03196
03197     int id, ip;
03198
03199     /* Initialize look-up tables... */
03200     if (!init) {
03201       init = 1;
03202       ALLOC(tbl, tbl_t, 1);
03203       init_tbl(ctl, tbl);
03204     }
03205
03206     /* Allocate... */
03207     ALLOC(los, los_t, 1);
03208
03209     /* Initialize... */
03210     for (id = 0; id < ctl->nd; id++) {
03211       obs->rad[id][ir] = 0;
03212       obs->tau[id][ir] = 1;
03213     }
03214
03215     /* Raytracing... */
03216     raytrace(ctl, atm, obs, los, ir);
03217
03218     /* Loop over LOS points... */
03219     for (ip = 0; ip < los->np; ip++) {
03220
03221       /* Get trace gas transmittance... */
03222       intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03223
03224       /* Get continuum absorption... */
03225       formod_continua(ctl, los, ip, beta_ctm);
03226
03227       /* Compute Planck function... */
03228       formod_srcfunc(ctl, tbl, los->t[ip], src_planck);
03229
03230       /* Loop over channels... */
03231       for (id = 0; id < ctl->nd; id++)
03232         if (tau_gas[id] > 0) {
03233
```

```
03234         /* Get segment emissivity... */
03235         eps = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03236
03237         /* Compute radiance... */
03238         obs->rad[id][ir] += src_planck[id] * eps * obs->tau[id][ir];
03239
03240         /* Compute path transmittance... */
03241         obs->tau[id][ir] *= (1 - eps);
03242       }
03243   }
03244
03245   /* Add surface... */
03246   if (los->tsurf > 0) {
03247     formod_srcfunc(ctl, tbl, los->tsurf, src_planck);
03248     for (id = 0; id < ctl->nd; id++)
03249       obs->rad[id][ir] += src_planck[id] * obs->tau[id][ir];
03250   }
03251
03252   /* Free... */
03253   free(los);
03254 }
```

Here is the call graph for this function:



5.7.2.17 void formod_srcfunc (ctl_t * *ctl*, tbl_t * *tbl*, double *t*, double * *src*)

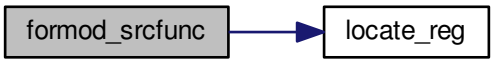
Compute Planck source function.

Definition at line 3258 of file jurassic.c.

```
03262                 {
03263
03264   int id, it;
```

```
03265
03266   /* Determine index in temperature array... */
03267   it = locate_reg(tbl->st, TBLNS, t);
03268
03269   /* Interpolate Planck function value... */
03270   for (id = 0; id < ctl->nd; id++)
03271     src[id] = LIN(tbl->st[it], tbl->sr[id][it],
03272                   tbl->st[it + 1], tbl->sr[id][it + 1], t);
03273 }
```

Here is the call graph for this function:



5.7.2.18 void geo2cart (double *z*, double *lon*, double *lat*, double * *x*)

Convert geolocation to Cartesian coordinates.

Definition at line 3277 of file jurassic.c.

```
03281             {
03282
03283   double radius;
03284
03285   radius = z + RE;
03286   x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);
03287   x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);
03288   x[2] = radius * sin(lat / 180 * M_PI);
03289 }
```

5.7.2.19 void hydrostatic (ctl_t * *ctl*, atm_t * *atm*)

Set hydrostatic equilibrium.

Definition at line 3293 of file jurassic.c.

```
03295               {
03296
03297   static int ig_h2o = -999;
03298
03299   double dzmin = 1e99, e = 0, mean, mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03300
03301   int i, ip, ipref = 0, ipts = 20;
03302
03303   /* Check reference height... */
03304   if (ctl->hydz < 0)
03305     return;
03306
03307   /* Determine emitter index of H2O... */
03308   if (ig_h2o == -999)
03309     ig_h2o = find_emitter(ctl, "H2O");
03310
03311   /* Find air parcel next to reference height... */
03312   for (ip = 0; ip < atm->np; ip++)
03313     if (fabs(atm->z[ip] - ctl->hydz) < dzmin) {
03314       dzmin = fabs(atm->z[ip] - ctl->hydz);
03315       ipref = ip;
03316     }
```

```
03317
03318   /* Upper part of profile... */
03319   for (ip = ipref + 1; ip < atm->np; ip++) {
03320     mean = 0;
03321     for (i = 0; i < ipts; i++) {
03322       if (ig_h2o >= 0)
03323         e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03324                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03325       mean += (e * mmh2o + (1 - e) * mmair)
03326         * G0 / RI
03327         / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03328     }
03329
03330     /* Compute p(z,T)... */
03331     atm->p[ip] =
03332       exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03333   }
03334
03335   /* Lower part of profile... */
03336   for (ip = ipref - 1; ip >= 0; ip--) {
03337     mean = 0;
03338     for (i = 0; i < ipts; i++) {
03339       if (ig_h2o >= 0)
03340         e = LIN(0.0, atm->q[ig_h2o][ip + 1],
03341                 ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03342       mean += (e * mmh2o + (1 - e) * mmair)
03343         * G0 / RI
03344         / LIN(0.0, atm->t[ip + 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03345     }
03346
03347     /* Compute p(z,T)... */
03348     atm->p[ip] =
03349       exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03350   }
03351 }
```

Here is the call graph for this function:

5.7.2.20 void idx2name (ctl_t * *ctl*, int *idx*, char * *quantity*)

Determine name of state vector quantity for given index.

Definition at line 3355 of file jurassic.c.

```
03358                   {
03359
03360   int ig, iw;
03361
03362   if (idx == IDXP)
03363     sprintf(quantity, "PRESSURE");
03364
03365   if (idx == IDXT)
03366     sprintf(quantity, "TEMPERATURE");
03367
03368   for (ig = 0; ig < ctl->ng; ig++)
03369     if (idx == IDXQ(ig))
03370       sprintf(quantity, "%s", ctl->emitter[ig]);
03371
03372   for (iw = 0; iw < ctl->nw; iw++)
03373     if (idx == IDXK(iw))
03374       sprintf(quantity, "EXTINCT_WINDOW%d", iw);
03375 }
```

5.7.2.21 void init_tbl (ctl_t * *ctl*, tbl_t * *tbl*)

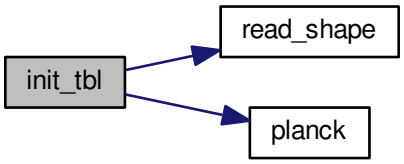
Initialize look-up tables.

Definition at line 3379 of file jurassic.c.

```
03381               {
03382
03383   FILE *in;
03384
03385   char filename[2 * LEN], line[LEN];
03386
03387   double eps, eps_old, press, press_old, temp, temp_old, u, u_old,
03388     f[NSHAPE], fsum, nu[NSHAPE];
03389
03390   int i, id, ig, ip, it, n;
03391
03392   /* Loop over trace gases and channels... */
03393   for (ig = 0; ig < ctl->ng; ig++)
03394 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(in,filename,line,eps,eps_old,press,
      press_old,temp,temp_old,u,u_old,id,ip,it)
03395     for (id = 0; id < ctl->nd; id++) {
03396
03397       /* Initialize... */
03398       tbl->np[ig][id] = -1;
03399       eps_old = -999;
03400       press_old = -999;
03401       temp_old = -999;
03402       u_old = -999;
03403
03404       /* Try to open file... */
03405       sprintf(filename, "%s_%.4f_%s.tab",
03406               ctl->tblbase, ctl->nu[id], ctl->emitter[ig]);
03407       if (!(in = fopen(filename, "r"))) {
03408         printf("Missing emissivity table: %s\n", filename);
03409         continue;
03410       }
03411       printf("Read emissivity table: %s\n", filename);
03412
03413       /* Read data... */
03414       while (fgets(line, LEN, in)) {
03415
03416         /* Parse line... */
03417         if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
03418           continue;
03419
03420         /* Determine pressure index... */
03421         if (press != press_old) {
03422           press_old = press;
03423           if ((++tbl->np[ig][id]) >= TBLNP)
03424             ERRMSG("Too many pressure levels!");
03425           tbl->nt[ig][id][tbl->np[ig][id]] = -1;
03426         }
03427
03428         /* Determine temperature index... */
03429         if (temp != temp_old) {
03430           temp_old = temp;
03431           if ((++tbl->nt[ig][id][tbl->np[ig][id]]) >= TBLNT)
03432             ERRMSG("Too many temperatures!");
03433           tbl->nu[ig][id][tbl->np[ig][id]]
03434             [tbl->nt[ig][id][tbl->np[ig][id]]] = -1;
03435         }
03436
03437         /* Determine column density index... */
03438         if ((eps > eps_old && u > u_old) || tbl->nu[ig][id][tbl->np[ig][id]]
03439             [tbl->nt[ig][id][tbl->np[ig][id]]] < 0) {
03440           eps_old = eps;
03441           u_old = u;
03442           if ((++tbl->nu[ig][id][tbl->np[ig][id]]
03443                [tbl->nt[ig][id][tbl->np[ig][id]]]) >= TBLNU) {
03444             tbl->nu[ig][id][tbl->np[ig][id]]
03445               [tbl->nt[ig][id][tbl->np[ig][id]]]--;
03446             continue;
03447           }
03448         }
03449
03450         /* Store data... */
03451         tbl->p[ig][id][tbl->np[ig][id]] = press;
03452         tbl->t[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03453           = temp;
03454         tbl->u[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03455           [tbl->nu[ig][id][tbl->np[ig][id]]
03456            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) u;
```

```
03457         tbl->eps[ig][id][tbl->np[ig][id]][tbl->nt[ig][id][tbl->np[ig][id]]]
03458           [tbl->nu[ig][id][tbl->np[ig][id]]
03459            [tbl->nt[ig][id][tbl->np[ig][id]]]] = (float) eps;
03460       }
03461
03462       /* Increment counters... */
03463       tbl->np[ig][id]++;
03464       for (ip = 0; ip < tbl->np[ig][id]; ip++) {
03465         tbl->nt[ig][id][ip]++;
03466         for (it = 0; it < tbl->nt[ig][id][ip]; it++)
03467           tbl->nu[ig][id][ip][it]++;
03468       }
03469
03470       /* Close file... */
03471       fclose(in);
03472     }
03473
03474   /* Write info... */
03475   printf("Initialize source function table...\n");
03476
03477   /* Loop over channels... */
03478 #pragma omp parallel for default(none) shared(ctl,tbl,ig) private(filename,it,i,n,f,fsum,nu)
03479   for (id = 0; id < ctl->nd; id++) {
03480
03481     /* Read filter function... */
03482     sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03483     read_shape(filename, nu, f, &n);
03484
03485     /* Compute source function table... */
03486     for (it = 0; it < TBLNS; it++) {
03487
03488       /* Set temperature... */
03489       tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03490
03491       /* Integrate Planck function... */
03492       fsum = 0;
03493       tbl->sr[id][it] = 0;
03494       for (i = 0; i < n; i++) {
03495         fsum += f[i];
03496         tbl->sr[id][it] += f[i] * planck(tbl->st[it], nu[i]);
03497       }
03498       tbl->sr[id][it] /= fsum;
03499     }
03500   }
03501 }
```

Here is the call graph for this function:



5.7.2.22 void intpol_atm (ctl_t * *ctl*, atm_t * *atm*, double *z*, double * *p*, double * *t*, double * *q*, double * *k*)

Interpolate atmospheric data.

Definition at line 3505 of file jurassic.c.

```
03512               {
03513
03514   int ig, ip, iw;
03515
```

```
03516   /* Get array index... */
03517   ip = locate_irr(atm->z, atm->np, z);
03518
03519   /* Interpolate... */
03520   *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);
03521   *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03522   for (ig = 0; ig < ctl->ng; ig++)
03523     q[ig] =
03524       LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03525   for (iw = 0; iw < ctl->nw; iw++)
03526     k[iw] =
03527       LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03528 }
```

Here is the call graph for this function:

5.7.2.23 void intpol_tbl (ctl_t * *ctl*, tbl_t * *tbl*, los_t * *los*, int *ip*, double *tau_path[NG][ND]*, double *tau_seg[ND]*)

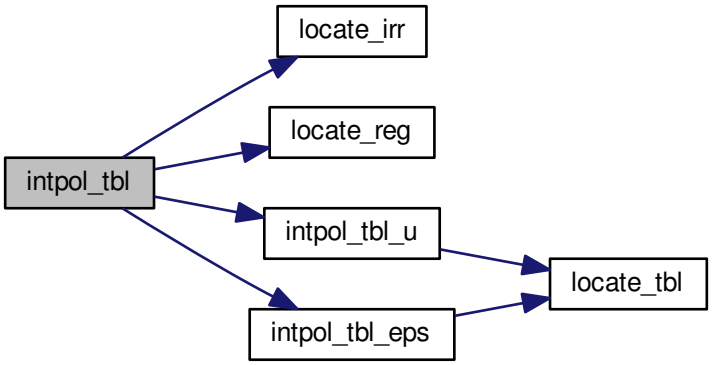
Get transmittance from look-up tables.

Definition at line 3532 of file jurassic.c.

```
03538                       {
03539
03540   double eps, eps00, eps01, eps10, eps11, u;
03541
03542   int id, ig, ipr, it0, it1;
03543
03544   /* Initialize... */
03545   if (ip <= 0)
03546     for (ig = 0; ig < ctl->ng; ig++)
03547       for (id = 0; id < ctl->nd; id++)
03548         tau_path[ig][id] = 1;
03549
03550   /* Loop over channels... */
03551   for (id = 0; id < ctl->nd; id++) {
03552
03553     /* Initialize... */
03554     tau_seg[id] = 1;
03555
03556     /* Loop over emitters.... */
03557     for (ig = 0; ig < ctl->ng; ig++) {
03558
03559       /* Check size of table (pressure)... */
03560       if (tbl->np[ig][id] < 2)
03561         eps = 0;
03562
03563       /* Check transmittance... */
03564       else if (tau_path[ig][id] < 1e-9)
03565         eps = 1;
03566
03567       /* Interpolate... */
03568       else {
03569
03570         /* Determine pressure and temperature indices... */
03571         ipr = locate_irr(tbl->p[ig][id], tbl->np[ig][id], los->p[ip]);
03572         it0 =
03573           locate_irr(tbl->t[ig][id][ipr], tbl->nt[ig][id][ipr], los->
    t[ip]);
03574         it1 =
03575           locate_reg(tbl->t[ig][id][ipr + 1], tbl->nt[ig][id][ipr + 1],
03576                      los->t[ip]);
```

```
03577
03578         /* Check size of table (temperature and column density)... */
03579         if (tbl->nt[ig][id][ipr] < 2 || tbl->nt[ig][id][ipr + 1] < 2
03580             || tbl->nu[ig][id][ipr][it0] < 2
03581             || tbl->nu[ig][id][ipr][it0 + 1] < 2
03582             || tbl->nu[ig][id][ipr + 1][it1] < 2
03583             || tbl->nu[ig][id][ipr + 1][it1 + 1] < 2)
03584           eps = 0;
03585
03586         else {
03587
03588           /* Get emissivities of extended path... */
03589           u = intpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[ig][id]);
03590           eps00 = intpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ig][ip]);
03591
03592           u = intpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[ig][id]);
03593           eps01 =
03594             intpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ig][ip]);
03595
03596           u = intpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[ig][id]);
03597           eps10 =
03598             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ig][ip]);
03599
03600           u =
03601             intpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[ig][id]);
03602           eps11 =
03603             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->
    u[ig][ip]);
03604
03605           /* Interpolate with respect to temperature... */
03606           eps00 = LIN(tbl->t[ig][id][ipr][it0], eps00,
03607                       tbl->t[ig][id][ipr][it0 + 1], eps01, los->t[ip]);
03608           eps11 = LIN(tbl->t[ig][id][ipr + 1][it1], eps10,
03609                       tbl->t[ig][id][ipr + 1][it1 + 1], eps11, los->t[ip]);
03610
03611           /* Interpolate with respect to pressure... */
03612           eps00 = LIN(tbl->p[ig][id][ipr], eps00,
03613                       tbl->p[ig][id][ipr + 1], eps11, los->p[ip]);
03614
03615           /* Check emssivity range... */
03616           eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03617
03618           /* Determine segment emissivity... */
03619           eps = 1 - (1 - eps00) / tau_path[ig][id];
03620         }
03621       }
03622
03623       /* Get transmittance of extended path... */
03624       tau_path[ig][id] *= (1 - eps);
03625
03626       /* Get segment transmittance... */
03627       tau_seg[id] *= (1 - eps);
03628     }
03629   }
03630 }
```

Here is the call graph for this function:



5.7.2.24 double intpol_tbl_eps (tbl_t * tbl, int ig, int id, int ip, int it, double u)

Interpolate emissivity from look-up tables.

Definition at line 3634 of file jurassic.c.

```
03640             {
03641
03642   int idx;
03643
03644   /* Lower boundary... */
03645   if (u < tbl->u[ig][id][ip][it][0])
03646     return LIN(0, 0, tbl->u[ig][id][ip][it][0], tbl->eps[ig][id][ip][it][0],
03647                u);
03648
03649   /* Upper boundary... */
03650   else if (u > tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
03651     return LIN(tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03652                tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03653                1e30, 1, u);
03654
03655   /* Interpolation... */
03656   else {
03657
03658     /* Get index... */
03659     idx = locate_tbl(tbl->u[ig][id][ip][it], tbl->nu[ig][id][ip][it], u);
03660
03661     /* Interpolate... */
03662     return
03663       LIN(tbl->u[ig][id][ip][it][idx], tbl->eps[ig][id][ip][it][idx],
03664           tbl->u[ig][id][ip][it][idx + 1], tbl->eps[ig][id][ip][it][idx + 1],
03665           u);
03666   }
03667 }
```

Here is the call graph for this function:

5.7.2.25 double intpol_tbl_u (tbl_t * tbl, int ig, int id, int ip, int it, double eps)

Interpolate column density from look-up tables.

Definition at line 3671 of file jurassic.c.

```
03677             {
03678
03679   int idx;
03680
03681   /* Lower boundary... */
03682   if (eps < tbl->eps[ig][id][ip][it][0])
03683     return LIN(0, 0, tbl->eps[ig][id][ip][it][0], tbl->u[ig][id][ip][it][0],
03684                eps);
03685
03686   /* Upper boundary... */
03687   else if (eps > tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1])
03688     return LIN(tbl->eps[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03689                tbl->u[ig][id][ip][it][tbl->nu[ig][id][ip][it] - 1],
03690                1, 1e30, eps);
03691
```

```
03692   /* Interpolation... */
03693   else {
03694
03695     /* Get index... */
03696     idx = locate_tbl(tbl->eps[ig][id][ip][it], tbl->nu[ig][id][ip][it], eps);
03697
03698     /* Interpolate... */
03699     return
03700       LIN(tbl->eps[ig][id][ip][it][idx], tbl->u[ig][id][ip][it][idx],
03701           tbl->eps[ig][id][ip][it][idx + 1], tbl->u[ig][id][ip][it][idx + 1],
03702           eps);
03703   }
03704 }
```

Here is the call graph for this function:

5.7.2.26 void jsec2time (double *jsec*, int * *year*, int * *mon*, int * *day*, int * *hour*, int * *min*, int * *sec*, double * *remain*)

Convert seconds to date.

Definition at line 3708 of file jurassic.c.

```
03716                   {
03717
03718   struct tm t0, *t1;
03719
03720   time_t jsec0;
03721
03722   t0.tm_year = 100;
03723   t0.tm_mon = 0;
03724   t0.tm_mday = 1;
03725   t0.tm_hour = 0;
03726   t0.tm_min = 0;
03727   t0.tm_sec = 0;
03728
03729   jsec0 = (time_t) jsec + timegm(&t0);
03730   t1 = gmtime(&jsec0);
03731
03732   *year = t1->tm_year + 1900;
03733   *mon = t1->tm_mon + 1;
03734   *day = t1->tm_mday;
03735   *hour = t1->tm_hour;
03736   *min = t1->tm_min;
03737   *sec = t1->tm_sec;
03738   *remain = jsec - floor(jsec);
03739 }
```

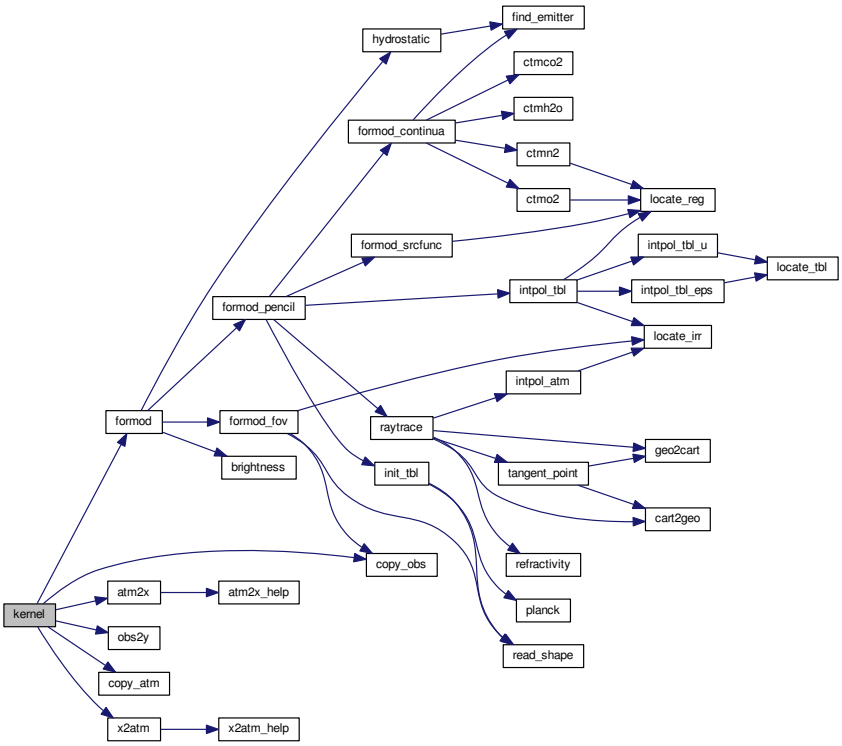
5.7.2.27 void kernel (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, gsl_matrix * *k*)

Compute Jacobians.

Definition at line 3743 of file jurassic.c.

```
03747                   {
03748
03749     atm_t *atm1;
03750     obs_t *obs1;
03751
03752     gsl_vector *x0, *x1, *yy0, *yy1;
03753
03754     int *iqa, j;
03755
03756     double h;
03757
03758     size_t i, n, m;
03759
03760     /* Get sizes... */
03761     m = k->size1;
03762     n = k->size2;
03763
03764     /* Allocate... */
03765     x0 = gsl_vector_alloc(n);
03766     yy0 = gsl_vector_alloc(m);
03767     ALLOC(iqa, int,
03768           N);
03769
03770     /* Compute radiance for undisturbed atmospheric data... */
03771     formod(ctl, atm, obs);
03772
03773     /* Compose vectors... */
03774     atm2x(ctl, atm, x0, iqa, NULL);
03775     obs2y(ctl, obs, yy0, NULL, NULL);
03776
03777     /* Initialize kernel matrix... */
03778     gsl_matrix_set_zero(k);
03779
03780     /* Loop over state vector elements... */
03781 #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(i, j, h, x1, yy1, atm1,
      obs1)
03782     for (j = 0; j < (int) n; j++) {
03783
03784       /* Allocate... */
03785       x1 = gsl_vector_alloc(n);
03786       yy1 = gsl_vector_alloc(m);
03787       ALLOC(atm1, atm_t, 1);
03788       ALLOC(obs1, obs_t, 1);
03789
03790       /* Set perturbation size... */
03791       if (iqa[j] == IDXP)
03792         h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-7);
03793       else if (iqa[j] == IDXT)
03794         h = 1;
03795       else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03796         h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, (size_t) j)), 1e-15);
03797       else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03798         h = 1e-4;
03799       else
03800         ERRMSG("Cannot set perturbation size!");
03801
03802       /* Disturb state vector element... */
03803       gsl_vector_memcpy(x1, x0);
03804       gsl_vector_set(x1, (size_t) j, gsl_vector_get(x1, (size_t) j) + h);
03805       copy_atm(ctl, atm1, atm, 0);
03806       copy_obs(ctl, obs1, obs, 0);
03807       x2atm(ctl, x1, atm1);
03808
03809       /* Compute radiance for disturbed atmospheric data... */
03810       formod(ctl, atm1, obs1);
03811
03812       /* Compose measurement vector for disturbed radiance data... */
03813       obs2y(ctl, obs1, yy1, NULL, NULL);
03814
03815       /* Compute derivatives... */
03816       for (i = 0; i < m; i++)
03817         gsl_matrix_set(k, i, (size_t) j,
03818                        (gsl_vector_get(yy1, i) - gsl_vector_get(yy0, i)) / h);
03819
03820       /* Free... */
03821       gsl_vector_free(x1);
03822       gsl_vector_free(yy1);
03823       free(atm1);
03824       free(obs1);
03825     }
03826
03827     /* Free... */
03828     gsl_vector_free(x0);
03829     gsl_vector_free(yy0);
03830     free(iqa);
03831   }
```

Here is the call graph for this function:



5.7.2.28 int locate_irr (double * *xx*, int *n*, double *x*)

Find array index for irregular grid.

Definition at line 3835 of file jurassic.c.

```
03838             {
03839
03840   int i, ilo, ihi;
03841
03842   ilo = 0;
03843   ihi = n - 1;
03844   i = (ihi + ilo) >> 1;
03845
03846   if (xx[i] < xx[i + 1])
03847     while (ihi > ilo + 1) {
03848       i = (ihi + ilo) >> 1;
03849       if (xx[i] > x)
03850         ihi = i;
03851       else
03852         ilo = i;
03853   } else
03854     while (ihi > ilo + 1) {
03855       i = (ihi + ilo) >> 1;
03856       if (xx[i] <= x)
03857         ihi = i;
03858       else
03859         ilo = i;
03860     }
03861
03862   return ilo;
03863 }
```

5.7.2.29 int locate_reg (double * xx, int n, double x)

Find array index for regular grid.

Definition at line 3867 of file jurassic.c.

```
03870             {
03871
03872   int i;
03873
03874   /* Calculate index... */
03875   i = (int) ((x - xx[0]) / (xx[1] - xx[0]));
03876
03877   /* Check range... */
03878   if (i < 0)
03879     i = 0;
03880   else if (i >= n - 2)
03881     i = n - 2;
03882
03883   return i;
03884 }
```

5.7.2.30 int locate_tbl (float * xx, int n, double x)

Find array index in float array.

Definition at line 3888 of file jurassic.c.

```
03891             {
03892
03893   int i, ilo, ihi;
03894
03895   ilo = 0;
03896   ihi = n - 1;
03897   i = (ihi + ilo) >> 1;
03898
03899   while (ihi > ilo + 1) {
03900     i = (ihi + ilo) >> 1;
03901     if (xx[i] > x)
03902       ihi = i;
03903     else
03904       ilo = i;
03905   }
03906
03907   return ilo;
03908 }
```

5.7.2.31 size_t obs2y (ctl_t * ctl, obs_t * obs, gsl_vector * y, int * ida, int * ira)

Compose measurement vector.

Definition at line 3912 of file jurassic.c.

```
03917             {
03918
03919   int id, ir;
03920
03921   size_t m = 0;
03922
03923   /* Determine measurement vector... */
03924   for (ir = 0; ir < obs->nr; ir++)
03925     for (id = 0; id < ctl->nd; id++)
03926       if (gsl_finite(obs->rad[id][ir])) {
03927         if (y != NULL)
03928           gsl_vector_set(y, m, obs->rad[id][ir]);
03929         if (ida != NULL)
03930           ida[m] = id;
03931         if (ira != NULL)
03932           ira[m] = ir;
03933         m++;
03934       }
03935
03936   return m;
03937 }
```

5.7.2.32 double planck (double *t*, double *nu*)

Compute Planck function.

Definition at line 3941 of file jurassic.c.

```
03943             {
03944
03945   return C1 * POW3(nu) / gsl_expm1(C2 * nu / t);
03946 }
```

5.7.2.33 void raytrace (ctl_t * *ctl*, atm_t * *atm*, obs_t * *obs*, los_t * *los*, int *ir*)

Do ray-tracing to determine LOS.

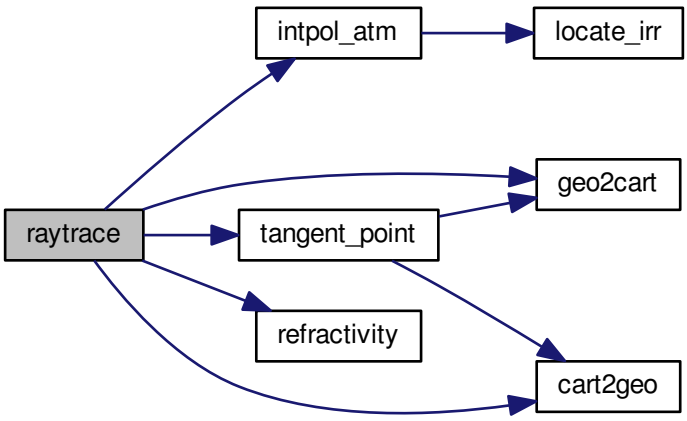
Definition at line 3950 of file jurassic.c.

```
03955           {
03956
03957   double cosa, d, dmax, dmin = 0, ds, ex0[3], ex1[3], frac, h = 0.02, k[NW],
03958     lat, lon, n, naux, ng[3], norm, p, q[NG], t, x[3], xh[3],
03959     xobs[3], xvp[3], z = 1e99, zmax, zmin, zrefrac = 60;
03960
03961   int i, ig, ip, iw, stop = 0;
03962
03963   /* Initialize... */
03964   los->np = 0;
03965   los->tsurf = -999;
03966   obs->tpz[ir] = obs->vpz[ir];
03967   obs->tplon[ir] = obs->vplon[ir];
03968   obs->tplat[ir] = obs->vplat[ir];
03969
03970   /* Get altitude range of atmospheric data... */
03971   gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03972
03973   /* Check observer altitude... */
03974   if (obs->obsz[ir] < zmin)
03975     ERRMSG("Observer below surface!");
03976
03977   /* Check view point altitude... */
03978   if (obs->vpz[ir] > zmax)
03979     return;
03980
03981   /* Determine Cartesian coordinates for observer and view point... */
03982   geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
03983   geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
03984
03985   /* Determine initial tangent vector... */
03986   for (i = 0; i < 3; i++)
03987     ex0[i] = xvp[i] - xobs[i];
03988   norm = NORM(ex0);
03989   for (i = 0; i < 3; i++)
03990     ex0[i] /= norm;
03991
03992   /* Observer within atmosphere... */
03993   for (i = 0; i < 3; i++)
03994     x[i] = xobs[i];
03995
03996   /* Observer above atmosphere (search entry point)... */
03997   if (obs->obsz[ir] > zmax) {
03998     dmax = norm;
03999     while (fabs(dmin - dmax) > 0.001) {
04000       d = (dmax + dmin) / 2;
04001       for (i = 0; i < 3; i++)
04002         x[i] = xobs[i] + d * ex0[i];
04003       cart2geo(x, &z, &lon, &lat);
04004       if (z <= zmax && z > zmax - 0.001)
04005         break;
04006       if (z < zmax - 0.0005)
04007         dmax = d;
04008       else
04009         dmin = d;
04010     }
04011   }
04012
04013   /* Ray-tracing... */
```

```
04014   while (1) {
04015
04016     /* Set step length... */
04017     ds = ctl->rayds;
04018     if (ctl->raydz > 0) {
04019       norm = NORM(x);
04020       for (i = 0; i < 3; i++)
04021         xh[i] = x[i] / norm;
04022       cosa = fabs(DOTP(ex0, xh));
04023       if (cosa != 0)
04024         ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04025     }
04026
04027     /* Determine geolocation... */
04028     cart2geo(x, &z, &lon, &lat);
04029
04030     /* Check if LOS hits the ground or has left atmosphere... */
04031     if (z < zmin || z > zmax) {
04032       stop = (z < zmin ? 2 : 1);
04033       frac =
04034         ((z <
04035           zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04036                                                                    1]);
04037       geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04038                los->lat[los->np - 1], xh);
04039       for (i = 0; i < 3; i++)
04040         x[i] = xh[i] + frac * (x[i] - xh[i]);
04041       cart2geo(x, &z, &lon, &lat);
04042       los->ds[los->np - 1] = ds * frac;
04043       ds = 0;
04044     }
04045
04046     /* Interpolate atmospheric data... */
04047     intpol_atm(ctl, atm, z, &p, &t, q, k);
04048
04049     /* Save data... */
04050     los->lon[los->np] = lon;
04051     los->lat[los->np] = lat;
04052     los->z[los->np] = z;
04053     los->p[los->np] = p;
04054     los->t[los->np] = t;
04055     for (ig = 0; ig < ctl->ng; ig++)
04056       los->q[ig][los->np] = q[ig];
04057     for (iw = 0; iw < ctl->nw; iw++)
04058       los->k[iw][los->np] = k[iw];
04059     los->ds[los->np] = ds;
04060
04061     /* Increment and check number of LOS points... */
04062     if ((++los->np) > NLOS)
04063       ERRMSG("Too many LOS points!");
04064
04065     /* Check stop flag... */
04066     if (stop) {
04067       los->tsurf = (stop == 2 ? t : -999);
04068       break;
04069     }
04070
04071     /* Determine refractivity... */
04072     if (ctl->refrac && z <= zrefrac)
04073       n = 1 + refractivity(p, t);
04074     else
04075       n = 1;
04076
04077     /* Construct new tangent vector (first term)... */
04078     for (i = 0; i < 3; i++)
04079       ex1[i] = ex0[i] * n;
04080
04081     /* Compute gradient of refractivity... */
04082     if (ctl->refrac && z <= zrefrac) {
04083       for (i = 0; i < 3; i++)
04084         xh[i] = x[i] + 0.5 * ds * ex0[i];
04085       cart2geo(xh, &z, &lon, &lat);
04086       intpol_atm(ctl, atm, z, &p, &t, q, k);
04087       n = refractivity(p, t);
04088       for (i = 0; i < 3; i++) {
04089         xh[i] += h;
04090         cart2geo(xh, &z, &lon, &lat);
04091         intpol_atm(ctl, atm, z, &p, &t, q, k);
04092         naux = refractivity(p, t);
04093         ng[i] = (naux - n) / h;
04094         xh[i] -= h;
04095       }
04096     } else
04097       for (i = 0; i < 3; i++)
04098         ng[i] = 0;
04099
04100     /* Construct new tangent vector (second term)... */
```

```
04101     for (i = 0; i < 3; i++)
04102       ex1[i] += ds * ng[i];
04103
04104     /* Normalize new tangent vector... */
04105     norm = NORM(ex1);
04106     for (i = 0; i < 3; i++)
04107       ex1[i] /= norm;
04108
04109     /* Determine next point of LOS... */
04110     for (i = 0; i < 3; i++)
04111       x[i] += 0.5 * ds * (ex0[i] + ex1[i]);
04112
04113     /* Copy tangent vector... */
04114     for (i = 0; i < 3; i++)
04115       ex0[i] = ex1[i];
04116   }
04117
04118   /* Get tangent point (to be done before changing segment lengths!)... */
04119   tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->
      tplat[ir]);
04120
04121   /* Change segment lengths according to trapezoid rule... */
04122   for (ip = los->np - 1; ip >= 1; ip--)
04123     los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04124   los->ds[0] *= 0.5;
04125
04126   /* Compute column density... */
04127   for (ip = 0; ip < los->np; ip++)
04128     for (ig = 0; ig < ctl->ng; ig++)
04129       los->u[ig][ip] = 10 * los->q[ig][ip] * los->p[ip]
04130         / (KB * los->t[ip]) * los->ds[ip];
04131 }
```

Here is the call graph for this function:



5.7.2.34 void read_atm (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, atm_t * *atm*)

Read atmospheric data.

Definition at line 4135 of file jurassic.c.

```
04139               {
04140
04141   FILE *in;
04142
04143   char file[LEN], line[LEN], *tok;
```

```
04144
04145   int ig, iw;
04146
04147   /* Init... */
04148   atm->np = 0;
04149
04150   /* Set filename... */
04151   if (dirname != NULL)
04152     sprintf(file, "%s/%s", dirname, filename);
04153   else
04154     sprintf(file, "%s", filename);
04155
04156   /* Write info... */
04157   printf("Read atmospheric data: %s\n", file);
04158
04159   /* Open file... */
04160   if (!(in = fopen(file, "r")))
04161     ERRMSG("Cannot open file!");
04162
04163   /* Read line... */
04164   while (fgets(line, LEN, in)) {
04165
04166     /* Read data... */
04167     TOK(line, tok, "%lg", atm->time[atm->np]);
04168     TOK(NULL, tok, "%lg", atm->z[atm->np]);
04169     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04170     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04171     TOK(NULL, tok, "%lg", atm->p[atm->np]);
04172     TOK(NULL, tok, "%lg", atm->t[atm->np]);
04173     for (ig = 0; ig < ctl->ng; ig++)
04174       TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04175     for (iw = 0; iw < ctl->nw; iw++)
04176       TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04177
04178     /* Increment data point counter... */
04179     if ((++atm->np) > NP)
04180       ERRMSG("Too many data points!");
04181   }
04182
04183   /* Close file... */
04184   fclose(in);
04185
04186   /* Check number of points... */
04187   if (atm->np < 1)
04188     ERRMSG("Could not read any data!");
04189 }
```

5.7.2.35 void read_ctl (int *argc*, char * *argv[]*, ctl_t * *ctl*)

Read forward model control parameters.

Definition at line 4193 of file jurassic.c.

```
04196                {
04197
04198   int id, ig, iw;
04199
04200   /* Write info... */
04201   printf("\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04202          "(executable: %s | compiled: %s, %s)\n\n",
04203          argv[0], __DATE__, __TIME__);
04204
04205   /* Emitters... */
04206   ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04207   if (ctl->ng < 0 || ctl->ng > NG)
04208     ERRMSG("Set 0 <= NG <= MAX!");
04209   for (ig = 0; ig < ctl->ng; ig++)
04210     scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04211
04212   /* Radiance channels... */
04213   ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04214   if (ctl->nd < 0 || ctl->nd > ND)
04215     ERRMSG("Set 0 <= ND <= MAX!");
04216   for (id = 0; id < ctl->nd; id++)
04217     ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04218
04219   /* Spectral windows... */
04220   ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04221   if (ctl->nw < 0 || ctl->nw > NW)
04222     ERRMSG("Set 0 <= NW <= MAX!");
```

```
04223   for (id = 0; id < ctl->nd; id++)
04224     ctl->window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04225
04226   /* Emissivity look-up tables... */
04227   scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04228
04229   /* Hydrostatic equilibrium... */
04230   ctl->hydz = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04231
04232   /* Continua... */
04233   ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04234   ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04235   ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04236   ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04237
04238   /* Ray-tracing... */
04239   ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04240   ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04241   ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.5", NULL);
04242
04243   /* Field of view... */
04244   scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04245
04246   /* Retrieval interface... */
04247   ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04248   ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04249   ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04250   ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04251   for (ig = 0; ig < ctl->ng; ig++) {
04252     ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETQ_ZMIN", ig, "-999", NULL);
04253     ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETQ_ZMAX", ig, "-999", NULL);
04254   }
04255   for (iw = 0; iw < ctl->nw; iw++) {
04256     ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04257     ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04258   }
04259
04260   /* Output flags... */
04261   ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04262   ctl->write_matrix =
04263     (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04264 }
```

Here is the call graph for this function:

5.7.2.36 void read_matrix (const char * *dirname*, const char * *filename*, gsl_matrix * *matrix*)

Read matrix.

Definition at line 4268 of file jurassic.c.

```
04271                       {
04272
04273   FILE *in;
04274
04275   char dum[LEN], file[LEN], line[LEN];
04276
04277   double value;
04278
04279   int i, j;
04280
04281   /* Set filename... */
```

```
04282   if (dirname != NULL)
04283     sprintf(file, "%s/%s", dirname, filename);
04284   else
04285     sprintf(file, "%s", filename);
04286
04287   /* Write info... */
04288   printf("Read matrix: %s\n", file);
04289
04290   /* Open file... */
04291   if (!(in = fopen(file, "r")))
04292     ERRMSG("Cannot open file!");
04293
04294   /* Read data... */
04295   gsl_matrix_set_zero(matrix);
04296   while (fgets(line, LEN, in))
04297     if (sscanf(line, "%d %s %s %s %s %s %d %s %s %s %s %s %lg",
04298                &i, dum, dum, dum, dum, dum,
04299                &j, dum, dum, dum, dum, dum, &value) == 13)
04300       gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04301
04302   /* Close file... */
04303   fclose(in);
04304 }
```

5.7.2.37 void read_obs (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, obs_t * *obs*)

Read observation data.

Definition at line 4308 of file jurassic.c.

```
04312                {
04313
04314   FILE *in;
04315
04316   char file[LEN], line[LEN], *tok;
04317
04318   int id;
04319
04320   /* Init... */
04321   obs->nr = 0;
04322
04323   /* Set filename... */
04324   if (dirname != NULL)
04325     sprintf(file, "%s/%s", dirname, filename);
04326   else
04327     sprintf(file, "%s", filename);
04328
04329   /* Write info... */
04330   printf("Read observation data: %s\n", file);
04331
04332   /* Open file... */
04333   if (!(in = fopen(file, "r")))
04334     ERRMSG("Cannot open file!");
04335
04336   /* Read line... */
04337   while (fgets(line, LEN, in)) {
04338
04339     /* Read data... */
04340     TOK(line, tok, "%lg", obs->time[obs->nr]);
04341     TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04342     TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04343     TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04344     TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04345     TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04346     TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04347     TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04348     TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04349     TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04350     for (id = 0; id < ctl->nd; id++)
04351       TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04352     for (id = 0; id < ctl->nd; id++)
04353       TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04354
04355     /* Increment counter... */
04356     if ((++obs->nr) > NR)
04357       ERRMSG("Too many rays!");
04358   }
04359
04360   /* Close file... */
04361   fclose(in);
```

```
04362
04363   /* Check number of points... */
04364   if (obs->nr < 1)
04365     ERRMSG("Could not read any data!");
04366 }
```

5.7.2.38 void read_shape (const char * *filename*, double * *x*, double * *y*, int * *n*)

Read shape function.

Definition at line 4370 of file jurassic.c.

```
04374           {
04375
04376   FILE *in;
04377
04378   char line[LEN];
04379
04380   /* Write info... */
04381   printf("Read shape function: %s\n", filename);
04382
04383   /* Open file... */
04384   if (!(in = fopen(filename, "r")))
04385     ERRMSG("Cannot open file!");
04386
04387   /* Read data... */
04388   *n = 0;
04389   while (fgets(line, LEN, in))
04390     if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04391       if ((++(*n)) > NSHAPE)
04392         ERRMSG("Too many data points!");
04393
04394   /* Check number of points... */
04395   if (*n < 1)
04396     ERRMSG("Could not read any data!");
04397
04398   /* Close file... */
04399   fclose(in);
04400 }
```

5.7.2.39 double refractivity (double *p*, double *t*)

Compute refractivity (return value is n - 1).

Definition at line 4404 of file jurassic.c.

```
04406             {
04407
04408   /* Refractivity of air at 4 to 15 micron... */
04409   return 7.753e-05 * p / t;
04410 }
```

5.7.2.40 double scan_ctl (int *argc*, char * *argv[]*, const char * *varname*, int *arridx*, const char * *defvalue*, char * *value*)

Search control parameter file for variable entry.

Definition at line 4414 of file jurassic.c.

```
04420               {
04421
04422   FILE *in = NULL;
04423
04424   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04425     msg[2 * LEN], rvarname[LEN], rval[LEN];
04426
04427   int contain = 0, i;
04428
04429   /* Open file... */
04430   if (argv[1][0] != '-')
04431     if (!(in = fopen(argv[1], "r")))
04432       ERRMSG("Cannot open file!");
04433
04434   /* Set full variable name... */
04435   if (arridx >= 0) {
04436     sprintf(fullname1, "%s[%d]", varname, arridx);
04437     sprintf(fullname2, "%s[*]", varname);
04438   } else {
04439     sprintf(fullname1, "%s", varname);
04440     sprintf(fullname2, "%s", varname);
04441   }
04442
04443   /* Read data... */
04444   if (in != NULL)
04445     while (fgets(line, LEN, in))
04446       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04447         if (strcasecmp(rvarname, fullname1) == 0 ||
04448             strcasecmp(rvarname, fullname2) == 0) {
04449           contain = 1;
04450           break;
04451         }
04452   for (i = 1; i < argc - 1; i++)
04453     if (strcasecmp(argv[i], fullname1) == 0 ||
04454         strcasecmp(argv[i], fullname2) == 0) {
04455       sprintf(rval, "%s", argv[i + 1]);
04456       contain = 1;
04457       break;
04458     }
04459
04460   /* Close file... */
04461   if (in != NULL)
04462     fclose(in);
04463
04464   /* Check for missing variables... */
04465   if (!contain) {
04466     if (strlen(defvalue) > 0)
04467       sprintf(rval, "%s", defvalue);
04468     else {
04469       sprintf(msg, "Missing variable %s!\n", fullname1);
04470       ERRMSG(msg);
04471     }
04472   }
04473
04474   /* Write info... */
04475   printf("%s = %s\n", fullname1, rval);
04476
04477   /* Return values... */
04478   if (value != NULL)
04479     sprintf(value, "%s", rval);
04480   return atof(rval);
04481 }
```

5.7.2.41 void tangent_point (los_t * *los*, double * *tpz*, double * *tplon*, double * *tplat*)

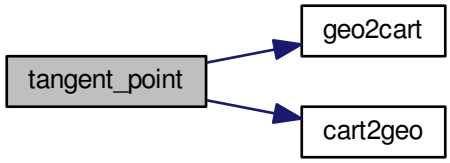
Find tangent point of a given LOS.

Definition at line 4485 of file jurassic.c.

```
04489               {
04490
04491   double a, b, c, dummy, v[3], v0[3], v2[3], x, x1, x2, yy0, yy1, yy2;
04492
04493   size_t i, ip;
04494
04495   /* Find minimum altitude... */
04496   ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
04497
04498   /* Nadir or zenith... */
04499   if (ip <= 0 || ip >= (size_t) los->np - 1) {
```

```
04500     *tpz = los->z[los->np - 1];
04501     *tplon = los->lon[los->np - 1];
04502     *tplat = los->lat[los->np - 1];
04503   }
04504
04505   /* Limb... */
04506   else {
04507
04508     /* Determine interpolating polynomial y=a*x^2+b*x+c... */
04509     yy0 = los->z[ip - 1];
04510     yy1 = los->z[ip];
04511     yy2 = los->z[ip + 1];
04512     x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
04513     x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
04514     a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);
04515     b = -(yy0 - yy1) / x1 - a * x1;
04516     c = yy0;
04517
04518     /* Get tangent point location... */
04519     x = -b / (2 * a);
04520     *tpz = a * x * x + b * x + c;
04521     geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
04522     geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);
04523     for (i = 0; i < 3; i++)
04524       v[i] = LIN(0.0, v0[i], x2, v2[i], x);
04525     cart2geo(v, &dummy, tplon, tplat);
04526   }
04527 }
```

Here is the call graph for this function:



5.7.2.42 void time2jsec (int *year*, int *mon*, int *day*, int *hour*, int *min*, int *sec*, double *remain*, double * *jsec*)

Convert date to seconds.

Definition at line 4531 of file jurassic.c.

```
04539                 {
04540
04541   struct tm t0, t1;
04542
04543   t0.tm_year = 100;
04544   t0.tm_mon = 0;
04545   t0.tm_mday = 1;
04546   t0.tm_hour = 0;
04547   t0.tm_min = 0;
04548   t0.tm_sec = 0;
04549
04550   t1.tm_year = year - 1900;
04551   t1.tm_mon = mon - 1;
04552   t1.tm_mday = day;
04553   t1.tm_hour = hour;
04554   t1.tm_min = min;
04555   t1.tm_sec = sec;
04556
04557   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
04558 }
```

5.7.2.43 void timer (const char * *name*, const char * *file*, const char * *func*, int *line*, int *mode*)

Measure wall-clock time.

Definition at line 4562 of file jurassic.c.

```
04567            {
04568
04569   static double w0[10];
04570
04571   static int l0[10], nt;
04572
04573   /* Start new timer... */
04574   if (mode == 1) {
04575     w0[nt] = omp_get_wtime();
04576     l0[nt] = line;
04577     if ((++nt) >= 10)
04578       ERRMSG("Too many timers!");
04579   }
04580
04581   /* Write elapsed time... */
04582   else {
04583
04584     /* Check timer index... */
04585     if (nt - 1 < 0)
04586       ERRMSG("Coding error!");
04587
04588     /* Write elapsed time... */
04589     printf("Timer '%s' (%s, %s, l%d-%d): %.3f sec\n",
04590            name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
04591   }
04592
04593   /* Stop timer... */
04594   if (mode == 3)
04595     nt--;
04596 }
```

5.7.2.44 void write_atm (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, atm_t * *atm*)

Write atmospheric data.

Definition at line 4600 of file jurassic.c.

```
04604               {
04605
04606   FILE *out;
04607
04608   char file[LEN];
04609
04610   int ig, ip, iw, n = 6;
04611
04612   /* Set filename... */
04613   if (dirname != NULL)
04614     sprintf(file, "%s/%s", dirname, filename);
04615   else
04616     sprintf(file, "%s", filename);
04617
04618   /* Write info... */
04619   printf("Write atmospheric data: %s\n", file);
04620
04621   /* Create file... */
04622   if (!(out = fopen(file, "w")))
04623     ERRMSG("Cannot create file!");
04624
04625   /* Write header... */
04626   fprintf(out,
04627           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04628           "# $2 = altitude [km]\n"
04629           "# $3 = longitude [deg]\n"
04630           "# $4 = latitude [deg]\n"
04631           "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
04632   for (ig = 0; ig < ctl->ng; ig++)
04633     fprintf(out, "# $%d = %s volume mixing ratio\n", ++n, ctl->emitter[ig]);
04634   for (iw = 0; iw < ctl->nw; iw++)
04635     fprintf(out, "# $%d = window %d: extinction [1/km]\n", ++n, iw);
04636
```

```
04637   /* Write data... */
04638   for (ip = 0; ip < atm->np; ip++) {
04639     if (ip == 0 || atm->lat[ip] != atm->lat[ip - 1]
04640         || atm->lon[ip] != atm->lon[ip - 1])
04641       fprintf(out, "\n");
04642     fprintf(out, "%.2f %g %g %g %g %g", atm->time[ip], atm->z[ip],
04643             atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
04644     for (ig = 0; ig < ctl->ng; ig++)
04645       fprintf(out, " %g", atm->q[ig][ip]);
04646     for (iw = 0; iw < ctl->nw; iw++)
04647       fprintf(out, " %g", atm->k[iw][ip]);
04648     fprintf(out, "\n");
04649   }
04650
04651   /* Close file... */
04652   fclose(out);
04653 }
```

5.7.2.45 void write_matrix (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, gsl_matrix * *matrix*, atm_t * *atm*, obs_t * *obs*, const char * *rowspace*, const char * *colspace*, const char * *sort*)

Write matrix.

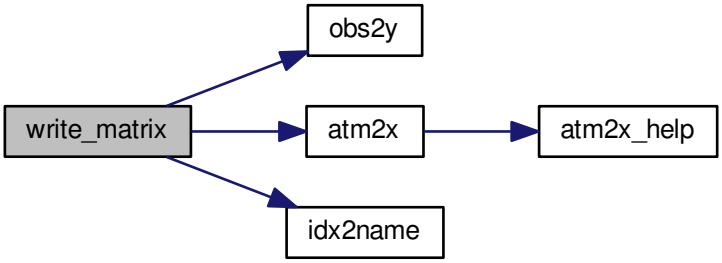
Definition at line 4657 of file jurassic.c.

```
04666                   {
04667
04668   FILE *out;
04669
04670   char file[LEN], quantity[LEN];
04671
04672   int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
04673
04674   size_t i, j, nc, nr;
04675
04676   /* Check output flag... */
04677   if (!ctl->write_matrix)
04678     return;
04679
04680   /* Allocate... */
04681   ALLOC(cida, int, M);
04682   ALLOC(ciqa, int,
04683         N);
04684   ALLOC(cipa, int,
04685         N);
04686   ALLOC(cira, int,
04687         M);
04688   ALLOC(rida, int,
04689         M);
04690   ALLOC(riqa, int,
04691         N);
04692   ALLOC(ripa, int,
04693         N);
04694   ALLOC(rira, int,
04695         M);
04696
04697   /* Set filename... */
04698   if (dirname != NULL)
04699     sprintf(file, "%s/%s", dirname, filename);
04700   else
04701     sprintf(file, "%s", filename);
04702
04703   /* Write info... */
04704   printf("Write matrix: %s\n", file);
04705
04706   /* Create file... */
04707   if (!(out = fopen(file, "w")))
04708     ERRMSG("Cannot create file!");
04709
04710   /* Write header (row space)... */
04711   if (rowspace[0] == 'y') {
04712
04713     fprintf(out,
04714             "# $1 = Row: index (measurement space)\n"
04715             "# $2 = Row: channel wavenumber [cm^-1]\n"
04716             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
04717             "# $4 = Row: view point altitude [km]\n"
04718             "# $5 = Row: view point longitude [deg]\n"
04719             "# $6 = Row: view point latitude [deg]\n");
04720
```

```
04721     /* Get number of rows... */
04722     nr = obs2y(ctl, obs, NULL, rida, rira);
04723
04724   } else {
04725
04726     fprintf(out,
04727             "# $1 = Row: index (state space)\n"
04728             "# $2 = Row: name of quantity\n"
04729             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
04730             "# $4 = Row: altitude [km]\n"
04731             "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
04732
04733     /* Get number of rows... */
04734     nr = atm2x(ctl, atm, NULL, riqa, ripa);
04735   }
04736
04737   /* Write header (column space)... */
04738   if (colspace[0] == 'y') {
04739
04740     fprintf(out,
04741             "# $7 = Col: index (measurement space)\n"
04742             "# $8 = Col: channel wavenumber [cm^-1]\n"
04743             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04744             "# $10 = Col: view point altitude [km]\n"
04745             "# $11 = Col: view point longitude [deg]\n"
04746             "# $12 = Col: view point latitude [deg]\n");
04747
04748     /* Get number of columns... */
04749     nc = obs2y(ctl, obs, NULL, cida, cira);
04750
04751   } else {
04752
04753     fprintf(out,
04754             "# $7 = Col: index (state space)\n"
04755             "# $8 = Col: name of quantity\n"
04756             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
04757             "# $10 = Col: altitude [km]\n"
04758             "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
04759
04760     /* Get number of columns... */
04761     nc = atm2x(ctl, atm, NULL, ciqa, cipa);
04762   }
04763
04764   /* Write header entry... */
04765   fprintf(out, "# $13 = Matrix element\n\n");
04766
04767   /* Write matrix data... */
04768   i = j = 0;
04769   while (i < nr && j < nc) {
04770
04771     /* Write info about the row... */
04772     if (rowspace[0] == 'y')
04773       fprintf(out, "%d %g %.2f %g %g %g",
04774               (int) i, ctl->nu[rida[i]],
04775               obs->time[rira[i]], obs->vpz[rira[i]],
04776               obs->vplon[rira[i]], obs->vplat[rira[i]]);
04777     else {
04778       idx2name(ctl, riqa[i], quantity);
04779       fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
04780               atm->time[ripa[i]], atm->z[ripa[i]],
04781               atm->lon[ripa[i]], atm->lat[ripa[i]]);
04782     }
04783
04784     /* Write info about the column... */
04785     if (colspace[0] == 'y')
04786       fprintf(out, " %d %g %.2f %g %g %g",
04787               (int) j, ctl->nu[cida[j]],
04788               obs->time[cira[j]], obs->vpz[cira[j]],
04789               obs->vplon[cira[j]], obs->vplat[cira[j]]);
04790     else {
04791       idx2name(ctl, ciqa[j], quantity);
04792       fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
04793               atm->time[cipa[j]], atm->z[cipa[j]],
04794               atm->lon[cipa[j]], atm->lat[cipa[j]]);
04795     }
04796
04797     /* Write matrix entry... */
04798     fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
04799
04800     /* Set matrix indices... */
04801     if (sort[0] == 'r') {
04802       j++;
04803       if (j >= nc) {
04804         j = 0;
04805         i++;
04806         fprintf(out, "\n");
04807       }
```

```
04808     } else {
04809       i++;
04810       if (i >= nr) {
04811         i = 0;
04812         j++;
04813         fprintf(out, "\n");
04814       }
04815     }
04816   }
04817
04818   /* Close file... */
04819   fclose(out);
04820
04821   /* Free... */
04822   free(cida);
04823   free(ciqa);
04824   free(cipa);
04825   free(cira);
04826   free(rida);
04827   free(riqa);
04828   free(ripa);
04829   free(rira);
04830 }
```

Here is the call graph for this function:



5.7.2.46 void write_obs (const char * *dirname*, const char * *filename*, ctl_t * *ctl*, obs_t * *obs*)

Write observation data.

Definition at line 4834 of file jurassic.c.

```
04838                {
04839
04840   FILE *out;
04841
04842   char file[LEN];
04843
04844   int id, ir, n = 10;
04845
04846   /* Set filename... */
04847   if (dirname != NULL)
04848     sprintf(file, "%s/%s", dirname, filename);
04849   else
04850     sprintf(file, "%s", filename);
04851
04852   /* Write info... */
04853   printf("Write observation data: %s\n", file);
04854
04855   /* Create file... */
04856   if (!(out = fopen(file, "w")))
04857     ERRMSG("Cannot create file!");
04858
```

```
04859   /* Write header... */
04860   fprintf(out,
04861           "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
04862           "# $2 = observer altitude [km]\n"
04863           "# $3 = observer longitude [deg]\n"
04864           "# $4 = observer latitude [deg]\n"
04865           "# $5 = view point altitude [km]\n"
04866           "# $6 = view point longitude [deg]\n"
04867           "# $7 = view point latitude [deg]\n"
04868           "# $8 = tangent point altitude [km]\n"
04869           "# $9 = tangent point longitude [deg]\n"
04870           "# $10 = tangent point latitude [deg]\n");
04871   for (id = 0; id < ctl->nd; id++)
04872     fprintf(out, "# $%d = channel %g: radiance [W/(m^2 sr cm^-1)]\n",
04873             ++n, ctl->nu[id]);
04874   for (id = 0; id < ctl->nd; id++)
04875     fprintf(out, "# $%d = channel %g: transmittance\n", ++n, ctl->nu[id]);
04876
04877   /* Write data... */
04878   for (ir = 0; ir < obs->nr; ir++) {
04879     if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
04880       fprintf(out, "\n");
04881     fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
04882             obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
04883             obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
04884             obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
04885     for (id = 0; id < ctl->nd; id++)
04886       fprintf(out, " %g", obs->rad[id][ir]);
04887     for (id = 0; id < ctl->nd; id++)
04888       fprintf(out, " %g", obs->tau[id][ir]);
04889     fprintf(out, "\n");
04890   }
04891
04892   /* Close file... */
04893   fclose(out);
04894 }
```

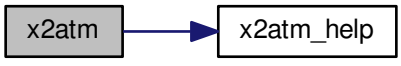
5.7.2.47 void x2atm (ctl_t * *ctl*, gsl_vector * *x*, atm_t * *atm*)

Decompose parameter vector or state vector.

Definition at line 4898 of file jurassic.c.

```
04901               {
04902
04903   int ig, iw;
04904
04905   size_t n = 0;
04906
04907   /* Set pressure... */
04908   x2atm_help(atm, ctl->retp_zmin, ctl->retp_zmax, atm->
      p, x, &n);
04909
04910   /* Set temperature... */
04911   x2atm_help(atm, ctl->rett_zmin, ctl->rett_zmax, atm->
      t, x, &n);
04912
04913   /* Set volume mixing ratio... */
04914   for (ig = 0; ig < ctl->ng; ig++)
04915     x2atm_help(atm, ctl->retq_zmin[ig], ctl->retq_zmax[ig],
04916                atm->q[ig], x, &n);
04917
04918   /* Set extinction... */
04919   for (iw = 0; iw < ctl->nw; iw++)
04920     x2atm_help(atm, ctl->retk_zmin[iw], ctl->retk_zmax[iw],
04921                atm->k[iw], x, &n);
04922 }
```

Here is the call graph for this function:



5.7.2.48 void x2atm_help (atm_t * *atm*, double *zmin*, double *zmax*, double * *value*, gsl_vector * *x*, size_t * *n*)

Extract elements from state vector.

Definition at line 4926 of file jurassic.c.

```
04932               {
04933
04934   int ip;
04935
04936   /* Extract state vector elements... */
04937   for (ip = 0; ip < atm->np; ip++)
04938     if (atm->z[ip] >= zmin && atm->z[ip] <= zmax) {
04939       value[ip] = gsl_vector_get(x, *n);
04940       (*n)++;
04941     }
04942 }
```

5.7.2.49 void y2obs (ctl_t * *ctl*, gsl_vector * *y*, obs_t * *obs*)

Decompose measurement vector.

Definition at line 4946 of file jurassic.c.

```
04949               {
04950
04951   int id, ir;
04952
04953   size_t m = 0;
04954
04955   /* Decompose measurement vector... */
04956   for (ir = 0; ir < obs->nr; ir++)
04957     for (id = 0; id < ctl->nd; id++)
04958       if (gsl_finite(obs->rad[id][ir])) {
04959         obs->rad[id][ir] = gsl_vector_get(y, m);
04960         m++;
04961       }
04962 }
```

5.8 jurassic.h

```
00001 /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU General Public License
00015   along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2003-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00034 #include <gsl/gsl_math.h>
00035 #include <gsl/gsl_blas.h>
00036 #include <gsl/gsl_linalg.h>
00037 #include <gsl/gsl_statistics.h>
00038 #include <math.h>
00039 #include <omp.h>
00040 #include <stdio.h>
00041 #include <stdlib.h>
00042 #include <string.h>
00043 #include <time.h>
00044
00045 /* ------------------------------------------------------------
```

```
00046    Macros...
00047    ------------------------------------------------------------ */
00048
00050 #define ALLOC(ptr, type, n)                             \
00051   if((ptr=malloc((size_t)(n)*sizeof(type)))==NULL)      \
00052     ERRMSG("Out of memory!");
00053
00055 #define DIST(a, b) sqrt(DIST2(a, b))
00056
00058 #define DIST2(a, b)                                                     \
00059   ((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1])+(a[2]-b[2])*(a[2]-b[2]))
00060
00062 #define DOTP(a, b) (a[0]*b[0]+a[1]*b[1]+a[2]*b[2])
00063
00065 #define ERRMSG(msg) {                                                   \
00066     printf("\nError (%s, %s, l%d): %s\n\n",                             \
00067            __FILE__, __func__, __LINE__, msg);                          \
00068     exit(EXIT_FAILURE);                                                 \
00069   }
00070
00072 #define EXP(x0, y0, x1, y1, x)                                  \
00073   (((y0)>0 && (y1)>0)                                           \
00074    ? ((y0)*exp(log((y1)/(y0))/((x1)-(x0))*((x)-(x0))))          \
00075    : LIN(x0, y0, x1, y1, x))
00076
00078 #define LIN(x0, y0, x1, y1, x)                  \
00079   ((y0)+((y1)-(y0))/((x1)-(x0))*((x)-(x0)))
00080
00082 #define NORM(a) sqrt(DOTP(a, a))
00083
00085 #define POW2(x) ((x)*(x))
00086
00088 #define POW3(x) ((x)*(x)*(x))
00089
00091 #define PRINT(format, var)                                              \
00092   printf("Print (%s, %s, l%d): %s= "format"\n",                         \
00093          __FILE__, __func__, __LINE__, #var, var);
00094
00096 #define TIMER(name, mode)                               \
00097   {timer(name, __FILE__, __func__, __LINE__, mode);}
00098
00100 #define TOK(line, tok, format, var) {                   \
00101     if(((tok)=strtok((line), " \t"))) {                 \
00102       if(sscanf(tok, format, &(var))!=1) continue;      \
00103     } else ERRMSG("Error while reading!");              \
00104   }
00105
00106 /* ------------------------------------------------------------
00107    Constants...
00108    ------------------------------------------------------------ */
00109
00111 #define TMIN 100.
00112
00114 #define TMAX 400.
00115
00117 #define C1 1.19104259e-8
00118
00120 #define C2 1.43877506
00121
00123 #define G0 9.80665
00124
00126 #define KB 1.3806504e-23
00127
00129 #define NA 6.02214199e23
00130
00132 #define P0 1013.25
00133
00135 #define T0 273.15
00136
00138 #define RE 6367.421
00139
00141 #define RI 8.3144598
00142
00144 #define ME 5.976e24
00145
00146 /* ------------------------------------------------------------
00147    Dimensions...
00148    ------------------------------------------------------------ */
00149
00151 #define ND 50
00152
00154 #define NG 20
00155
00157 #define NP 1000
00158
00160 #define NR 1000
00161
```

```
00163 #define NW 5
00164
00166 #define LEN 5000
00167
00169 #define M (NR*ND)
00170
00172 #define N (NQ*NP)
00173
00175 #define NQ (2+NG+NW)
00176
00178 #define NLOS 1000
00179
00181 #define NSHAPE 10000
00182
00184 #define NFOV 5
00185
00187 #define TBLNP 41
00188
00190 #define TBLNT 30
00191
00193 #define TBLNU 320
00194
00196 #define TBLNS 1200
00197
00198 /* -------------------------------------------------------------
00199    Quantity indices...
00200    ------------------------------------------------------------- */
00201
00203 #define IDXP 0
00204
00206 #define IDXT 1
00207
00209 #define IDXQ(ig) (2+ig)
00210
00212 #define IDXK(iw) (2+ctl->ng+iw)
00213
00214 /* -------------------------------------------------------------
00215    Structs...
00216    ------------------------------------------------------------- */
00217
00219 typedef struct {
00220
00222   int np;
00223
00225   double time[NP];
00226
00228   double z[NP];
00229
00231   double lon[NP];
00232
00234   double lat[NP];
00235
00237   double p[NP];
00238
00240   double t[NP];
00241
00243   double q[NG][NP];
00244
00246   double k[NW][NP];
00247
00248 } atm_t;
00249
00251 typedef struct {
00252
00254   int ng;
00255
00257   char emitter[NG][LEN];
00258
00260   int nd;
00261
00263   int nw;
00264
00266   double nu[ND];
00267
00269   int window[ND];
00270
00272   char tblbase[LEN];
00273
00275   double hydz;
00276
00278   int ctm_co2;
00279
00281   int ctm_h2o;
00282
00284   int ctm_n2;
00285
00287   int ctm_o2;
```

```
00288
00290   int refrac;
00291
00293   double rayds;
00294
00296   double raydz;
00297
00299   char fov[LEN];
00300
00302   double retp_zmin;
00303
00305   double retp_zmax;
00306
00308   double rett_zmin;
00309
00311   double rett_zmax;
00312
00314   double retq_zmin[NG];
00315
00317   double retq_zmax[NG];
00318
00320   double retk_zmin[NW];
00321
00323   double retk_zmax[NW];
00324
00326   int write_bbt;
00327
00329   int write_matrix;
00330
00331 } ctl_t;
00332
00334 typedef struct {
00335
00337   int np;
00338
00340   double z[NLOS];
00341
00343   double lon[NLOS];
00344
00346   double lat[NLOS];
00347
00349   double p[NLOS];
00350
00352   double t[NLOS];
00353
00355   double q[NG][NLOS];
00356
00358   double k[NW][NLOS];
00359
00361   double tsurf;
00362
00364   double ds[NLOS];
00365
00367   double u[NG][NLOS];
00368
00369 } los_t;
00370
00372 typedef struct {
00373
00375   int nr;
00376
00378   double time[NR];
00379
00381   double obsz[NR];
00382
00384   double obslon[NR];
00385
00387   double obslat[NR];
00388
00390   double vpz[NR];
00391
00393   double vplon[NR];
00394
00396   double vplat[NR];
00397
00399   double tpz[NR];
00400
00402   double tplon[NR];
00403
00405   double tplat[NR];
00406
00408   double tau[ND][NR];
00409
00411   double rad[ND][NR];
00412
00413 } obs_t;
00414
```

```
00416 typedef struct {
00417
00419   int np[NG][ND];
00420
00422   int nt[NG][ND][TBLNP];
00423
00425   int nu[NG][ND][TBLNP][TBLNT];
00426
00428   double p[NG][ND][TBLNP];
00429
00431   double t[NG][ND][TBLNP][TBLNT];
00432
00434   float u[NG][ND][TBLNP][TBLNT][TBLNU];
00435
00437   float eps[NG][ND][TBLNP][TBLNT][TBLNU];
00438
00440   double st[TBLNS];
00441
00443   double sr[ND][TBLNS];
00444
00445 } tbl_t;
00446
00447 /* ------------------------------------------------------------
00448    Functions...
00449    ------------------------------------------------------------ */
00450
00452 size_t atm2x(
00453   ctl_t * ctl,
00454   atm_t * atm,
00455   gsl_vector * x,
00456   int *iqa,
00457   int *ipa);
00458
00460 void atm2x_help(
00461   atm_t * atm,
00462   double zmin,
00463   double zmax,
00464   double *value,
00465   int val_iqa,
00466   gsl_vector * x,
00467   int *iqa,
00468   int *ipa,
00469   size_t * n);
00470
00472 double brightness(
00473   double rad,
00474   double nu);
00475
00477 void cart2geo(
00478   double *x,
00479   double *z,
00480   double *lon,
00481   double *lat);
00482
00484 void climatology(
00485   ctl_t * ctl,
00486   atm_t * atm_mean);
00487
00489 double ctmco2(
00490   double nu,
00491   double p,
00492   double t,
00493   double u);
00494
00496 double ctmh2o(
00497   double nu,
00498   double p,
00499   double t,
00500   double q,
00501   double u);
00502
00504 double ctmn2(
00505   double nu,
00506   double p,
00507   double t);
00508
00510 double ctmo2(
00511   double nu,
00512   double p,
00513   double t);
00514
00516 void copy_atm(
00517   ctl_t * ctl,
00518   atm_t * atm_dest,
00519   atm_t * atm_src,
00520   int init);
00521
```

```
00523 void copy_obs(
00524   ctl_t * ctl,
00525   obs_t * obs_dest,
00526   obs_t * obs_src,
00527   int init);
00528
00530 int find_emitter(
00531   ctl_t * ctl,
00532   const char *emitter);
00533
00535 void formod(
00536   ctl_t * ctl,
00537   atm_t * atm,
00538   obs_t * obs);
00539
00541 void formod_continua(
00542   ctl_t * ctl,
00543   los_t * los,
00544   int ip,
00545   double *beta);
00546
00548 void formod_fov(
00549   ctl_t * ctl,
00550   obs_t * obs);
00551
00553 void formod_pencil(
00554   ctl_t * ctl,
00555   atm_t * atm,
00556   obs_t * obs,
00557   int ir);
00558
00560 void formod_srcfunc(
00561   ctl_t * ctl,
00562   tbl_t * tbl,
00563   double t,
00564   double *src);
00565
00567 void geo2cart(
00568   double z,
00569   double lon,
00570   double lat,
00571   double *x);
00572
00574 void hydrostatic(
00575   ctl_t * ctl,
00576   atm_t * atm);
00577
00579 void idx2name(
00580   ctl_t * ctl,
00581   int idx,
00582   char *quantity);
00583
00585 void init_tbl(
00586   ctl_t * ctl,
00587   tbl_t * tbl);
00588
00590 void intpol_atm(
00591   ctl_t * ctl,
00592   atm_t * atm,
00593   double z,
00594   double *p,
00595   double *t,
00596   double *q,
00597   double *k);
00598
00600 void intpol_tbl(
00601   ctl_t * ctl,
00602   tbl_t * tbl,
00603   los_t * los,
00604   int ip,
00605   double tau_path[NG][ND],
00606   double tau_seg[ND]);
00607
00609 double intpol_tbl_eps(
00610   tbl_t * tbl,
00611   int ig,
00612   int id,
00613   int ip,
00614   int it,
00615   double u);
00616
00618 double intpol_tbl_u(
00619   tbl_t * tbl,
00620   int ig,
00621   int id,
00622   int ip,
00623   int it,
```

```
00624   double eps);
00625
00627 void jsec2time(
00628   double jsec,
00629   int *year,
00630   int *mon,
00631   int *day,
00632   int *hour,
00633   int *min,
00634   int *sec,
00635   double *remain);
00636
00638 void kernel(
00639   ctl_t * ctl,
00640   atm_t * atm,
00641   obs_t * obs,
00642   gsl_matrix * k);
00643
00645 int locate_irr(
00646   double *xx,
00647   int n,
00648   double x);
00649
00651 int locate_reg(
00652   double *xx,
00653   int n,
00654   double x);
00655
00657 int locate_tbl(
00658   float *xx,
00659   int n,
00660   double x);
00661
00663 size_t obs2y(
00664   ctl_t * ctl,
00665   obs_t * obs,
00666   gsl_vector * y,
00667   int *ida,
00668   int *ira);
00669
00671 double planck(
00672   double t,
00673   double nu);
00674
00676 void raytrace(
00677   ctl_t * ctl,
00678   atm_t * atm,
00679   obs_t * obs,
00680   los_t * los,
00681   int ir);
00682
00684 void read_atm(
00685   const char *dirname,
00686   const char *filename,
00687   ctl_t * ctl,
00688   atm_t * atm);
00689
00691 void read_ctl(
00692   int argc,
00693   char *argv[],
00694   ctl_t * ctl);
00695
00697 void read_matrix(
00698   const char *dirname,
00699   const char *filename,
00700   gsl_matrix * matrix);
00701
00703 void read_obs(
00704   const char *dirname,
00705   const char *filename,
00706   ctl_t * ctl,
00707   obs_t * obs);
00708
00710 void read_shape(
00711   const char *filename,
00712   double *x,
00713   double *y,
00714   int *n);
00715
00717 double refractivity(
00718   double p,
00719   double t);
00720
00722 double scan_ctl(
00723   int argc,
00724   char *argv[],
00725   const char *varname,
```

```
00726   int arridx,
00727   const char *defvalue,
00728   char *value);
00729
00731 void tangent_point(
00732   los_t * los,
00733   double *tpz,
00734   double *tplon,
00735   double *tplat);
00736
00738 void time2jsec(
00739   int year,
00740   int mon,
00741   int day,
00742   int hour,
00743   int min,
00744   int sec,
00745   double remain,
00746   double *jsec);
00747
00749 void timer(
00750   const char *name,
00751   const char *file,
00752   const char *func,
00753   int line,
00754   int mode);
00755
00757 void write_atm(
00758   const char *dirname,
00759   const char *filename,
00760   ctl_t * ctl,
00761   atm_t * atm);
00762
00764 void write_matrix(
00765   const char *dirname,
00766   const char *filename,
00767   ctl_t * ctl,
00768   gsl_matrix * matrix,
00769   atm_t * atm,
00770   obs_t * obs,
00771   const char *rowspace,
00772   const char *colspace,
00773   const char *sort);
00774
00776 void write_obs(
00777   const char *dirname,
00778   const char *filename,
00779   ctl_t * ctl,
00780   obs_t * obs);
00781
00783 void x2atm(
00784   ctl_t * ctl,
00785   gsl_vector * x,
00786   atm_t * atm);
00787
00789 void x2atm_help(
00790   atm_t * atm,
00791   double zmin,
00792   double zmax,
00793   double *value,
00794   gsl_vector * x,
00795   size_t * n);
00796
00798 void y2obs(
00799   ctl_t * ctl,
00800   gsl_vector * y,
00801   obs_t * obs);
```

5.9 libiasi.c File Reference

Functions

- void add_var (int ncid, const char *varname, const char *unit, const char *longname, int type, int dimid[ ], int *varid, int ndims)

  Add variable to netCDF file.
- void background_poly_help (double *xx, double *yy, int n, int dim)

  Get background based on polynomial fits.
- void background_poly (wave_t *wave, int dim_x, int dim_y)

Get background based on polynomial fits.

- void iasi_read (char *filename, iasi_rad_t *iasi_rad)

  Read IASI Level-1 data and convert to radiation type.
- void noise (wave_t *wave, double *mu, double *sig)

  Estimate noise.
- void pert2wave (pert_t *pert, wave_t *wave, int track0, int track1, int xtrack0, int xtrack1)

  Convert radiance perturbation data to wave analysis struct.
- void variance (wave_t *wave, double dh)

  Compute local variance.
- double wgs84 (double lat)

  Calculate Earth radius according to WGS-84 reference ellipsoid.
- void write_l1 (char *filename, iasi_l1_t *l1)

  Write IASI Level-1 data.
- void write_l2 (char *filename, iasi_l2_t *l2)

  Write IASI Level-2 data.

5.9.1 Function Documentation

5.9.1.1 void add_var (int *ncid*, const char * *varname*, const char * *unit*, const char * *longname*, int *type*, int *dimid[]*, int * *varid*, int *ndims*)

Add variable to netCDF file.

Definition at line 5 of file libiasi.c.

```
00013             {
00014
00015   /* Check if variable exists... */
00016   if (nc_inq_varid(ncid, varname, varid) != NC_NOERR) {
00017
00018     /* Define variable... */
00019     NC(nc_def_var(ncid, varname, type, ndims, dimid, varid));
00020
00021     /* Set long name... */
00022     NC(nc_put_att_text
00023        (ncid, *varid, "long_name", strlen(longname), longname));
00024
00025     /* Set units... */
00026     NC(nc_put_att_text(ncid, *varid, "units", strlen(unit), unit));
00027   }
00028 }
```

5.9.1.2 void background_poly_help (double * *xx*, double * *yy*, int *n*, int *dim*)

Get background based on polynomial fits.

Definition at line 32 of file libiasi.c.

```
00036           {
00037
00038   gsl_multifit_linear_workspace *work;
00039   gsl_matrix *cov, *X;
00040   gsl_vector *c, *x, *y;
00041
00042   double chisq, xx2[WX > WY ? WX : WY], yy2[WX > WY ? WX : WY];
00043
00044   size_t i, i2, n2 = 0;
00045
00046   /* Check for nan... */
00047   for (i = 0; i < (size_t) n; i++)
00048     if (gsl_finite(yy[i])) {
```

```
00049       xx2[n2] = xx[i];
00050       yy2[n2] = yy[i];
00051       n2++;
00052     }
00053   if ((int) n2 < dim || n2 < 0.9 * n) {
00054     for (i = 0; i < (size_t) n; i++)
00055       yy[i] = GSL_NAN;
00056     return;
00057   }
00058
00059   /* Allocate... */
00060   work = gsl_multifit_linear_alloc((size_t) n2, (size_t) dim);
00061   cov = gsl_matrix_alloc((size_t) dim, (size_t) dim);
00062   X = gsl_matrix_alloc((size_t) n2, (size_t) dim);
00063   c = gsl_vector_alloc((size_t) dim);
00064   x = gsl_vector_alloc((size_t) n2);
00065   y = gsl_vector_alloc((size_t) n2);
00066
00067   /* Compute polynomial fit... */
00068   for (i = 0; i < (size_t) n2; i++) {
00069     gsl_vector_set(x, i, xx2[i]);
00070     gsl_vector_set(y, i, yy2[i]);
00071     for (i2 = 0; i2 < (size_t) dim; i2++)
00072       gsl_matrix_set(X, i, i2, pow(gsl_vector_get(x, i), (double) i2));
00073   }
00074   gsl_multifit_linear(X, y, c, cov, &chisq, work);
00075   for (i = 0; i < (size_t) n; i++)
00076     yy[i] = gsl_poly_eval(c->data, (int) dim, xx[i]);
00077
00078   /* Free... */
00079   gsl_multifit_linear_free(work);
00080   gsl_matrix_free(cov);
00081   gsl_matrix_free(X);
00082   gsl_vector_free(c);
00083   gsl_vector_free(x);
00084   gsl_vector_free(y);
00085 }
```

5.9.1.3 void background_poly (wave_t * *wave*, int *dim_x*, int *dim_y*)

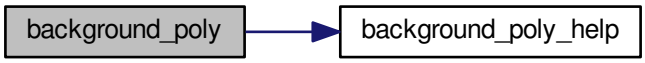
Get background based on polynomial fits.

Definition at line 89 of file libiasi.c.

```
00092             {
00093
00094   double help[WX], x[WX], x2[WY], y[WX], y2[WY];
00095
00096   int ix, iy;
00097
00098   /* Copy temperatures to background... */
00099   for (ix = 0; ix < wave->nx; ix++)
00100     for (iy = 0; iy < wave->ny; iy++) {
00101       wave->bg[ix][iy] = wave->temp[ix][iy];
00102       wave->pt[ix][iy] = 0;
00103     }
00104
00105   /* Check parameters... */
00106   if (dim_x <= 0 && dim_y <= 0)
00107     return;
00108
00109   /* Compute fit in x-direction... */
00110   if (dim_x > 0)
00111     for (iy = 0; iy < wave->ny; iy++) {
00112       for (ix = 0; ix <= 53; ix++) {
00113         x[ix] = (double) ix;
00114         y[ix] = wave->bg[ix][iy];
00115       }
00116       background_poly_help(x, y, 54, dim_x);
00117       for (ix = 0; ix <= 29; ix++)
00118         help[ix] = y[ix];
00119
00120       for (ix = 6; ix <= 59; ix++) {
00121         x[ix - 6] = (double) ix;
00122         y[ix - 6] = wave->bg[ix][iy];
00123       }
00124       background_poly_help(x, y, 54, dim_x);
00125       for (ix = 30; ix <= 59; ix++)
00126         help[ix] = y[ix - 6];
00127
```

```
00128       for (ix = 0; ix < wave->nx; ix++)
00129         wave->bg[ix][iy] = help[ix];
00130     }
00131
00132   /* Compute fit in y-direction... */
00133   if (dim_y > 0)
00134     for (ix = 0; ix < wave->nx; ix++) {
00135       for (iy = 0; iy < wave->ny; iy++) {
00136         x2[iy] = (int) iy;
00137         y2[iy] = wave->bg[ix][iy];
00138       }
00139       background_poly_help(x2, y2, wave->ny, dim_y);
00140       for (iy = 0; iy < wave->ny; iy++)
00141         wave->bg[ix][iy] = y2[iy];
00142     }
00143
00144   /* Recompute perturbations... */
00145   for (ix = 0; ix < wave->nx; ix++)
00146     for (iy = 0; iy < wave->ny; iy++)
00147       wave->pt[ix][iy] = wave->temp[ix][iy] - wave->bg[ix][iy];
00148 }
```

Here is the call graph for this function:



5.9.1.4 void iasi_read (char * *filename*, iasi_rad_t * *iasi_rad*)

Read IASI Level-1 data and convert to radiation type.

Definition at line 152 of file libiasi.c.

```
00154                           {
00155
00156   const char *product_class;
00157
00158   coda_product *pf;
00159
00160   coda_cursor cursor;
00161
00162   iasi_raw_t *iasi_raw;
00163
00164   int i, j, w, tr1, tr2, tr1_lpm, tr1_rpm, tr2_lpm, tr2_rpm,
00165     ichan, mdr_i, num_dims = 1;
00166
00167   long dim[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
00168
00169   short int IDefScaleSondNbScale, IDefScaleSondNsfirst[10],
00170     IDefScaleSondNslast[10], IDefScaleSondScaleFactor[10];
00171
00172   float sc, scaling[IASI_L1_NCHAN];
00173
00174   /* Initialize CODA... */
00175   coda_init();
00176
00177   /* Allocate... */
00178   ALLOC(iasi_raw, iasi_raw_t, 1);
00179
00180   /* Open IASI file... */
00181   CODA(coda_open(filename, &pf));
00182   CODA(coda_get_product_class(pf, &product_class));
00183   CODA(coda_cursor_set_product(&cursor, pf));
00184
00185   /* Get scaling parameters... */
```

```
00186   CODA(coda_cursor_goto_record_field_by_name(&cursor, "GIADR_ScaleFactors"));
00187
00188   CODA(coda_cursor_goto_record_field_by_name
00189        (&cursor, "IDefScaleSondNbScale"));
00190   CODA(coda_cursor_read_int16(&cursor, &IDefScaleSondNbScale));
00191   CODA(coda_cursor_goto_parent(&cursor));
00192
00193   CODA(coda_cursor_goto_record_field_by_name
00194        (&cursor, "IDefScaleSondNsfirst"));
00195   CODA(coda_cursor_read_int16_array
00196        (&cursor, IDefScaleSondNsfirst, coda_array_ordering_c));
00197   CODA(coda_cursor_goto_parent(&cursor));
00198
00199   CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefScaleSondNslast"));
00200   CODA(coda_cursor_read_int16_array
00201        (&cursor, IDefScaleSondNslast, coda_array_ordering_c));
00202   CODA(coda_cursor_goto_parent(&cursor));
00203
00204   CODA(coda_cursor_goto_record_field_by_name
00205        (&cursor, "IDefScaleSondScaleFactor"));
00206   CODA(coda_cursor_read_int16_array
00207        (&cursor, IDefScaleSondScaleFactor, coda_array_ordering_c));
00208
00209   /* Compute scaling factors... */
00210   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00211     scaling[ichan] = GSL_NAN;
00212   for (i = 0; i < IDefScaleSondNbScale; i++) {
00213     sc = (float) pow(10.0, -IDefScaleSondScaleFactor[i]);
00214     for (ichan = IDefScaleSondNsfirst[i] - 1;
00215          ichan < IDefScaleSondNslast[i]; ichan++) {
00216       w = ichan - IASI_IDefNsfirst1b + 1;
00217       if (w >= 0 && w < IASI_L1_NCHAN)
00218         scaling[w] = sc;
00219     }
00220   }
00221
00222   /* Get number of tracks in record... */
00223   CODA(coda_cursor_goto_root(&cursor));
00224   CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00225   CODA(coda_cursor_get_array_dim(&cursor, &num_dims, dim));
00226   iasi_raw->ntrack = dim[0];
00227
00228   /* Read tracks one by one... */
00229   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00230
00231     /* Reset cursor position... */
00232     CODA(coda_cursor_goto_root(&cursor));
00233
00234     /* Move cursor to radiation data... */
00235     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00236     CODA(coda_cursor_goto_array_element_by_index(&cursor, mdr_i));
00237     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00238     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GS1cSpect"));
00239     CODA(coda_cursor_read_int16_array
00240          (&cursor, &iasi_raw->Radiation[mdr_i][0][0][0],
00241           coda_array_ordering_c));
00242
00243     /* Read time... */
00244     CODA(coda_cursor_goto_parent(&cursor));
00245     CODA(coda_cursor_goto_record_field_by_name(&cursor, "OnboardUTC"));
00246     CODA(coda_cursor_read_double_array
00247          (&cursor, &iasi_raw->Time[mdr_i][0], coda_array_ordering_c));
00248
00249     /* Read coordinates... */
00250     CODA(coda_cursor_goto_parent(&cursor));
00251     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GGeoSondLoc"));
00252     CODA(coda_cursor_read_double_array
00253          (&cursor, &iasi_raw->Loc[mdr_i][0][0][0], coda_array_ordering_c));
00254
00255     /* Read satellite altitude... */
00256     CODA(coda_cursor_goto_parent(&cursor));
00257     CODA(coda_cursor_goto_record_field_by_name(&cursor,
00258                                                "EARTH_SATELLITE_DISTANCE"));
00259     CODA(coda_cursor_read_uint32(&cursor, &iasi_raw->Sat_z[mdr_i]));
00260
00261     /* Read spectral range... */
00262     iasi_raw->IDefSpectDWn1b[mdr_i] = IASI_IDefSpectDWn1b / 100.0;
00263
00264     CODA(coda_cursor_goto_parent(&cursor));
00265     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNsfirst1b"));
00266     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNsfirst1b[mdr_i]));
00267     if (iasi_raw->IDefNsfirst1b[mdr_i] != IASI_IDefNsfirst1b)
00268       ERRMSG("Unexpected value for IDefNsfirst1b!");
00269
00270     CODA(coda_cursor_goto_parent(&cursor));
00271     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNslast1b"));
00272     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNslast1b[mdr_i]));
```

```
00273     if (iasi_raw->IDefNslast1b[mdr_i] != IASI_IDefNslast1b)
00274       ERRMSG("Unexpected value for IDefNslast1b!");
00275
00276     /* Compute wavenumber... */
00277     if (mdr_i == 0)
00278       for (i = 0; i < IASI_L1_NCHAN; i++)
00279         iasi_raw->Wavenumber[i] =
00280           iasi_raw->IDefSpectDWn1b[mdr_i] *
00281           (float) (iasi_raw->IDefNsfirst1b[mdr_i] + i - 1);
00282   }
00283
00284   /* Close file... */
00285   CODA(coda_close(pf));
00286
00287   /* Finalize CODA... */
00288   coda_done();
00289
00290   /* Set number of tracks... */
00291   iasi_rad->ntrack = (int) (iasi_raw->ntrack * 2);
00292
00293   /* Copy wavenumbers... */
00294   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00295     iasi_rad->freq[ichan] = iasi_raw->Wavenumber[ichan];
00296
00297   /* Copy footprint data... */
00298   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00299     tr1 = mdr_i * 2;
00300     tr2 = mdr_i * 2 + 1;
00301     tr1_lpm = 3;
00302     tr1_rpm = 0;
00303     tr2_lpm = 2;
00304     tr2_rpm = 1;
00305
00306     /* Copy time (2x2 matrix has same measurement time)...  */
00307     for (i = 0; i < IASI_NXTRACK; i++) {
00308       iasi_rad->Time[tr1][i * 2] = iasi_raw->Time[mdr_i][i];
00309       iasi_rad->Time[tr1][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00310       iasi_rad->Time[tr2][i * 2] = iasi_raw->Time[mdr_i][i];
00311       iasi_rad->Time[tr2][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00312     }
00313
00314     /* Copy location... */
00315     for (i = 0; i < IASI_NXTRACK; i++) {
00316       iasi_rad->Longitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][0];
00317       iasi_rad->Longitude[tr1][i * 2 + 1] =
00318         iasi_raw->Loc[mdr_i][i][tr1_rpm][0];
00319       iasi_rad->Latitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][1];
00320       iasi_rad->Latitude[tr1][i * 2 + 1] =
00321         iasi_raw->Loc[mdr_i][i][tr1_rpm][1];
00322
00323       iasi_rad->Longitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][0];
00324       iasi_rad->Longitude[tr2][i * 2 + 1] =
00325         iasi_raw->Loc[mdr_i][i][tr2_rpm][0];
00326       iasi_rad->Latitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][1];
00327       iasi_rad->Latitude[tr2][i * 2 + 1] =
00328         iasi_raw->Loc[mdr_i][i][tr2_rpm][1];
00329     }
00330
00331     /* Copy satellite location (we only have one height value)... */
00332     iasi_rad->Sat_lon[tr1] = iasi_rad->Longitude[tr1][28];
00333     iasi_rad->Sat_lat[tr1] = iasi_rad->Latitude[tr1][28];
00334     iasi_rad->Sat_lon[tr2] = iasi_rad->Longitude[tr2][28];
00335     iasi_rad->Sat_lat[tr2] = iasi_rad->Latitude[tr2][28];
00336     iasi_rad->Sat_z[tr1] =
00337       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr1]);
00338     iasi_rad->Sat_z[tr2] =
00339       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr2]);
00340
00341     /* Copy radiation data... */
00342     for (i = 0; i < IASI_NXTRACK; i++) {
00343       for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++) {
00344         sc = scaling[ichan] * 100.0f;
00345         iasi_rad->Rad[tr1][i * 2][ichan] =
00346           iasi_raw->Radiation[mdr_i][i][tr1_lpm][ichan] * sc;
00347         iasi_rad->Rad[tr1][i * 2 + 1][ichan] =
00348           iasi_raw->Radiation[mdr_i][i][tr1_rpm][ichan] * sc;
00349         iasi_rad->Rad[tr2][i * 2][ichan] =
00350           iasi_raw->Radiation[mdr_i][i][tr2_lpm][ichan] * sc;
00351         iasi_rad->Rad[tr2][i * 2 + 1][ichan] =
00352           iasi_raw->Radiation[mdr_i][i][tr2_rpm][ichan] * sc;
00353       }
00354     }
00355   }
00356
00357   /* Check radiance data... */
00358   for (i = 0; i < iasi_rad->ntrack; i++)
00359     for (j = 0; j < L1_NXTRACK; j++)
```

```
00360       if (iasi_rad->Rad[i][j][6753] > iasi_rad->Rad[i][j][6757]
00361           || iasi_rad->Rad[i][j][6753] < 0)
00362         for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00363           iasi_rad->Rad[i][j][ichan] = GSL_NAN;
00364
00365   /* Free... */
00366   free(iasi_raw);
00367 }
```

Here is the call graph for this function:

5.9.1.5 void noise (wave_t * *wave*, double * *mu*, double * *sig*)

Estimate noise.

Definition at line 371 of file libiasi.c.

```
00374               {
00375
00376   int ix, ix2, iy, iy2, n = 0, okay;
00377
00378   /* Init... */
00379   *mu = 0;
00380   *sig = 0;
00381
00382   /* Estimate noise (Immerkaer, 1996)... */
00383   for (ix = 1; ix < wave->nx - 1; ix++)
00384     for (iy = 1; iy < wave->ny - 1; iy++) {
00385
00386       /* Check data... */
00387       okay = 1;
00388       for (ix2 = ix - 1; ix2 <= ix + 1; ix2++)
00389         for (iy2 = iy - 1; iy2 <= iy + 1; iy2++)
00390           if (!gsl_finite(wave->temp[ix2][iy2]))
00391             okay = 0;
00392       if (!okay)
00393         continue;
00394
00395       /* Get mean noise... */
00396       n++;
00397       *mu += wave->temp[ix][iy];
00398       *sig += gsl_pow_2(+4. / 6. * wave->temp[ix][iy]
00399                         - 2. / 6. * (wave->temp[ix - 1][iy]
00400                                      + wave->temp[ix + 1][iy]
00401                                      + wave->temp[ix][iy - 1]
00402                                      + wave->temp[ix][iy + 1])
00403                         + 1. / 6. * (wave->temp[ix - 1][iy - 1]
00404                                      + wave->temp[ix + 1][iy - 1]
00405                                      + wave->temp[ix - 1][iy + 1]
00406                                      + wave->temp[ix + 1][iy + 1]));
00407     }
00408
00409   /* Normalize... */
00410   *mu /= (double) n;
00411   *sig = sqrt(*sig / (double) n);
00412 }
```

5.9.1.6 void pert2wave (pert_t * *pert*, wave_t * *wave*, int *track0*, int *track1*, int *xtrack0*, int *xtrack1*)

Convert radiance perturbation data to wave analysis struct.

Definition at line 416 of file libiasi.c.

```
00422               {
00423
00424   double x0[3], x1[3];
00425
00426   int itrack, ixtrack;
00427
00428   /* Check ranges... */
00429   track0 = GSL_MIN(GSL_MAX(track0, 0), pert->ntrack - 1);
00430   track1 = GSL_MIN(GSL_MAX(track1, 0), pert->ntrack - 1);
00431   xtrack0 = GSL_MIN(GSL_MAX(xtrack0, 0), pert->nxtrack - 1);
00432   xtrack1 = GSL_MIN(GSL_MAX(xtrack1, 0), pert->nxtrack - 1);
00433
00434   /* Set size... */
00435   wave->nx = xtrack1 - xtrack0 + 1;
00436   if (wave->nx > WX)
00437     ERRMSG("Too many across-track values!");
00438   wave->ny = track1 - track0 + 1;
00439   if (wave->ny > WY)
00440     ERRMSG("Too many along-track values!");
00441
00442   /* Loop over footprints... */
00443   for (itrack = track0; itrack <= track1; itrack++)
00444     for (ixtrack = xtrack0; ixtrack <= xtrack1; ixtrack++) {
00445
00446       /* Get distances... */
00447       if (itrack == track0) {
00448         wave->x[0] = 0;
00449         if (ixtrack > xtrack0) {
00450           geo2cart(0, pert->lon[itrack][ixtrack - 1],
00451                    pert->lat[itrack][ixtrack - 1], x0);
00452           geo2cart(0, pert->lon[itrack][ixtrack],
00453                    pert->lat[itrack][ixtrack], x1);
00454           wave->x[ixtrack - xtrack0] =
00455             wave->x[ixtrack - xtrack0 - 1] + DIST(x0, x1);
00456         }
00457       }
00458       if (ixtrack == xtrack0) {
00459         wave->y[0] = 0;
00460         if (itrack > track0) {
00461           geo2cart(0, pert->lon[itrack - 1][ixtrack],
00462                    pert->lat[itrack - 1][ixtrack], x0);
00463           geo2cart(0, pert->lon[itrack][ixtrack],
00464                    pert->lat[itrack][ixtrack], x1);
00465           wave->y[itrack - track0] =
00466             wave->y[itrack - track0 - 1] + DIST(x0, x1);
00467         }
00468       }
00469
00470       /* Save geolocation... */
00471       wave->time = pert->time[(track0 + track1) / 2][(xtrack0 + xtrack1) / 2];
00472       wave->z = 0;
00473       wave->lon[ixtrack - xtrack0][itrack - track0] =
00474         pert->lon[itrack][ixtrack];
00475       wave->lat[ixtrack - xtrack0][itrack - track0] =
00476         pert->lat[itrack][ixtrack];
00477
00478       /* Save temperature data... */
00479       wave->temp[ixtrack - xtrack0][itrack - track0]
00480         = pert->bt[itrack][ixtrack];
00481       wave->bg[ixtrack - xtrack0][itrack - track0]
00482         = pert->bt[itrack][ixtrack] - pert->pt[itrack][ixtrack];
00483       wave->pt[ixtrack - xtrack0][itrack - track0]
00484         = pert->pt[itrack][ixtrack];
00485       wave->var[ixtrack - xtrack0][itrack - track0]
00486         = pert->var[itrack][ixtrack];
00487     }
00488 }
```

Here is the call graph for this function:

5.9.1.7 void variance (wave_t * *wave*, double *dh*)

Compute local variance.

Definition at line 492 of file libiasi.c.

```
00494             {
00495
00496   double dh2, mu, help;
00497
00498   int dx, dy, ix, ix2, iy, iy2, n;
00499
00500   /* Check parameters... */
00501   if (dh <= 0)
00502     return;
00503
00504   /* Compute squared radius... */
00505   dh2 = gsl_pow_2(dh);
00506
00507   /* Get sampling distances... */
00508   dx =
00509     (int) (dh / fabs(wave->x[wave->nx - 1] - wave->x[0]) * (wave->nx - 1.0) +
00510            1);
00511   dy =
00512     (int) (dh / fabs(wave->y[wave->ny - 1] - wave->y[0]) * (wave->ny - 1.0) +
00513            1);
00514
00515   /* Loop over data points... */
00516   for (ix = 0; ix < wave->nx; ix++)
00517     for (iy = 0; iy < wave->ny; iy++) {
00518
00519       /* Init... */
00520       mu = help = 0;
00521       n = 0;
00522
00523       /* Get data... */
00524       for (ix2 = GSL_MAX(ix - dx, 0); ix2 <= GSL_MIN(ix + dx, wave->nx - 1);
00525            ix2++)
00526         for (iy2 = GSL_MAX(iy - dy, 0); iy2 <= GSL_MIN(iy + dy, wave->ny - 1);
00527              iy2++)
00528           if ((gsl_pow_2(wave->x[ix] - wave->x[ix2])
00529                + gsl_pow_2(wave->y[iy] - wave->y[iy2])) <= dh2)
00530             if (gsl_finite(wave->pt[ix2][iy2])) {
00531               mu += wave->pt[ix2][iy2];
00532               help += gsl_pow_2(wave->pt[ix2][iy2]);
00533               n++;
00534             }
00535
00536       /* Compute local variance... */
00537       if (n > 1)
00538         wave->var[ix][iy] = help / n - gsl_pow_2(mu / n);
00539       else
00540         wave->var[ix][iy] = GSL_NAN;
00541     }
00542 }
```

5.9.1.8 double wgs84 (double *lat*)

Calculate Earth radius according to WGS-84 reference ellipsoid.

Definition at line 546 of file libiasi.c.

```
00547               {
00548
00549   const double a = 6378.1370, b = 6356.7523;
00550
00551   double cphi, sphi;
00552
00553   cphi = cos(lat * M_PI / 180.);
00554   sphi = sin(lat * M_PI / 180.);
00555
00556   return sqrt((gsl_pow_2(a * a * cphi) + gsl_pow_2(b * b * sphi))
00557               / (gsl_pow_2(a * cphi) + gsl_pow_2(b * sphi)));
00558 }
```

5.9.1.9 void write_l1 (char * *filename*, iasi_l1_t * *l1*)

Write IASI Level-1 data.

Definition at line 562 of file libiasi.c.

```
00564                   {
00565
00566   int dimid[10], ncid, time_id, lon_id, lat_id,
00567     sat_z_id, sat_lon_id, sat_lat_id, nu_id, rad_id;
00568
00569   /* Open or create netCDF file... */
00570   printf("Write IASI Level-1 file: %s\n", filename);
00571   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00572     NC(nc_create(filename, NC_CLOBBER, &ncid));
00573   } else {
00574     NC(nc_redef(ncid));
00575   }
00576
00577   /* Set dimensions... */
00578   if (nc_inq_dimid(ncid, "L1_NTRACK", &dimid[0]) != NC_NOERR)
00579     NC(nc_def_dim(ncid, "L1_NTRACK", l1->ntrack, &dimid[0]));
00580   if (nc_inq_dimid(ncid, "L1_NXTRACK", &dimid[1]) != NC_NOERR)
00581     NC(nc_def_dim(ncid, "L1_NXTRACK", L1_NXTRACK, &dimid[1]));
00582   if (nc_inq_dimid(ncid, "L1_NCHAN", &dimid[2]) != NC_NOERR)
00583     NC(nc_def_dim(ncid, "L1_NCHAN", L1_NCHAN, &dimid[2]));
00584
00585   /* Add variables... */
00586   add_var(ncid, "l1_time", "s", "time (seconds since 2000-01-01T00:00Z)",
00587           NC_DOUBLE, dimid, &time_id, 2);
00588   add_var(ncid, "l1_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00589   add_var(ncid, "l1_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00590   add_var(ncid, "l1_sat_z", "km", "satellite altitude",
00591           NC_DOUBLE, dimid, &sat_z_id, 1);
00592   add_var(ncid, "l1_sat_lon", "deg", "(estimated) satellite longitude",
00593           NC_DOUBLE, dimid, &sat_lon_id, 1);
00594   add_var(ncid, "l1_sat_lat", "deg", "(estimated) satellite latitude",
00595           NC_DOUBLE, dimid, &sat_lat_id, 1);
00596   add_var(ncid, "l1_nu", "cm^-1", "channel wavenumber",
00597           NC_DOUBLE, &dimid[2], &nu_id, 1);
00598   add_var(ncid, "l1_rad", "W/(m^2 sr cm^-1)", "channel radiance",
00599           NC_FLOAT, dimid, &rad_id, 3);
00600
00601   /* Leave define mode... */
00602   NC(nc_enddef(ncid));
00603
00604   /* Write data... */
00605   NC(nc_put_var_double(ncid, time_id, l1->time[0]));
00606   NC(nc_put_var_double(ncid, lon_id, l1->lon[0]));
00607   NC(nc_put_var_double(ncid, lat_id, l1->lat[0]));
00608   NC(nc_put_var_double(ncid, sat_z_id, l1->sat_z));
00609   NC(nc_put_var_double(ncid, sat_lon_id, l1->sat_lon));
00610   NC(nc_put_var_double(ncid, sat_lat_id, l1->sat_lat));
00611   NC(nc_put_var_double(ncid, nu_id, l1->nu));
00612   NC(nc_put_var_float(ncid, rad_id, &l1->rad[0][0][0]));
00613
00614   /* Close file... */
00615   NC(nc_close(ncid));
00616 }
```

Here is the call graph for this function:

5.9.1.10 void write_l2 (char * *filename*, iasi_l2_t * *l2*)

Write IASI Level-2 data.

Definition at line 620 of file libiasi.c.

```
00622                   {
00623
00624   int dimid[10], ncid, time_id, z_id, lon_id, lat_id, p_id, t_id;
00625
00626   /* Create netCDF file... */
00627   printf("Write IASI Level-2 file: %s\n", filename);
00628   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00629     NC(nc_create(filename, NC_CLOBBER, &ncid));
00630   } else {
00631     NC(nc_redef(ncid));
00632   }
00633
00634   /* Set dimensions... */
00635   if (nc_inq_dimid(ncid, "L2_NTRACK", &dimid[0]) != NC_NOERR)
00636     NC(nc_def_dim(ncid, "L2_NTRACK", l2->ntrack, &dimid[0]));
00637   if (nc_inq_dimid(ncid, "L2_NXTRACK", &dimid[1]) != NC_NOERR)
00638     NC(nc_def_dim(ncid, "L2_NXTRACK", L2_NXTRACK, &dimid[1]));
00639   if (nc_inq_dimid(ncid, "L2_NLAY", &dimid[2]) != NC_NOERR)
00640     NC(nc_def_dim(ncid, "L2_NLAY", L2_NLAY, &dimid[2]));
00641
00642   /* Add variables... */
00643   add_var(ncid, "l2_time", "s", "time (seconds since 2000-01-01T00:00Z)",
00644           NC_DOUBLE, dimid, &time_id, 2);
00645   add_var(ncid, "l2_z", "km", "altitude", NC_DOUBLE, dimid, &z_id, 3);
00646   add_var(ncid, "l2_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00647   add_var(ncid, "l2_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00648   add_var(ncid, "l2_press", "hPa", "pressure",
00649           NC_DOUBLE, &dimid[2], &p_id, 1);
00650   add_var(ncid, "l2_temp", "K", "temperature", NC_DOUBLE, dimid, &t_id, 3);
00651
00652   /* Leave define mode... */
00653   NC(nc_enddef(ncid));
00654
00655   /* Write data... */
00656   NC(nc_put_var_double(ncid, time_id, l2->time[0]));
00657   NC(nc_put_var_double(ncid, z_id, l2->z[0][0]));
00658   NC(nc_put_var_double(ncid, lon_id, l2->lon[0]));
00659   NC(nc_put_var_double(ncid, lat_id, l2->lat[0]));
00660   NC(nc_put_var_double(ncid, p_id, l2->p));
00661   NC(nc_put_var_double(ncid, t_id, l2->t[0][0]));
00662
00663   /* Close file... */
00664   NC(nc_close(ncid));
00665 }
```

Here is the call graph for this function:

5.10 libiasi.c

```
00001 #include "libiasi.h"
00002
00003 /*****************************************************************************/
00004
00005 void add_var(
00006   int ncid,
00007   const char *varname,
00008   const char *unit,
00009   const char *longname,
00010   int type,
00011   int dimid[],
00012   int *varid,
00013   int ndims) {
00014
00015   /* Check if variable exists... */
00016   if (nc_inq_varid(ncid, varname, varid) != NC_NOERR) {
00017
00018     /* Define variable... */
00019     NC(nc_def_var(ncid, varname, type, ndims, dimid, varid));
00020
00021     /* Set long name... */
00022     NC(nc_put_att_text
00023        (ncid, *varid, "long_name", strlen(longname), longname));
00024
00025     /* Set units... */
00026     NC(nc_put_att_text(ncid, *varid, "units", strlen(unit), unit));
00027   }
00028 }
00029
00030 /*****************************************************************************/
00031
00032 void background_poly_help(
00033   double *xx,
00034   double *yy,
00035   int n,
00036   int dim) {
00037
00038   gsl_multifit_linear_workspace *work;
00039   gsl_matrix *cov, *X;
00040   gsl_vector *c, *x, *y;
00041
00042   double chisq, xx2[WX > WY ? WX : WY], yy2[WX > WY ? WX : WY];
00043
00044   size_t i, i2, n2 = 0;
00045
00046   /* Check for nan... */
00047   for (i = 0; i < (size_t) n; i++)
00048     if (gsl_finite(yy[i])) {
00049       xx2[n2] = xx[i];
00050       yy2[n2] = yy[i];
00051       n2++;
00052     }
00053   if ((int) n2 < dim || n2 < 0.9 * n) {
00054     for (i = 0; i < (size_t) n; i++)
00055       yy[i] = GSL_NAN;
00056     return;
00057   }
00058
00059   /* Allocate... */
00060   work = gsl_multifit_linear_alloc((size_t) n2, (size_t) dim);
00061   cov = gsl_matrix_alloc((size_t) dim, (size_t) dim);
00062   X = gsl_matrix_alloc((size_t) n2, (size_t) dim);
00063   c = gsl_vector_alloc((size_t) dim);
```

```
00064   x = gsl_vector_alloc((size_t) n2);
00065   y = gsl_vector_alloc((size_t) n2);
00066
00067   /* Compute polynomial fit... */
00068   for (i = 0; i < (size_t) n2; i++) {
00069     gsl_vector_set(x, i, xx2[i]);
00070     gsl_vector_set(y, i, yy2[i]);
00071     for (i2 = 0; i2 < (size_t) dim; i2++)
00072       gsl_matrix_set(X, i, i2, pow(gsl_vector_get(x, i), (double) i2));
00073   }
00074   gsl_multifit_linear(X, y, c, cov, &chisq, work);
00075   for (i = 0; i < (size_t) n; i++)
00076     yy[i] = gsl_poly_eval(c->data, (int) dim, xx[i]);
00077
00078   /* Free... */
00079   gsl_multifit_linear_free(work);
00080   gsl_matrix_free(cov);
00081   gsl_matrix_free(X);
00082   gsl_vector_free(c);
00083   gsl_vector_free(x);
00084   gsl_vector_free(y);
00085 }
00086
00087 /*****************************************************************************/
00088
00089 void background_poly(
00090   wave_t * wave,
00091   int dim_x,
00092   int dim_y) {
00093
00094   double help[WX], x[WX], x2[WY], y[WX], y2[WY];
00095
00096   int ix, iy;
00097
00098   /* Copy temperatures to background... */
00099   for (ix = 0; ix < wave->nx; ix++)
00100     for (iy = 0; iy < wave->ny; iy++) {
00101       wave->bg[ix][iy] = wave->temp[ix][iy];
00102       wave->pt[ix][iy] = 0;
00103     }
00104
00105   /* Check parameters... */
00106   if (dim_x <= 0 && dim_y <= 0)
00107     return;
00108
00109   /* Compute fit in x-direction... */
00110   if (dim_x > 0)
00111     for (iy = 0; iy < wave->ny; iy++) {
00112       for (ix = 0; ix <= 53; ix++) {
00113         x[ix] = (double) ix;
00114         y[ix] = wave->bg[ix][iy];
00115       }
00116       background_poly_help(x, y, 54, dim_x);
00117       for (ix = 0; ix <= 29; ix++)
00118         help[ix] = y[ix];
00119
00120       for (ix = 6; ix <= 59; ix++) {
00121         x[ix - 6] = (double) ix;
00122         y[ix - 6] = wave->bg[ix][iy];
00123       }
00124       background_poly_help(x, y, 54, dim_x);
00125       for (ix = 30; ix <= 59; ix++)
00126         help[ix] = y[ix - 6];
00127
00128       for (ix = 0; ix < wave->nx; ix++)
00129         wave->bg[ix][iy] = help[ix];
00130     }
00131
00132   /* Compute fit in y-direction... */
00133   if (dim_y > 0)
00134     for (ix = 0; ix < wave->nx; ix++) {
00135       for (iy = 0; iy < wave->ny; iy++) {
00136         x2[iy] = (int) iy;
00137         y2[iy] = wave->bg[ix][iy];
00138       }
00139       background_poly_help(x2, y2, wave->ny, dim_y);
00140       for (iy = 0; iy < wave->ny; iy++)
00141         wave->bg[ix][iy] = y2[iy];
00142     }
00143
00144   /* Recompute perturbations... */
00145   for (ix = 0; ix < wave->nx; ix++)
00146     for (iy = 0; iy < wave->ny; iy++)
00147       wave->pt[ix][iy] = wave->temp[ix][iy] - wave->bg[ix][iy];
00148 }
00149
00150 /*****************************************************************************/
```

```
00151
00152 void iasi_read(
00153   char *filename,
00154   iasi_rad_t * iasi_rad) {
00155
00156   const char *product_class;
00157
00158   coda_product *pf;
00159
00160   coda_cursor cursor;
00161
00162   iasi_raw_t *iasi_raw;
00163
00164   int i, j, w, tr1, tr2, tr1_lpm, tr1_rpm, tr2_lpm, tr2_rpm,
00165     ichan, mdr_i, num_dims = 1;
00166
00167   long dim[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
00168
00169   short int IDefScaleSondNbScale, IDefScaleSondNsfirst[10],
00170     IDefScaleSondNslast[10], IDefScaleSondScaleFactor[10];
00171
00172   float sc, scaling[IASI_L1_NCHAN];
00173
00174   /* Initialize CODA... */
00175   coda_init();
00176
00177   /* Allocate... */
00178   ALLOC(iasi_raw, iasi_raw_t, 1);
00179
00180   /* Open IASI file... */
00181   CODA(coda_open(filename, &pf));
00182   CODA(coda_get_product_class(pf, &product_class));
00183   CODA(coda_cursor_set_product(&cursor, pf));
00184
00185   /* Get scaling parameters... */
00186   CODA(coda_cursor_goto_record_field_by_name(&cursor, "GIADR_ScaleFactors"));
00187
00188   CODA(coda_cursor_goto_record_field_by_name
00189        (&cursor, "IDefScaleSondNbScale"));
00190   CODA(coda_cursor_read_int16(&cursor, &IDefScaleSondNbScale));
00191   CODA(coda_cursor_goto_parent(&cursor));
00192
00193   CODA(coda_cursor_goto_record_field_by_name
00194        (&cursor, "IDefScaleSondNsfirst"));
00195   CODA(coda_cursor_read_int16_array
00196        (&cursor, IDefScaleSondNsfirst, coda_array_ordering_c));
00197   CODA(coda_cursor_goto_parent(&cursor));
00198
00199   CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefScaleSondNslast"));
00200   CODA(coda_cursor_read_int16_array
00201        (&cursor, IDefScaleSondNslast, coda_array_ordering_c));
00202   CODA(coda_cursor_goto_parent(&cursor));
00203
00204   CODA(coda_cursor_goto_record_field_by_name
00205        (&cursor, "IDefScaleSondScaleFactor"));
00206   CODA(coda_cursor_read_int16_array
00207        (&cursor, IDefScaleSondScaleFactor, coda_array_ordering_c));
00208
00209   /* Compute scaling factors... */
00210   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00211     scaling[ichan] = GSL_NAN;
00212   for (i = 0; i < IDefScaleSondNbScale; i++) {
00213     sc = (float) pow(10.0, -IDefScaleSondScaleFactor[i]);
00214     for (ichan = IDefScaleSondNsfirst[i] - 1;
00215          ichan < IDefScaleSondNslast[i]; ichan++) {
00216       w = ichan - IASI_IDefNsfirst1b + 1;
00217       if (w >= 0 && w < IASI_L1_NCHAN)
00218         scaling[w] = sc;
00219     }
00220   }
00221
00222   /* Get number of tracks in record... */
00223   CODA(coda_cursor_goto_root(&cursor));
00224   CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00225   CODA(coda_cursor_get_array_dim(&cursor, &num_dims, dim));
00226   iasi_raw->ntrack = dim[0];
00227
00228   /* Read tracks one by one... */
00229   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00230
00231     /* Reset cursor position... */
00232     CODA(coda_cursor_goto_root(&cursor));
00233
00234     /* Move cursor to radiation data... */
00235     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00236     CODA(coda_cursor_goto_array_element_by_index(&cursor, mdr_i));
00237     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
```

```
00238     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GS1cSpect"));
00239     CODA(coda_cursor_read_int16_array
00240          (&cursor, &iasi_raw->Radiation[mdr_i][0][0][0],
00241           coda_array_ordering_c));
00242
00243     /* Read time... */
00244     CODA(coda_cursor_goto_parent(&cursor));
00245     CODA(coda_cursor_goto_record_field_by_name(&cursor, "OnboardUTC"));
00246     CODA(coda_cursor_read_double_array
00247          (&cursor, &iasi_raw->Time[mdr_i][0], coda_array_ordering_c));
00248
00249     /* Read coordinates... */
00250     CODA(coda_cursor_goto_parent(&cursor));
00251     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GGeoSondLoc"));
00252     CODA(coda_cursor_read_double_array
00253          (&cursor, &iasi_raw->Loc[mdr_i][0][0][0], coda_array_ordering_c));
00254
00255     /* Read satellite altitude... */
00256     CODA(coda_cursor_goto_parent(&cursor));
00257     CODA(coda_cursor_goto_record_field_by_name(&cursor,
00258                                                "EARTH_SATELLITE_DISTANCE"));
00259     CODA(coda_cursor_read_uint32(&cursor, &iasi_raw->Sat_z[mdr_i]));
00260
00261     /* Read spectral range... */
00262     iasi_raw->IDefSpectDWn1b[mdr_i] = IASI_IDefSpectDWn1b / 100.0;
00263
00264     CODA(coda_cursor_goto_parent(&cursor));
00265     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNsfirst1b"));
00266     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNsfirst1b[mdr_i]));
00267     if (iasi_raw->IDefNsfirst1b[mdr_i] != IASI_IDefNsfirst1b)
00268       ERRMSG("Unexpected value for IDefNsfirst1b!");
00269
00270     CODA(coda_cursor_goto_parent(&cursor));
00271     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNslast1b"));
00272     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNslast1b[mdr_i]));
00273     if (iasi_raw->IDefNslast1b[mdr_i] != IASI_IDefNslast1b)
00274       ERRMSG("Unexpected value for IDefNslast1b!");
00275
00276     /* Compute wavenumber... */
00277     if (mdr_i == 0)
00278       for (i = 0; i < IASI_L1_NCHAN; i++)
00279         iasi_raw->Wavenumber[i] =
00280           iasi_raw->IDefSpectDWn1b[mdr_i] *
00281           (float) (iasi_raw->IDefNsfirst1b[mdr_i] + i - 1);
00282   }
00283
00284   /* Close file... */
00285   CODA(coda_close(pf));
00286
00287   /* Finalize CODA... */
00288   coda_done();
00289
00290   /* Set number of tracks... */
00291   iasi_rad->ntrack = (int) (iasi_raw->ntrack * 2);
00292
00293   /* Copy wavenumbers... */
00294   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00295     iasi_rad->freq[ichan] = iasi_raw->Wavenumber[ichan];
00296
00297   /* Copy footprint data... */
00298   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00299     tr1 = mdr_i * 2;
00300     tr2 = mdr_i * 2 + 1;
00301     tr1_lpm = 3;
00302     tr1_rpm = 0;
00303     tr2_lpm = 2;
00304     tr2_rpm = 1;
00305
00306     /* Copy time (2x2 matrix has same measurement time)...  */
00307     for (i = 0; i < IASI_NXTRACK; i++) {
00308       iasi_rad->Time[tr1][i * 2] = iasi_raw->Time[mdr_i][i];
00309       iasi_rad->Time[tr1][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00310       iasi_rad->Time[tr2][i * 2] = iasi_raw->Time[mdr_i][i];
00311       iasi_rad->Time[tr2][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00312     }
00313
00314     /* Copy location... */
00315     for (i = 0; i < IASI_NXTRACK; i++) {
00316       iasi_rad->Longitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][0];
00317       iasi_rad->Longitude[tr1][i * 2 + 1] =
00318         iasi_raw->Loc[mdr_i][i][tr1_rpm][0];
00319       iasi_rad->Latitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][1];
00320       iasi_rad->Latitude[tr1][i * 2 + 1] =
00321         iasi_raw->Loc[mdr_i][i][tr1_rpm][1];
00322
00323       iasi_rad->Longitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][0];
00324       iasi_rad->Longitude[tr2][i * 2 + 1] =
```

```
00325         iasi_raw->Loc[mdr_i][i][tr2_rpm][0];
00326       iasi_rad->Latitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][1];
00327       iasi_rad->Latitude[tr2][i * 2 + 1] =
00328         iasi_raw->Loc[mdr_i][i][tr2_rpm][1];
00329     }
00330
00331     /* Copy satellite location (we only have one height value)... */
00332     iasi_rad->Sat_lon[tr1] = iasi_rad->Longitude[tr1][28];
00333     iasi_rad->Sat_lat[tr1] = iasi_rad->Latitude[tr1][28];
00334     iasi_rad->Sat_lon[tr2] = iasi_rad->Longitude[tr2][28];
00335     iasi_rad->Sat_lat[tr2] = iasi_rad->Latitude[tr2][28];
00336     iasi_rad->Sat_z[tr1] =
00337       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr1]);
00338     iasi_rad->Sat_z[tr2] =
00339       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr2]);
00340
00341     /* Copy radiation data... */
00342     for (i = 0; i < IASI_NXTRACK; i++) {
00343       for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++) {
00344         sc = scaling[ichan] * 100.0f;
00345         iasi_rad->Rad[tr1][i * 2][ichan] =
00346           iasi_raw->Radiation[mdr_i][i][tr1_lpm][ichan] * sc;
00347         iasi_rad->Rad[tr1][i * 2 + 1][ichan] =
00348           iasi_raw->Radiation[mdr_i][i][tr1_rpm][ichan] * sc;
00349         iasi_rad->Rad[tr2][i * 2][ichan] =
00350           iasi_raw->Radiation[mdr_i][i][tr2_lpm][ichan] * sc;
00351         iasi_rad->Rad[tr2][i * 2 + 1][ichan] =
00352           iasi_raw->Radiation[mdr_i][i][tr2_rpm][ichan] * sc;
00353       }
00354     }
00355   }
00356
00357   /* Check radiance data... */
00358   for (i = 0; i < iasi_rad->ntrack; i++)
00359     for (j = 0; j < L1_NXTRACK; j++)
00360       if (iasi_rad->Rad[i][j][6753] > iasi_rad->Rad[i][j][6757]
00361           || iasi_rad->Rad[i][j][6753] < 0)
00362         for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00363           iasi_rad->Rad[i][j][ichan] = GSL_NAN;
00364
00365   /* Free... */
00366   free(iasi_raw);
00367 }
00368
00369 /*****************************************************************************/
00370
00371 void noise(
00372   wave_t * wave,
00373   double *mu,
00374   double *sig) {
00375
00376   int ix, ix2, iy, iy2, n = 0, okay;
00377
00378   /* Init... */
00379   *mu = 0;
00380   *sig = 0;
00381
00382   /* Estimate noise (Immerkaer, 1996)... */
00383   for (ix = 1; ix < wave->nx - 1; ix++)
00384     for (iy = 1; iy < wave->ny - 1; iy++) {
00385
00386       /* Check data... */
00387       okay = 1;
00388       for (ix2 = ix - 1; ix2 <= ix + 1; ix2++)
00389         for (iy2 = iy - 1; iy2 <= iy + 1; iy2++)
00390           if (!gsl_finite(wave->temp[ix2][iy2]))
00391             okay = 0;
00392       if (!okay)
00393         continue;
00394
00395       /* Get mean noise... */
00396       n++;
00397       *mu += wave->temp[ix][iy];
00398       *sig += gsl_pow_2(+4. / 6. * wave->temp[ix][iy]
00399                         - 2. / 6. * (wave->temp[ix - 1][iy]
00400                                      + wave->temp[ix + 1][iy]
00401                                      + wave->temp[ix][iy - 1]
00402                                      + wave->temp[ix][iy + 1])
00403                         + 1. / 6. * (wave->temp[ix - 1][iy - 1]
00404                                      + wave->temp[ix + 1][iy - 1]
00405                                      + wave->temp[ix - 1][iy + 1]
00406                                      + wave->temp[ix + 1][iy + 1]));
00407     }
00408
00409   /* Normalize... */
00410   *mu /= (double) n;
00411   *sig = sqrt(*sig / (double) n);
```

```
00412 }
00413
00414 /*****************************************************************************/
00415
00416 void pert2wave(
00417   pert_t * pert,
00418   wave_t * wave,
00419   int track0,
00420   int track1,
00421   int xtrack0,
00422   int xtrack1) {
00423
00424   double x0[3], x1[3];
00425
00426   int itrack, ixtrack;
00427
00428   /* Check ranges... */
00429   track0 = GSL_MIN(GSL_MAX(track0, 0), pert->ntrack - 1);
00430   track1 = GSL_MIN(GSL_MAX(track1, 0), pert->ntrack - 1);
00431   xtrack0 = GSL_MIN(GSL_MAX(xtrack0, 0), pert->nxtrack - 1);
00432   xtrack1 = GSL_MIN(GSL_MAX(xtrack1, 0), pert->nxtrack - 1);
00433
00434   /* Set size... */
00435   wave->nx = xtrack1 - xtrack0 + 1;
00436   if (wave->nx > WX)
00437     ERRMSG("Too many across-track values!");
00438   wave->ny = track1 - track0 + 1;
00439   if (wave->ny > WY)
00440     ERRMSG("Too many along-track values!");
00441
00442   /* Loop over footprints... */
00443   for (itrack = track0; itrack <= track1; itrack++)
00444     for (ixtrack = xtrack0; ixtrack <= xtrack1; ixtrack++) {
00445
00446       /* Get distances... */
00447       if (itrack == track0) {
00448         wave->x[0] = 0;
00449         if (ixtrack > xtrack0) {
00450           geo2cart(0, pert->lon[itrack][ixtrack - 1],
00451                    pert->lat[itrack][ixtrack - 1], x0);
00452           geo2cart(0, pert->lon[itrack][ixtrack],
00453                    pert->lat[itrack][ixtrack], x1);
00454           wave->x[ixtrack - xtrack0] =
00455             wave->x[ixtrack - xtrack0 - 1] + DIST(x0, x1);
00456         }
00457       }
00458       if (ixtrack == xtrack0) {
00459         wave->y[0] = 0;
00460         if (itrack > track0) {
00461           geo2cart(0, pert->lon[itrack - 1][ixtrack],
00462                    pert->lat[itrack - 1][ixtrack], x0);
00463           geo2cart(0, pert->lon[itrack][ixtrack],
00464                    pert->lat[itrack][ixtrack], x1);
00465           wave->y[itrack - track0] =
00466             wave->y[itrack - track0 - 1] + DIST(x0, x1);
00467         }
00468       }
00469
00470       /* Save geolocation... */
00471       wave->time = pert->time[(track0 + track1) / 2][(xtrack0 + xtrack1) / 2];
00472       wave->z = 0;
00473       wave->lon[ixtrack - xtrack0][itrack - track0] =
00474         pert->lon[itrack][ixtrack];
00475       wave->lat[ixtrack - xtrack0][itrack - track0] =
00476         pert->lat[itrack][ixtrack];
00477
00478       /* Save temperature data... */
00479       wave->temp[ixtrack - xtrack0][itrack - track0]
00480         = pert->bt[itrack][ixtrack];
00481       wave->bg[ixtrack - xtrack0][itrack - track0]
00482         = pert->bt[itrack][ixtrack] - pert->pt[itrack][ixtrack];
00483       wave->pt[ixtrack - xtrack0][itrack - track0]
00484         = pert->pt[itrack][ixtrack];
00485       wave->var[ixtrack - xtrack0][itrack - track0]
00486         = pert->var[itrack][ixtrack];
00487     }
00488 }
00489
00490 /*****************************************************************************/
00491
00492 void variance(
00493   wave_t * wave,
00494   double dh) {
00495
00496   double dh2, mu, help;
00497
00498   int dx, dy, ix, ix2, iy, iy2, n;
```

```
00499
00500   /* Check parameters... */
00501   if (dh <= 0)
00502     return;
00503
00504   /* Compute squared radius... */
00505   dh2 = gsl_pow_2(dh);
00506
00507   /* Get sampling distances... */
00508   dx =
00509     (int) (dh / fabs(wave->x[wave->nx - 1] - wave->x[0]) * (wave->nx - 1.0) +
00510            1);
00511   dy =
00512     (int) (dh / fabs(wave->y[wave->ny - 1] - wave->y[0]) * (wave->ny - 1.0) +
00513            1);
00514
00515   /* Loop over data points... */
00516   for (ix = 0; ix < wave->nx; ix++)
00517     for (iy = 0; iy < wave->ny; iy++) {
00518
00519       /* Init... */
00520       mu = help = 0;
00521       n = 0;
00522
00523       /* Get data... */
00524       for (ix2 = GSL_MAX(ix - dx, 0); ix2 <= GSL_MIN(ix + dx, wave->nx - 1);
00525            ix2++)
00526         for (iy2 = GSL_MAX(iy - dy, 0); iy2 <= GSL_MIN(iy + dy, wave->ny - 1);
00527              iy2++)
00528           if ((gsl_pow_2(wave->x[ix] - wave->x[ix2])
00529                + gsl_pow_2(wave->y[iy] - wave->y[iy2])) <= dh2)
00530             if (gsl_finite(wave->pt[ix2][iy2])) {
00531               mu += wave->pt[ix2][iy2];
00532               help += gsl_pow_2(wave->pt[ix2][iy2]);
00533               n++;
00534             }
00535
00536       /* Compute local variance... */
00537       if (n > 1)
00538         wave->var[ix][iy] = help / n - gsl_pow_2(mu / n);
00539       else
00540         wave->var[ix][iy] = GSL_NAN;
00541     }
00542 }
00543
00544 /*****************************************************************************/
00545
00546 double wgs84(
00547   double lat) {
00548
00549   const double a = 6378.1370, b = 6356.7523;
00550
00551   double cphi, sphi;
00552
00553   cphi = cos(lat * M_PI / 180.);
00554   sphi = sin(lat * M_PI / 180.);
00555
00556   return sqrt((gsl_pow_2(a * a * cphi) + gsl_pow_2(b * b * sphi))
00557               / (gsl_pow_2(a * cphi) + gsl_pow_2(b * sphi)));
00558 }
00559
00560 /*****************************************************************************/
00561
00562 void write_l1(
00563   char *filename,
00564   iasi_l1_t * l1) {
00565
00566   int dimid[10], ncid, time_id, lon_id, lat_id,
00567     sat_z_id, sat_lon_id, sat_lat_id, nu_id, rad_id;
00568
00569   /* Open or create netCDF file... */
00570   printf("Write IASI Level-1 file: %s\n", filename);
00571   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00572     NC(nc_create(filename, NC_CLOBBER, &ncid));
00573   } else {
00574     NC(nc_redef(ncid));
00575   }
00576
00577   /* Set dimensions... */
00578   if (nc_inq_dimid(ncid, "L1_NTRACK", &dimid[0]) != NC_NOERR)
00579     NC(nc_def_dim(ncid, "L1_NTRACK", l1->ntrack, &dimid[0]));
00580   if (nc_inq_dimid(ncid, "L1_NXTRACK", &dimid[1]) != NC_NOERR)
00581     NC(nc_def_dim(ncid, "L1_NXTRACK", L1_NXTRACK, &dimid[1]));
00582   if (nc_inq_dimid(ncid, "L1_NCHAN", &dimid[2]) != NC_NOERR)
00583     NC(nc_def_dim(ncid, "L1_NCHAN", L1_NCHAN, &dimid[2]));
00584
00585   /* Add variables... */
```

```
00586   add_var(ncid, "l1_time", "s", "time (seconds since 2000-01-01T00:00Z)",
00587           NC_DOUBLE, dimid, &time_id, 2);
00588   add_var(ncid, "l1_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00589   add_var(ncid, "l1_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00590   add_var(ncid, "l1_sat_z", "km", "satellite altitude",
00591           NC_DOUBLE, dimid, &sat_z_id, 1);
00592   add_var(ncid, "l1_sat_lon", "deg", "(estimated) satellite longitude",
00593           NC_DOUBLE, dimid, &sat_lon_id, 1);
00594   add_var(ncid, "l1_sat_lat", "deg", "(estimated) satellite latitude",
00595           NC_DOUBLE, dimid, &sat_lat_id, 1);
00596   add_var(ncid, "l1_nu", "cm^-1", "channel wavenumber",
00597           NC_DOUBLE, &dimid[2], &nu_id, 1);
00598   add_var(ncid, "l1_rad", "W/(m^2 sr cm^-1)", "channel radiance",
00599           NC_FLOAT, dimid, &rad_id, 3);
00600
00601   /* Leave define mode... */
00602   NC(nc_enddef(ncid));
00603
00604   /* Write data... */
00605   NC(nc_put_var_double(ncid, time_id, l1->time[0]));
00606   NC(nc_put_var_double(ncid, lon_id, l1->lon[0]));
00607   NC(nc_put_var_double(ncid, lat_id, l1->lat[0]));
00608   NC(nc_put_var_double(ncid, sat_z_id, l1->sat_z));
00609   NC(nc_put_var_double(ncid, sat_lon_id, l1->sat_lon));
00610   NC(nc_put_var_double(ncid, sat_lat_id, l1->sat_lat));
00611   NC(nc_put_var_double(ncid, nu_id, l1->nu));
00612   NC(nc_put_var_float(ncid, rad_id, &l1->rad[0][0][0]));
00613
00614   /* Close file... */
00615   NC(nc_close(ncid));
00616 }
00617
00618 /*****************************************************************************/
00619
00620 void write_l2(
00621   char *filename,
00622   iasi_l2_t * l2) {
00623
00624   int dimid[10], ncid, time_id, z_id, lon_id, lat_id, p_id, t_id;
00625
00626   /* Create netCDF file... */
00627   printf("Write IASI Level-2 file: %s\n", filename);
00628   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00629     NC(nc_create(filename, NC_CLOBBER, &ncid));
00630   } else {
00631     NC(nc_redef(ncid));
00632   }
00633
00634   /* Set dimensions... */
00635   if (nc_inq_dimid(ncid, "L2_NTRACK", &dimid[0]) != NC_NOERR)
00636     NC(nc_def_dim(ncid, "L2_NTRACK", l2->ntrack, &dimid[0]));
00637   if (nc_inq_dimid(ncid, "L2_NXTRACK", &dimid[1]) != NC_NOERR)
00638     NC(nc_def_dim(ncid, "L2_NXTRACK", L2_NXTRACK, &dimid[1]));
00639   if (nc_inq_dimid(ncid, "L2_NLAY", &dimid[2]) != NC_NOERR)
00640     NC(nc_def_dim(ncid, "L2_NLAY", L2_NLAY, &dimid[2]));
00641
00642   /* Add variables... */
00643   add_var(ncid, "l2_time", "s", "time (seconds since 2000-01-01T00:00Z)",
00644           NC_DOUBLE, dimid, &time_id, 2);
00645   add_var(ncid, "l2_z", "km", "altitude", NC_DOUBLE, dimid, &z_id, 3);
00646   add_var(ncid, "l2_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00647   add_var(ncid, "l2_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00648   add_var(ncid, "l2_press", "hPa", "pressure",
00649           NC_DOUBLE, &dimid[2], &p_id, 1);
00650   add_var(ncid, "l2_temp", "K", "temperature", NC_DOUBLE, dimid, &t_id, 3);
00651
00652   /* Leave define mode... */
00653   NC(nc_enddef(ncid));
00654
00655   /* Write data... */
00656   NC(nc_put_var_double(ncid, time_id, l2->time[0]));
00657   NC(nc_put_var_double(ncid, z_id, l2->z[0][0]));
00658   NC(nc_put_var_double(ncid, lon_id, l2->lon[0]));
00659   NC(nc_put_var_double(ncid, lat_id, l2->lat[0]));
00660   NC(nc_put_var_double(ncid, p_id, l2->p));
00661   NC(nc_put_var_double(ncid, t_id, l2->t[0][0]));
00662
00663   /* Close file... */
00664   NC(nc_close(ncid));
00665 }
```

5.11 libiasi.h File Reference

Data Structures

- struct iasi_l1_t

  IASI Level-1 data.
- struct iasi_l2_t

  IASI Level-2 data.
- struct pert_t

  Perturbation data.
- struct iasi_raw_t

  IASI raw Level-1 data.
- struct iasi_rad_t

  IASI converted Level-1 radiation data.
- struct wave_t

  Wave analysis data.

Functions

- void add_var (int ncid, const char *varname, const char *unit, const char *longname, int type, int dimid[ ], int *varid, int ndims)

  Add variable to netCDF file.
- void background_poly (wave_t *wave, int dim_x, int dim_y)

  Get background based on polynomial fits.
- void background_poly_help (double *xx, double *yy, int n, int dim)

  Get background based on polynomial fits.
- int get_chan_for_wavenumber (float wavenumber)

  Get closest channel for a wavenumber [cm] (uses expected min wavenumber).
- void iasi_read (char *filename, iasi_rad_t *iasi_rad)

  Read IASI Level-1 data and convert to radiation type.
- void noise (wave_t *wave, double *mu, double *sig)

  Estimate noise.
- void pert2wave (pert_t *pert, wave_t *wave, int track0, int track1, int xtrack0, int xtrack1)

  Convert radiance perturbation data to wave analysis struct.
- void variance (wave_t *wave, double dh)

  Compute local variance.
- double wgs84 (double lat)

  Calculate Earth radius according to WGS-84 reference ellipsoid.
- void write_l1 (char *filename, iasi_l1_t *l1)

  Write IASI Level-1 data.
- void write_l2 (char *filename, iasi_l2_t *l2)

  Write IASI Level-2 data.

5.11.1 Function Documentation

5.11.1.1 void add_var (int *ncid*, const char * *varname*, const char * *unit*, const char * *longname*, int *type*, int *dimid[ ]*, int * *varid*, int *ndims*)

Add variable to netCDF file.

Add variable to netCDF file.

Definition at line 5 of file libiasi.c.

```
00013             {
00014
00015   /* Check if variable exists... */
00016   if (nc_inq_varid(ncid, varname, varid) != NC_NOERR) {
00017
00018     /* Define variable... */
00019     NC(nc_def_var(ncid, varname, type, ndims, dimid, varid));
00020
00021     /* Set long name... */
00022     NC(nc_put_att_text
00023        (ncid, *varid, "long_name", strlen(longname), longname));
00024
00025     /* Set units... */
00026     NC(nc_put_att_text(ncid, *varid, "units", strlen(unit), unit));
00027   }
00028 }
```

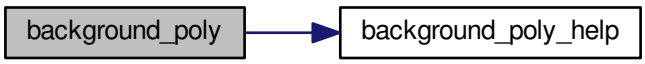
5.11.1.2 void background_poly (wave_t * *wave*, int *dim_x*, int *dim_y*)

Get background based on polynomial fits.

Definition at line 89 of file libiasi.c.

```
00092             {
00093
00094   double help[WX], x[WX], x2[WY], y[WX], y2[WY];
00095
00096   int ix, iy;
00097
00098   /* Copy temperatures to background... */
00099   for (ix = 0; ix < wave->nx; ix++)
00100     for (iy = 0; iy < wave->ny; iy++) {
00101       wave->bg[ix][iy] = wave->temp[ix][iy];
00102       wave->pt[ix][iy] = 0;
00103     }
00104
00105   /* Check parameters... */
00106   if (dim_x <= 0 && dim_y <= 0)
00107     return;
00108
00109   /* Compute fit in x-direction... */
00110   if (dim_x > 0)
00111     for (iy = 0; iy < wave->ny; iy++) {
00112       for (ix = 0; ix <= 53; ix++) {
00113         x[ix] = (double) ix;
00114         y[ix] = wave->bg[ix][iy];
00115       }
00116       background_poly_help(x, y, 54, dim_x);
00117       for (ix = 0; ix <= 29; ix++)
00118         help[ix] = y[ix];
00119
00120       for (ix = 6; ix <= 59; ix++) {
00121         x[ix - 6] = (double) ix;
00122         y[ix - 6] = wave->bg[ix][iy];
00123       }
00124       background_poly_help(x, y, 54, dim_x);
00125       for (ix = 30; ix <= 59; ix++)
00126         help[ix] = y[ix - 6];
00127
00128       for (ix = 0; ix < wave->nx; ix++)
00129         wave->bg[ix][iy] = help[ix];
00130     }
00131
00132   /* Compute fit in y-direction... */
00133   if (dim_y > 0)
00134     for (ix = 0; ix < wave->nx; ix++) {
00135       for (iy = 0; iy < wave->ny; iy++) {
00136         x2[iy] = (int) iy;
00137         y2[iy] = wave->bg[ix][iy];
00138       }
00139       background_poly_help(x2, y2, wave->ny, dim_y);
00140       for (iy = 0; iy < wave->ny; iy++)
00141         wave->bg[ix][iy] = y2[iy];
00142     }
00143
00144   /* Recompute perturbations... */
00145   for (ix = 0; ix < wave->nx; ix++)
00146     for (iy = 0; iy < wave->ny; iy++)
00147       wave->pt[ix][iy] = wave->temp[ix][iy] - wave->bg[ix][iy];
00148 }
```

Here is the call graph for this function:



5.11.1.3 void background_poly_help (double * *xx*, double * *yy*, int *n*, int *dim*)

Get background based on polynomial fits.

Definition at line 32 of file libiasi.c.

```
00036           {
00037
00038   gsl_multifit_linear_workspace *work;
00039   gsl_matrix *cov, *X;
00040   gsl_vector *c, *x, *y;
00041
00042   double chisq, xx2[WX > WY ? WX : WY], yy2[WX > WY ? WX : WY];
00043
00044   size_t i, i2, n2 = 0;
00045
00046   /* Check for nan... */
00047   for (i = 0; i < (size_t) n; i++)
00048     if (gsl_finite(yy[i])) {
00049       xx2[n2] = xx[i];
00050       yy2[n2] = yy[i];
00051       n2++;
00052     }
00053   if ((int) n2 < dim || n2 < 0.9 * n) {
00054     for (i = 0; i < (size_t) n; i++)
00055       yy[i] = GSL_NAN;
00056     return;
00057   }
00058
00059   /* Allocate... */
00060   work = gsl_multifit_linear_alloc((size_t) n2, (size_t) dim);
00061   cov = gsl_matrix_alloc((size_t) dim, (size_t) dim);
00062   X = gsl_matrix_alloc((size_t) n2, (size_t) dim);
00063   c = gsl_vector_alloc((size_t) dim);
00064   x = gsl_vector_alloc((size_t) n2);
00065   y = gsl_vector_alloc((size_t) n2);
00066
00067   /* Compute polynomial fit... */
00068   for (i = 0; i < (size_t) n2; i++) {
00069     gsl_vector_set(x, i, xx2[i]);
00070     gsl_vector_set(y, i, yy2[i]);
00071     for (i2 = 0; i2 < (size_t) dim; i2++)
00072       gsl_matrix_set(X, i, i2, pow(gsl_vector_get(x, i), (double) i2));
00073   }
00074   gsl_multifit_linear(X, y, c, cov, &chisq, work);
00075   for (i = 0; i < (size_t) n; i++)
00076     yy[i] = gsl_poly_eval(c->data, (int) dim, xx[i]);
00077
00078   /* Free... */
00079   gsl_multifit_linear_free(work);
00080   gsl_matrix_free(cov);
00081   gsl_matrix_free(X);
00082   gsl_vector_free(c);
00083   gsl_vector_free(x);
00084   gsl_vector_free(y);
00085 }
```

5.11.1.4 int get_chan_for_wavenumber (float *wavenumber*)

Get closest channel for a wavenumber [cm] (uses expected min wavenumber).

5.11.1.5 void iasi_read (char * *filename*, iasi_rad_t * *iasi_rad*)

Read IASI Level-1 data and convert to radiation type.

Definition at line 152 of file libiasi.c.

```
00154                           {
00155
00156   const char *product_class;
00157
00158   coda_product *pf;
00159
00160   coda_cursor cursor;
00161
00162   iasi_raw_t *iasi_raw;
00163
00164   int i, j, w, tr1, tr2, tr1_lpm, tr1_rpm, tr2_lpm, tr2_rpm,
00165     ichan, mdr_i, num_dims = 1;
00166
00167   long dim[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
00168
00169   short int IDefScaleSondNbScale, IDefScaleSondNsfirst[10],
00170     IDefScaleSondNslast[10], IDefScaleSondScaleFactor[10];
00171
00172   float sc, scaling[IASI_L1_NCHAN];
00173
00174   /* Initialize CODA... */
00175   coda_init();
00176
00177   /* Allocate... */
00178   ALLOC(iasi_raw, iasi_raw_t, 1);
00179
00180   /* Open IASI file... */
00181   CODA(coda_open(filename, &pf));
00182   CODA(coda_get_product_class(pf, &product_class));
00183   CODA(coda_cursor_set_product(&cursor, pf));
00184
00185   /* Get scaling parameters... */
00186   CODA(coda_cursor_goto_record_field_by_name(&cursor, "GIADR_ScaleFactors"));
00187
00188   CODA(coda_cursor_goto_record_field_by_name
00189        (&cursor, "IDefScaleSondNbScale"));
00190   CODA(coda_cursor_read_int16(&cursor, &IDefScaleSondNbScale));
00191   CODA(coda_cursor_goto_parent(&cursor));
00192
00193   CODA(coda_cursor_goto_record_field_by_name
00194        (&cursor, "IDefScaleSondNsfirst"));
00195   CODA(coda_cursor_read_int16_array
00196        (&cursor, IDefScaleSondNsfirst, coda_array_ordering_c));
00197   CODA(coda_cursor_goto_parent(&cursor));
00198
00199   CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefScaleSondNslast"));
00200   CODA(coda_cursor_read_int16_array
00201        (&cursor, IDefScaleSondNslast, coda_array_ordering_c));
00202   CODA(coda_cursor_goto_parent(&cursor));
00203
00204   CODA(coda_cursor_goto_record_field_by_name
00205        (&cursor, "IDefScaleSondScaleFactor"));
00206   CODA(coda_cursor_read_int16_array
00207        (&cursor, IDefScaleSondScaleFactor, coda_array_ordering_c));
00208
00209   /* Compute scaling factors... */
00210   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00211     scaling[ichan] = GSL_NAN;
00212   for (i = 0; i < IDefScaleSondNbScale; i++) {
00213     sc = (float) pow(10.0, -IDefScaleSondScaleFactor[i]);
00214     for (ichan = IDefScaleSondNsfirst[i] - 1;
00215          ichan < IDefScaleSondNslast[i]; ichan++) {
00216       w = ichan - IASI_IDefNsfirst1b + 1;
00217       if (w >= 0 && w < IASI_L1_NCHAN)
00218         scaling[w] = sc;
00219     }
00220   }
00221
00222   /* Get number of tracks in record... */
00223   CODA(coda_cursor_goto_root(&cursor));
00224   CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00225   CODA(coda_cursor_get_array_dim(&cursor, &num_dims, dim));
00226   iasi_raw->ntrack = dim[0];
00227
00228   /* Read tracks one by one... */
00229   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00230
```

```
00231     /* Reset cursor position... */
00232     CODA(coda_cursor_goto_root(&cursor));
00233
00234     /* Move cursor to radiation data... */
00235     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00236     CODA(coda_cursor_goto_array_element_by_index(&cursor, mdr_i));
00237     CODA(coda_cursor_goto_record_field_by_name(&cursor, "MDR"));
00238     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GS1cSpect"));
00239     CODA(coda_cursor_read_int16_array
00240          (&cursor, &iasi_raw->Radiation[mdr_i][0][0][0],
00241           coda_array_ordering_c));
00242
00243     /* Read time... */
00244     CODA(coda_cursor_goto_parent(&cursor));
00245     CODA(coda_cursor_goto_record_field_by_name(&cursor, "OnboardUTC"));
00246     CODA(coda_cursor_read_double_array
00247          (&cursor, &iasi_raw->Time[mdr_i][0], coda_array_ordering_c));
00248
00249     /* Read coordinates... */
00250     CODA(coda_cursor_goto_parent(&cursor));
00251     CODA(coda_cursor_goto_record_field_by_name(&cursor, "GGeoSondLoc"));
00252     CODA(coda_cursor_read_double_array
00253          (&cursor, &iasi_raw->Loc[mdr_i][0][0][0], coda_array_ordering_c));
00254
00255     /* Read satellite altitude... */
00256     CODA(coda_cursor_goto_parent(&cursor));
00257     CODA(coda_cursor_goto_record_field_by_name(&cursor,
00258                                                "EARTH_SATELLITE_DISTANCE"));
00259     CODA(coda_cursor_read_uint32(&cursor, &iasi_raw->Sat_z[mdr_i]));
00260
00261     /* Read spectral range... */
00262     iasi_raw->IDefSpectDWn1b[mdr_i] = IASI_IDefSpectDWn1b / 100.0;
00263
00264     CODA(coda_cursor_goto_parent(&cursor));
00265     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNsfirst1b"));
00266     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNsfirst1b[mdr_i]));
00267     if (iasi_raw->IDefNsfirst1b[mdr_i] != IASI_IDefNsfirst1b)
00268       ERRMSG("Unexpected value for IDefNsfirst1b!");
00269
00270     CODA(coda_cursor_goto_parent(&cursor));
00271     CODA(coda_cursor_goto_record_field_by_name(&cursor, "IDefNslast1b"));
00272     CODA(coda_cursor_read_int32(&cursor, &iasi_raw->IDefNslast1b[mdr_i]));
00273     if (iasi_raw->IDefNslast1b[mdr_i] != IASI_IDefNslast1b)
00274       ERRMSG("Unexpected value for IDefNslast1b!");
00275
00276     /* Compute wavenumber... */
00277     if (mdr_i == 0)
00278       for (i = 0; i < IASI_L1_NCHAN; i++)
00279         iasi_raw->Wavenumber[i] =
00280           iasi_raw->IDefSpectDWn1b[mdr_i] *
00281           (float) (iasi_raw->IDefNsfirst1b[mdr_i] + i - 1);
00282   }
00283
00284   /* Close file... */
00285   CODA(coda_close(pf));
00286
00287   /* Finalize CODA... */
00288   coda_done();
00289
00290   /* Set number of tracks... */
00291   iasi_rad->ntrack = (int) (iasi_raw->ntrack * 2);
00292
00293   /* Copy wavenumbers... */
00294   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00295     iasi_rad->freq[ichan] = iasi_raw->Wavenumber[ichan];
00296
00297   /* Copy footprint data... */
00298   for (mdr_i = 0; mdr_i < iasi_raw->ntrack; mdr_i++) {
00299     tr1 = mdr_i * 2;
00300     tr2 = mdr_i * 2 + 1;
00301     tr1_lpm = 3;
00302     tr1_rpm = 0;
00303     tr2_lpm = 2;
00304     tr2_rpm = 1;
00305
00306     /* Copy time (2x2 matrix has same measurement time)...  */
00307     for (i = 0; i < IASI_NXTRACK; i++) {
00308       iasi_rad->Time[tr1][i * 2] = iasi_raw->Time[mdr_i][i];
00309       iasi_rad->Time[tr1][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00310       iasi_rad->Time[tr2][i * 2] = iasi_raw->Time[mdr_i][i];
00311       iasi_rad->Time[tr2][i * 2 + 1] = iasi_raw->Time[mdr_i][i];
00312     }
00313
00314     /* Copy location... */
00315     for (i = 0; i < IASI_NXTRACK; i++) {
00316       iasi_rad->Longitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][0];
00317       iasi_rad->Longitude[tr1][i * 2 + 1] =
```

```
00318         iasi_raw->Loc[mdr_i][i][tr1_rpm][0];
00319       iasi_rad->Latitude[tr1][i * 2] = iasi_raw->Loc[mdr_i][i][tr1_lpm][1];
00320       iasi_rad->Latitude[tr1][i * 2 + 1] =
00321         iasi_raw->Loc[mdr_i][i][tr1_rpm][1];
00322
00323       iasi_rad->Longitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][0];
00324       iasi_rad->Longitude[tr2][i * 2 + 1] =
00325         iasi_raw->Loc[mdr_i][i][tr2_rpm][0];
00326       iasi_rad->Latitude[tr2][i * 2] = iasi_raw->Loc[mdr_i][i][tr2_lpm][1];
00327       iasi_rad->Latitude[tr2][i * 2 + 1] =
00328         iasi_raw->Loc[mdr_i][i][tr2_rpm][1];
00329     }
00330
00331     /* Copy satellite location (we only have one height value)... */
00332     iasi_rad->Sat_lon[tr1] = iasi_rad->Longitude[tr1][28];
00333     iasi_rad->Sat_lat[tr1] = iasi_rad->Latitude[tr1][28];
00334     iasi_rad->Sat_lon[tr2] = iasi_rad->Longitude[tr2][28];
00335     iasi_rad->Sat_lat[tr2] = iasi_rad->Latitude[tr2][28];
00336     iasi_rad->Sat_z[tr1] =
00337       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr1]);
00338     iasi_rad->Sat_z[tr2] =
00339       iasi_raw->Sat_z[mdr_i] / 1000.0 - wgs84(iasi_rad->Sat_lat[tr2]);
00340
00341     /* Copy radiation data... */
00342     for (i = 0; i < IASI_NXTRACK; i++) {
00343       for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++) {
00344         sc = scaling[ichan] * 100.0f;
00345         iasi_rad->Rad[tr1][i * 2][ichan] =
00346           iasi_raw->Radiation[mdr_i][i][tr1_lpm][ichan] * sc;
00347         iasi_rad->Rad[tr1][i * 2 + 1][ichan] =
00348           iasi_raw->Radiation[mdr_i][i][tr1_rpm][ichan] * sc;
00349         iasi_rad->Rad[tr2][i * 2][ichan] =
00350           iasi_raw->Radiation[mdr_i][i][tr2_lpm][ichan] * sc;
00351         iasi_rad->Rad[tr2][i * 2 + 1][ichan] =
00352           iasi_raw->Radiation[mdr_i][i][tr2_rpm][ichan] * sc;
00353       }
00354     }
00355   }
00356
00357   /* Check radiance data... */
00358   for (i = 0; i < iasi_rad->ntrack; i++)
00359     for (j = 0; j < L1_NXTRACK; j++)
00360       if (iasi_rad->Rad[i][j][6753] > iasi_rad->Rad[i][j][6757]
00361           || iasi_rad->Rad[i][j][6753] < 0)
00362         for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00363           iasi_rad->Rad[i][j][ichan] = GSL_NAN;
00364
00365   /* Free... */
00366   free(iasi_raw);
00367 }
```

Here is the call graph for this function:

5.11.1.6 void noise (wave_t * *wave*, double * *mu*, double * *sig*)

Estimate noise.

Definition at line 371 of file libiasi.c.

```
00374                 {
00375
00376   int ix, ix2, iy, iy2, n = 0, okay;
```

```
00377
00378   /* Init... */
00379   *mu = 0;
00380   *sig = 0;
00381
00382   /* Estimate noise (Immerkaer, 1996)... */
00383   for (ix = 1; ix < wave->nx - 1; ix++)
00384     for (iy = 1; iy < wave->ny - 1; iy++) {
00385
00386       /* Check data... */
00387       okay = 1;
00388       for (ix2 = ix - 1; ix2 <= ix + 1; ix2++)
00389         for (iy2 = iy - 1; iy2 <= iy + 1; iy2++)
00390           if (!gsl_finite(wave->temp[ix2][iy2]))
00391             okay = 0;
00392       if (!okay)
00393         continue;
00394
00395       /* Get mean noise... */
00396       n++;
00397       *mu += wave->temp[ix][iy];
00398       *sig += gsl_pow_2(+4. / 6. * wave->temp[ix][iy]
00399                         - 2. / 6. * (wave->temp[ix - 1][iy]
00400                                      + wave->temp[ix + 1][iy]
00401                                      + wave->temp[ix][iy - 1]
00402                                      + wave->temp[ix][iy + 1])
00403                         + 1. / 6. * (wave->temp[ix - 1][iy - 1]
00404                                      + wave->temp[ix + 1][iy - 1]
00405                                      + wave->temp[ix - 1][iy + 1]
00406                                      + wave->temp[ix + 1][iy + 1]));
00407     }
00408
00409   /* Normalize... */
00410   *mu /= (double) n;
00411   *sig = sqrt(*sig / (double) n);
00412 }
```

5.11.1.7 void pert2wave (pert_t * *pert*, wave_t * *wave*, int *track0*, int *track1*, int *xtrack0*, int *xtrack1*)

Convert radiance perturbation data to wave analysis struct.

Definition at line 416 of file libiasi.c.

```
00422               {
00423
00424   double x0[3], x1[3];
00425
00426   int itrack, ixtrack;
00427
00428   /* Check ranges... */
00429   track0 = GSL_MIN(GSL_MAX(track0, 0), pert->ntrack - 1);
00430   track1 = GSL_MIN(GSL_MAX(track1, 0), pert->ntrack - 1);
00431   xtrack0 = GSL_MIN(GSL_MAX(xtrack0, 0), pert->nxtrack - 1);
00432   xtrack1 = GSL_MIN(GSL_MAX(xtrack1, 0), pert->nxtrack - 1);
00433
00434   /* Set size... */
00435   wave->nx = xtrack1 - xtrack0 + 1;
00436   if (wave->nx > WX)
00437     ERRMSG("Too many across-track values!");
00438   wave->ny = track1 - track0 + 1;
00439   if (wave->ny > WY)
00440     ERRMSG("Too many along-track values!");
00441
00442   /* Loop over footprints... */
00443   for (itrack = track0; itrack <= track1; itrack++)
00444     for (ixtrack = xtrack0; ixtrack <= xtrack1; ixtrack++) {
00445
00446       /* Get distances... */
00447       if (itrack == track0) {
00448         wave->x[0] = 0;
00449         if (ixtrack > xtrack0) {
00450           geo2cart(0, pert->lon[itrack][ixtrack - 1],
00451                    pert->lat[itrack][ixtrack - 1], x0);
00452           geo2cart(0, pert->lon[itrack][ixtrack],
00453                    pert->lat[itrack][ixtrack], x1);
00454           wave->x[ixtrack - xtrack0] =
00455             wave->x[ixtrack - xtrack0 - 1] + DIST(x0, x1);
00456         }
00457       }
00458       if (ixtrack == xtrack0) {
```

```
00459         wave->y[0] = 0;
00460         if (itrack > track0) {
00461           geo2cart(0, pert->lon[itrack - 1][ixtrack],
00462                    pert->lat[itrack - 1][ixtrack], x0);
00463           geo2cart(0, pert->lon[itrack][ixtrack],
00464                    pert->lat[itrack][ixtrack], x1);
00465           wave->y[itrack - track0] =
00466             wave->y[itrack - track0 - 1] + DIST(x0, x1);
00467         }
00468       }
00469
00470       /* Save geolocation... */
00471       wave->time = pert->time[(track0 + track1) / 2][(xtrack0 + xtrack1) / 2];
00472       wave->z = 0;
00473       wave->lon[ixtrack - xtrack0][itrack - track0] =
00474         pert->lon[itrack][ixtrack];
00475       wave->lat[ixtrack - xtrack0][itrack - track0] =
00476         pert->lat[itrack][ixtrack];
00477
00478       /* Save temperature data... */
00479       wave->temp[ixtrack - xtrack0][itrack - track0]
00480         = pert->bt[itrack][ixtrack];
00481       wave->bg[ixtrack - xtrack0][itrack - track0]
00482         = pert->bt[itrack][ixtrack] - pert->pt[itrack][ixtrack];
00483       wave->pt[ixtrack - xtrack0][itrack - track0]
00484         = pert->pt[itrack][ixtrack];
00485       wave->var[ixtrack - xtrack0][itrack - track0]
00486         = pert->var[itrack][ixtrack];
00487     }
00488 }
```

Here is the call graph for this function:

5.11.1.8 void variance (wave_t * *wave*, double *dh*)

Compute local variance.

Definition at line 492 of file libiasi.c.

```
00494             {
00495
00496   double dh2, mu, help;
00497
00498   int dx, dy, ix, ix2, iy, iy2, n;
00499
00500   /* Check parameters... */
00501   if (dh <= 0)
00502     return;
00503
00504   /* Compute squared radius... */
00505   dh2 = gsl_pow_2(dh);
00506
00507   /* Get sampling distances... */
00508   dx =
00509     (int) (dh / fabs(wave->x[wave->nx - 1] - wave->x[0]) * (wave->nx - 1.0) +
00510            1);
00511   dy =
00512     (int) (dh / fabs(wave->y[wave->ny - 1] - wave->y[0]) * (wave->ny - 1.0) +
00513            1);
00514
00515   /* Loop over data points... */
00516   for (ix = 0; ix < wave->nx; ix++)
```

```
00517     for (iy = 0; iy < wave->ny; iy++) {
00518
00519       /* Init... */
00520       mu = help = 0;
00521       n = 0;
00522
00523       /* Get data... */
00524       for (ix2 = GSL_MAX(ix - dx, 0); ix2 <= GSL_MIN(ix + dx, wave->nx - 1);
00525            ix2++)
00526         for (iy2 = GSL_MAX(iy - dy, 0); iy2 <= GSL_MIN(iy + dy, wave->ny - 1);
00527              iy2++)
00528           if ((gsl_pow_2(wave->x[ix] - wave->x[ix2])
00529                + gsl_pow_2(wave->y[iy] - wave->y[iy2])) <= dh2)
00530             if (gsl_finite(wave->pt[ix2][iy2])) {
00531               mu += wave->pt[ix2][iy2];
00532               help += gsl_pow_2(wave->pt[ix2][iy2]);
00533               n++;
00534             }
00535
00536       /* Compute local variance... */
00537       if (n > 1)
00538         wave->var[ix][iy] = help / n - gsl_pow_2(mu / n);
00539       else
00540         wave->var[ix][iy] = GSL_NAN;
00541     }
00542 }
```

5.11.1.9 double wgs84 (double *lat*)

Calculate Earth radius according to WGS-84 reference ellipsoid.

Definition at line 546 of file libiasi.c.

```
00547              {
00548
00549   const double a = 6378.1370, b = 6356.7523;
00550
00551   double cphi, sphi;
00552
00553   cphi = cos(lat * M_PI / 180.);
00554   sphi = sin(lat * M_PI / 180.);
00555
00556   return sqrt((gsl_pow_2(a * a * cphi) + gsl_pow_2(b * b * sphi))
00557               / (gsl_pow_2(a * cphi) + gsl_pow_2(b * sphi)));
00558 }
```

5.11.1.10 void write_l1 (char * *filename*, iasi_l1_t * *l1*)

Write IASI Level-1 data.

Definition at line 562 of file libiasi.c.

```
00564                  {
00565
00566   int dimid[10], ncid, time_id, lon_id, lat_id,
00567     sat_z_id, sat_lon_id, sat_lat_id, nu_id, rad_id;
00568
00569   /* Open or create netCDF file... */
00570   printf("Write IASI Level-1 file: %s\n", filename);
00571   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00572     NC(nc_create(filename, NC_CLOBBER, &ncid));
00573   } else {
00574     NC(nc_redef(ncid));
00575   }
00576
00577   /* Set dimensions... */
00578   if (nc_inq_dimid(ncid, "L1_NTRACK", &dimid[0]) != NC_NOERR)
00579     NC(nc_def_dim(ncid, "L1_NTRACK", l1->ntrack, &dimid[0]));
00580   if (nc_inq_dimid(ncid, "L1_NXTRACK", &dimid[1]) != NC_NOERR)
00581     NC(nc_def_dim(ncid, "L1_NXTRACK", L1_NXTRACK, &dimid[1]));
00582   if (nc_inq_dimid(ncid, "L1_NCHAN", &dimid[2]) != NC_NOERR)
00583     NC(nc_def_dim(ncid, "L1_NCHAN", L1_NCHAN, &dimid[2]));
00584
00585   /* Add variables... */
```

```
00586   add_var(ncid, "l1_time", "s", "time (seconds since 2000-01-01T00:00Z)",
00587           NC_DOUBLE, dimid, &time_id, 2);
00588   add_var(ncid, "l1_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00589   add_var(ncid, "l1_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00590   add_var(ncid, "l1_sat_z", "km", "satellite altitude",
00591           NC_DOUBLE, dimid, &sat_z_id, 1);
00592   add_var(ncid, "l1_sat_lon", "deg", "(estimated) satellite longitude",
00593           NC_DOUBLE, dimid, &sat_lon_id, 1);
00594   add_var(ncid, "l1_sat_lat", "deg", "(estimated) satellite latitude",
00595           NC_DOUBLE, dimid, &sat_lat_id, 1);
00596   add_var(ncid, "l1_nu", "cm^-1", "channel wavenumber",
00597           NC_DOUBLE, &dimid[2], &nu_id, 1);
00598   add_var(ncid, "l1_rad", "W/(m^2 sr cm^-1)", "channel radiance",
00599           NC_FLOAT, dimid, &rad_id, 3);
00600
00601   /* Leave define mode... */
00602   NC(nc_enddef(ncid));
00603
00604   /* Write data... */
00605   NC(nc_put_var_double(ncid, time_id, l1->time[0]));
00606   NC(nc_put_var_double(ncid, lon_id, l1->lon[0]));
00607   NC(nc_put_var_double(ncid, lat_id, l1->lat[0]));
00608   NC(nc_put_var_double(ncid, sat_z_id, l1->sat_z));
00609   NC(nc_put_var_double(ncid, sat_lon_id, l1->sat_lon));
00610   NC(nc_put_var_double(ncid, sat_lat_id, l1->sat_lat));
00611   NC(nc_put_var_double(ncid, nu_id, l1->nu));
00612   NC(nc_put_var_float(ncid, rad_id, &l1->rad[0][0][0]));
00613
00614   /* Close file... */
00615   NC(nc_close(ncid));
00616 }
```

Here is the call graph for this function:

5.11.1.11 void write_l2 (char * *filename*, iasi_l2_t * *l2*)

Write IASI Level-2 data.

Definition at line 620 of file libiasi.c.

```
00622                   {
00623
00624   int dimid[10], ncid, time_id, z_id, lon_id, lat_id, p_id, t_id;
00625
00626   /* Create netCDF file... */
00627   printf("Write IASI Level-2 file: %s\n", filename);
00628   if (nc_open(filename, NC_WRITE, &ncid) != NC_NOERR) {
00629     NC(nc_create(filename, NC_CLOBBER, &ncid));
00630   } else {
00631     NC(nc_redef(ncid));
00632   }
00633
00634   /* Set dimensions... */
00635   if (nc_inq_dimid(ncid, "L2_NTRACK", &dimid[0]) != NC_NOERR)
00636     NC(nc_def_dim(ncid, "L2_NTRACK", l2->ntrack, &dimid[0]));
00637   if (nc_inq_dimid(ncid, "L2_NXTRACK", &dimid[1]) != NC_NOERR)
00638     NC(nc_def_dim(ncid, "L2_NXTRACK", L2_NXTRACK, &dimid[1]));
00639   if (nc_inq_dimid(ncid, "L2_NLAY", &dimid[2]) != NC_NOERR)
00640     NC(nc_def_dim(ncid, "L2_NLAY", L2_NLAY, &dimid[2]));
00641
00642   /* Add variables... */
00643   add_var(ncid, "l2_time", "s", "time (seconds since 2000-01-01T00:00Z)",
```

```
00644           NC_DOUBLE, dimid, &time_id, 2);
00645   add_var(ncid, "l2_z", "km", "altitude", NC_DOUBLE, dimid, &z_id, 3);
00646   add_var(ncid, "l2_lon", "deg", "longitude", NC_DOUBLE, dimid, &lon_id, 2);
00647   add_var(ncid, "l2_lat", "deg", "latitude", NC_DOUBLE, dimid, &lat_id, 2);
00648   add_var(ncid, "l2_press", "hPa", "pressure",
00649           NC_DOUBLE, &dimid[2], &p_id, 1);
00650   add_var(ncid, "l2_temp", "K", "temperature", NC_DOUBLE, dimid, &t_id, 3);
00651
00652   /* Leave define mode... */
00653   NC(nc_enddef(ncid));
00654
00655   /* Write data... */
00656   NC(nc_put_var_double(ncid, time_id, l2->time[0]));
00657   NC(nc_put_var_double(ncid, z_id, l2->z[0][0]));
00658   NC(nc_put_var_double(ncid, lon_id, l2->lon[0]));
00659   NC(nc_put_var_double(ncid, lat_id, l2->lat[0]));
00660   NC(nc_put_var_double(ncid, p_id, l2->p));
00661   NC(nc_put_var_double(ncid, t_id, l2->t[0][0]));
00662
00663   /* Close file... */
00664   NC(nc_close(ncid));
00665 }
```

Here is the call graph for this function:

5.12 libiasi.h

```
00001 #include <netcdf.h>
00002 #include <gsl/gsl_randist.h>
00003 #include <gsl/gsl_fft_complex.h>
00004 #include <gsl/gsl_multifit.h>
00005 #include <gsl/gsl_poly.h>
00006 #include <gsl/gsl_sort.h>
00007 #include <gsl/gsl_spline.h>
00008 #include "coda.h"
00009 #include "jurassic.h"
00010
00011 /* ------------------------------------------------------------
00012    Dimensions...
00013    ------------------------------------------------------------ */
00014
00016 #define L1_NCHAN 33
00017
00019 #define L1_NTRACK 1800
00020
00022 #define L1_NXTRACK 60
00023
00025 #define L2_NLAY 27
00026
00028 #define L2_NTRACK 1800
00029
00031 #define L2_NXTRACK 60
00032
00034 #define IASI_L1_NCHAN 8700
00035
00037 #define IASI_NXTRACK 30
00038
00040 #define IASI_PM 4
00041
00043 #define IASI_IDefNsfirst1b 2581
00044
00046 #define IASI_IDefNslast1b 11041
00047
00049 #define IASI_IDefSpectDWn1b 25
00050
```

```
00052 #define PERT_NTRACK 132000
00053
00055 #define PERT_NXTRACK 360
00056
00058 #define WX 300
00059
00061 #define WY 33000
00062
00063 /* ------------------------------------------------------------
00064    Macros...
00065    ------------------------------------------------------------ */
00066
00068 #define CODA(cmd) {                              \
00069     if((cmd)!=0)                                 \
00070       ERRMSG(coda_errno_to_string(coda_errno));  \
00071   }
00072
00074 #define NC(cmd) {                                   \
00075     if((cmd)!=NC_NOERR)                             \
00076       ERRMSG(nc_strerror(cmd));                     \
00077   }
00078
00079 /* ------------------------------------------------------------
00080    Structs...
00081    ------------------------------------------------------------ */
00082
00084 typedef struct {
00085
00087   size_t ntrack;
00088
00090   double time[L1_NTRACK][L1_NXTRACK];
00091
00093   double lon[L1_NTRACK][L1_NXTRACK];
00094
00096   double lat[L1_NTRACK][L1_NXTRACK];
00097
00099   double sat_z[L1_NTRACK];
00100
00102   double sat_lon[L1_NTRACK];
00103
00105   double sat_lat[L1_NTRACK];
00106
00108   double nu[L1_NCHAN];
00109
00111   float rad[L1_NTRACK][L1_NXTRACK][L1_NCHAN];
00112
00113 } iasi_l1_t;
00114
00116 typedef struct {
00117
00119   size_t ntrack;
00120
00122   double time[L2_NTRACK][L2_NXTRACK];
00123
00125   double z[L2_NTRACK][L2_NXTRACK][L2_NLAY];
00126
00128   double lon[L2_NTRACK][L2_NXTRACK];
00129
00131   double lat[L2_NTRACK][L2_NXTRACK];
00132
00134   double p[L2_NLAY];
00135
00137   double t[L2_NTRACK][L2_NXTRACK][L2_NLAY];
00138
00139 } iasi_l2_t;
00140
00142 typedef struct {
00143
00145   int ntrack;
00146
00148   int nxtrack;
00149
00151   double time[PERT_NTRACK][PERT_NXTRACK];
00152
00154   double lon[PERT_NTRACK][PERT_NXTRACK];
00155
00157   double lat[PERT_NTRACK][PERT_NXTRACK];
00158
00160   double dc[PERT_NTRACK][PERT_NXTRACK];
00161
00163   double bt[PERT_NTRACK][PERT_NXTRACK];
00164
00166   double pt[PERT_NTRACK][PERT_NXTRACK];
00167
00169   double var[PERT_NTRACK][PERT_NXTRACK];
00170
00171 } pert_t;
```

```
00172
00174 typedef struct {
00175
00177   long ntrack;
00178
00180   float IDefSpectDWn1b[L1_NTRACK];
00181
00183   int32_t IDefNsfirst1b[L1_NTRACK];
00184
00186   int32_t IDefNslast1b[L1_NTRACK];
00187
00189   double Time[L1_NTRACK][IASI_NXTRACK];
00190
00192   double Loc[L1_NTRACK][IASI_NXTRACK][IASI_PM][2];
00193
00195   float Wavenumber[IASI_L1_NCHAN];
00196
00198   short int Radiation[L1_NTRACK][IASI_NXTRACK][IASI_PM][IASI_L1_NCHAN];
00199
00201   unsigned int Sat_z[L1_NTRACK];
00202
00203 } iasi_raw_t;
00204
00206 typedef struct {
00207
00209   int ntrack;
00210
00212   double freq[IASI_L1_NCHAN];
00213
00215   double Time[L1_NTRACK][L1_NXTRACK];
00216
00218   double Longitude[L1_NTRACK][L1_NXTRACK];
00219
00221   double Latitude[L1_NTRACK][L1_NXTRACK];
00222
00224   float Rad[L1_NTRACK][L1_NXTRACK][IASI_L1_NCHAN];
00225
00227   double Sat_z[L1_NTRACK];
00228
00230   double Sat_lon[L1_NTRACK];
00231
00233   double Sat_lat[L1_NTRACK];
00234
00235 } iasi_rad_t;
00236
00238 typedef struct {
00239
00241   int nx;
00242
00244   int ny;
00245
00247   double time;
00248
00250   double z;
00251
00253   double lon[WX][WY];
00254
00256   double lat[WX][WY];
00257
00259   double x[WX];
00260
00262   double y[WY];
00263
00265   double temp[WX][WY];
00266
00268   double bg[WX][WY];
00269
00271   double pt[WX][WY];
00272
00274   double var[WX][WY];
00275
00276 } wave_t;
00277
00278 /* ------------------------------------------------------------
00279    Functions...
00280    ------------------------------------------------------------ */
00281
00283 void add_var(
00284   int ncid,
00285   const char *varname,
00286   const char *unit,
00287   const char *longname,
00288   int type,
00289   int dimid[],
00290   int *varid,
00291   int ndims);
00292
```

```
00294 void background_poly(
00295   wave_t * wave,
00296   int dim_x,
00297   int dim_y);
00298
00300 void background_poly_help(
00301   double *xx,
00302   double *yy,
00303   int n,
00304   int dim);
00305
00307 int get_chan_for_wavenumber(
00308   float wavenumber);
00309
00311 void iasi_read(
00312   char *filename,
00313   iasi_rad_t * iasi_rad);
00314
00316 void noise(
00317   wave_t * wave,
00318   double *mu,
00319   double *sig);
00320
00322 void pert2wave(
00323   pert_t * pert,
00324   wave_t * wave,
00325   int track0,
00326   int track1,
00327   int xtrack0,
00328   int xtrack1);
00329
00331 void variance(
00332   wave_t * wave,
00333   double dh);
00334
00336 double wgs84(
00337   double lat);
00338
00340 void write_l1(
00341   char *filename,
00342   iasi_l1_t * l1);
00343
00345 void write_l2(
00346   char *filename,
00347   iasi_l2_t * l2);
```

5.13 noise.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

5.13.1 Function Documentation

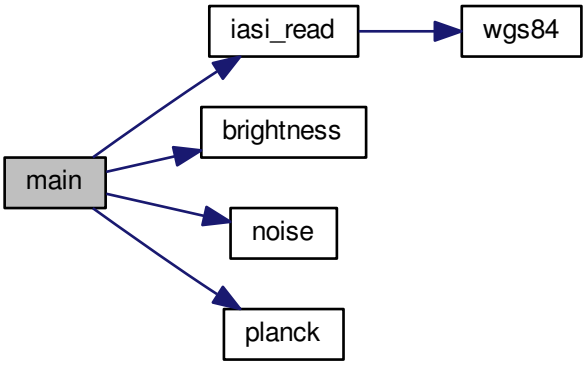
5.13.1.1 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 3 of file noise.c.

```
00005                 {
00006
00007   static iasi_rad_t *iasi_rad;
00008
00009   static wave_t wave;
00010
00011   static FILE *out;
00012
00013   static double mu, nesr, sigma;
00014
00015   static int ichan, itrack, ix, iy;
00016
00017   /* Check arguments... */
00018   if (argc < 4)
00019     ERRMSG("Give parameters: <ctl> <iasi_l1_file> <noise.tab>");
00020
```

```
00021   /* Allocate... */
00022   ALLOC(iasi_rad, iasi_rad_t, 1);
00023
00024   /* Read IASI data... */
00025   printf("Read IASI data: %s\n", argv[2]);
00026   iasi_read(argv[2], iasi_rad);
00027
00028   /* Create file... */
00029   printf("Write noise data: %s\n", argv[3]);
00030   if (!(out = fopen(argv[3], "w")))
00031     ERRMSG("Cannot create file!");
00032
00033   /* Write header... */
00034   fprintf(out,
00035           "# $1 = track index\n"
00036           "# $2 = channel index\n"
00037           "# $3 = wavenumber [1/cm]\n"
00038           "# $4 = mean BT [K]\n"
00039           "# $5 = NEDT [K]\n"
00040           "# $6 = NESR [W/(m^2 sr cm^-1)]\n");
00041
00042   /* Analyze blocks of data... */
00043   for (itrack = 0; itrack < iasi_rad->ntrack; itrack += 60) {
00044
00045     /* Write empty line... */
00046     fprintf(out, "\n");
00047
00048     /* Loop over channels... */
00049     for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++) {
00050
00051       /* Set wave struct... */
00052       wave.nx = L1_NXTRACK;
00053       wave.ny = 0;
00054       for (iy = itrack; iy < GSL_MIN(itrack + 60, iasi_rad->ntrack); iy++) {
00055         for (ix = 0; ix < wave.nx; ix++)
00056           wave.temp[ix][wave.ny] = brightness(iasi_rad->Rad[iy][ix][ichan],
00057                                               iasi_rad->freq[ichan]);
00058         wave.ny++;
00059       }
00060
00061       /* Check number of data points... */
00062       if (wave.ny >= 55) {
00063
00064         /* Get noise... */
00065         noise(&wave, &mu, &sigma);
00066
00067         /* Get NESR... */
00068         nesr=planck(mu+sigma, iasi_rad->freq[ichan])
00069           -planck(mu, iasi_rad->freq[ichan]);
00070
00071         /* Write output... */
00072         if (gsl_finite(sigma))
00073           fprintf(out, "%d %d %.4f %g %g %g\n", itrack, ichan,
00074                   iasi_rad->freq[ichan], mu, sigma, nesr);
00075       }
00076     }
00077   }
00078
00079   /* Close file... */
00080   fclose(out);
00081
00082   /* Free... */
00083   free(iasi_rad);
00084
00085   return EXIT_SUCCESS;
00086 }
```

Here is the call graph for this function:



5.14 noise.c

```
00001 #include "libiasi.h"
00002
00003 int main(
00004   int argc,
00005   char *argv[]) {
00006
00007   static iasi_rad_t *iasi_rad;
00008
00009   static wave_t wave;
00010
00011   static FILE *out;
00012
00013   static double mu, nesr, sigma;
00014
00015   static int ichan, itrack, ix, iy;
00016
00017   /* Check arguments... */
00018   if (argc < 4)
00019     ERRMSG("Give parameters: <ctl> <iasi_l1_file> <noise.tab>");
00020
00021   /* Allocate... */
00022   ALLOC(iasi_rad, iasi_rad_t, 1);
00023
00024   /* Read IASI data... */
00025   printf("Read IASI data: %s\n", argv[2]);
00026   iasi_read(argv[2], iasi_rad);
00027
00028   /* Create file... */
00029   printf("Write noise data: %s\n", argv[3]);
00030   if (!(out = fopen(argv[3], "w")))
00031     ERRMSG("Cannot create file!");
00032
00033   /* Write header... */
00034   fprintf(out,
00035           "# $1 = track index\n"
00036           "# $2 = channel index\n"
00037           "# $3 = wavenumber [1/cm]\n"
00038           "# $4 = mean BT [K]\n"
00039           "# $5 = NEDT [K]\n"
00040           "# $6 = NESR [W/(m^2 sr cm^-1)]\n");
00041
00042   /* Analyze blocks of data... */
00043   for (itrack = 0; itrack < iasi_rad->ntrack; itrack += 60) {
00044
00045     /* Write empty line... */
00046     fprintf(out, "\n");
00047
00048     /* Loop over channels... */
00049     for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++) {
```

```
00050
00051       /* Set wave struct... */
00052       wave.nx = L1_NXTRACK;
00053       wave.ny = 0;
00054       for (iy = itrack; iy < GSL_MIN(itrack + 60, iasi_rad->ntrack); iy++) {
00055         for (ix = 0; ix < wave.nx; ix++)
00056           wave.temp[ix][wave.ny] = brightness(iasi_rad->Rad[iy][ix][ichan],
00057                                               iasi_rad->freq[ichan]);
00058         wave.ny++;
00059       }
00060
00061       /* Check number of data points... */
00062       if (wave.ny >= 55) {
00063
00064         /* Get noise... */
00065         noise(&wave, &mu, &sigma);
00066
00067         /* Get NESR... */
00068         nesr=planck(mu+sigma, iasi_rad->freq[ichan])
00069           -planck(mu, iasi_rad->freq[ichan]);
00070
00071         /* Write output... */
00072         if (gsl_finite(sigma))
00073           fprintf(out, "%d %d %.4f %g %g %g\n", itrack, ichan,
00074                   iasi_rad->freq[ichan], mu, sigma, nesr);
00075       }
00076     }
00077   }
00078
00079   /* Close file... */
00080   fclose(out);
00081
00082   /* Free... */
00083   free(iasi_rad);
00084
00085   return EXIT_SUCCESS;
00086 }
```

5.15 perturbation.c File Reference

Functions

- void addatt (int ncid, int varid, const char *unit, const char *long_name)
- int main (int argc, char *argv[])

5.15.1 Function Documentation

5.15.1.1 void addatt (int *ncid*, int *varid*, const char * *unit*, const char * *long_name*)

Definition at line 384 of file perturbation.c.

```
00388                           {
00389
00390   /* Set long name... */
00391   NC(nc_put_att_text(ncid, varid, "long_name", strlen(long_name), long_name));
00392
00393   /* Set units... */
00394   NC(nc_put_att_text(ncid, varid, "units", strlen(unit), unit));
00395 }
```

5.15.1.2 int main (int *argc*, char * *argv[]*)

Definition at line 31 of file perturbation.c.

```
00033                {
00034
00035   static iasi_rad_t *iasi_rad;
00036
00037   static pert_t *pert_4mu, *pert_15mu_low, *pert_15mu_high;
00038
00039   static wave_t wave;
00040
00041   static double numean, radmean, var_dh = 100.;
00042
00043   static int list_4mu[N4]
00044     = { 6711, 6712, 6713, 6714, 6715, 6716, 6717, 6718, 6719, 6720,
00045     6721, 6722, 6723, 6724, 6725, 6726, 6727, 6728, 6729, 6730, 6731,
00046     6732, 6733, 6734, 6735, 6736, 6737, 6738, 6739, 6740, 6741, 6742,
00047     6743, 6744, 6745, 6746, 6747, 6748, 6749, 6750, 6751, 6752, 6753,
00048     6754, 6755, 6756, 6757, 6758, 6759, 6760, 6761, 6762, 6763, 6764,
00049     6765, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775,
00050     6776, 6777, 6778, 6779, 6780, 6781, 6782, 6783, 6784, 6785, 6786,
00051     6787, 6788, 6789, 6790, 6791, 6792, 6793, 6794, 6795, 6796, 6797,
00052     6798, 6799, 6800, 6801, 6802, 6803, 6804, 6830, 6831, 6832, 6833,
00053     6834, 6835, 6836, 6837, 6838, 6839, 6840, 6841, 6842, 6843, 6844,
00054     6845, 6846, 6847, 6848, 6849, 6850, 6851, 6852, 6853, 6854, 6855,
00055     6856, 6857, 6858, 6859, 6860, 6861, 6862, 6863, 6864, 6865, 6866,
00056     6867, 6868, 6869, 6870, 6871, 6872, 6873, 6874, 6875, 6876, 6877,
00057     6878, 6879, 6880, 6881, 6882, 6883, 6884, 6885, 6886, 6887
00058   };
00059
00060   static int list_15mu_low[N15_LOW]
00061     = { 22, 28, 34, 40, 46, 52, 58, 72, 100, 105, 112, 118, 119,
00062     124, 125, 130, 131, 136, 137, 143, 144
00063   };
00064
00065   static int list_15mu_high[N15_HIGH]
00066   = { 91, 92 };
00067
00068   static int ix, iy, dimid[2], i, n, ncid, track, track0, xtrack,
00069     time_varid, lon_varid, lat_varid, bt_4mu_varid, bt_4mu_pt_varid,
00070     bt_4mu_var_varid, bt_8mu_varid, bt_15mu_low_varid, bt_15mu_low_pt_varid,
00071     bt_15mu_low_var_varid, bt_15mu_high_varid, bt_15mu_high_pt_varid,
00072     bt_15mu_high_var_varid, iarg;
00073
00074   static size_t start[2], count[2];
00075
00076   /* Check arguments... */
00077   if (argc < 3)
00078     ERRMSG("Give parameters: <out.nc> <l1b_file1> [<l1b_file2> ...]");
00079
00080   /* Allocate... */
00081   ALLOC(iasi_rad, iasi_rad_t, 1);
00082   ALLOC(pert_4mu, pert_t, 1);
00083   ALLOC(pert_15mu_low, pert_t, 1);
00084   ALLOC(pert_15mu_high, pert_t, 1);
00085
00086   /* ------------------------------------------------------------
00087      Read HDF files...
00088      ------------------------------------------------------------ */
00089
00090   /* Loop over HDF files... */
00091   for (iarg = 2; iarg < argc; iarg++) {
00092
00093     /* Read IASI data... */
00094     printf("Read IASI Level-1C data file: %s\n", argv[iarg]);
00095     iasi_read(argv[iarg], iasi_rad);
00096
00097     /* Save geolocation... */
00098     pert_4mu->ntrack += iasi_rad->ntrack;
00099     if (pert_4mu->ntrack > PERT_NTRACK)
00100       ERRMSG("Too many granules!");
00101     pert_4mu->nxtrack = L1_NXTRACK;
00102     if (pert_4mu->nxtrack > PERT_NXTRACK)
00103       ERRMSG("Too many tracks!");
00104     for (track = 0; track < iasi_rad->ntrack; track++)
00105       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00106         pert_4mu->time[track0 + track][xtrack]
00107           = iasi_rad->Time[track][xtrack];
00108         pert_4mu->lon[track0 + track][xtrack]
00109           = iasi_rad->Longitude[track][xtrack];
00110         pert_4mu->lat[track0 + track][xtrack]
00111           = iasi_rad->Latitude[track][xtrack];
00112       }
```

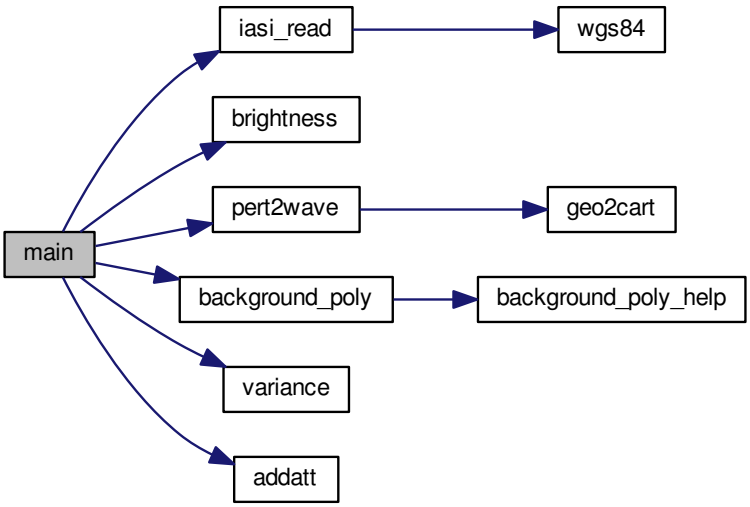
```
00113
00114     pert_15mu_low->ntrack += iasi_rad->ntrack;
00115     if (pert_15mu_low->ntrack > PERT_NTRACK)
00116       ERRMSG("Too many granules!");
00117     pert_15mu_low->nxtrack = L1_NXTRACK;
00118     if (pert_15mu_low->nxtrack > PERT_NXTRACK)
00119       ERRMSG("Too many tracks!");
00120     for (track = 0; track < iasi_rad->ntrack; track++)
00121       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00122         pert_15mu_low->time[track0 + track][xtrack]
00123           = iasi_rad->Time[track][xtrack];
00124         pert_15mu_low->lon[track0 + track][xtrack]
00125           = iasi_rad->Longitude[track][xtrack];
00126         pert_15mu_low->lat[track0 + track][xtrack]
00127           = iasi_rad->Latitude[track][xtrack];
00128       }
00129
00130     pert_15mu_high->ntrack += iasi_rad->ntrack;
00131     if (pert_15mu_high->ntrack > PERT_NTRACK)
00132       ERRMSG("Too many granules!");
00133     pert_15mu_high->nxtrack = L1_NXTRACK;
00134     if (pert_15mu_high->nxtrack > PERT_NXTRACK)
00135       ERRMSG("Too many tracks!");
00136     for (track = 0; track < iasi_rad->ntrack; track++)
00137       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00138         pert_15mu_high->time[track0 + track][xtrack]
00139           = iasi_rad->Time[track][xtrack];
00140         pert_15mu_high->lon[track0 + track][xtrack]
00141           = iasi_rad->Longitude[track][xtrack];
00142         pert_15mu_high->lat[track0 + track][xtrack]
00143           = iasi_rad->Latitude[track][xtrack];
00144       }
00145
00146     /* Get 8.1 micron brightness temperature... */
00147     for (track = 0; track < iasi_rad->ntrack; track++)
00148       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++)
00149         pert_4mu->dc[track0 + track][xtrack]
00150           = brightness(iasi_rad->Rad[track][xtrack][2345],
00151                        iasi_rad->freq[2345]);
00152
00153     /* Get 4.3 micron brightness temperature... */
00154     for (track = 0; track < iasi_rad->ntrack; track++)
00155       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00156         n = 0;
00157         numean = radmean = 0;
00158         for (i = 0; i < N4; i++)
00159           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_4mu[i]])) {
00160             radmean += iasi_rad->Rad[track][xtrack][list_4mu[i]];
00161             numean += iasi_rad->freq[list_4mu[i]];
00162             n++;
00163           }
00164         if (n > 0.9 * N4)
00165           pert_4mu->bt[track0 + track][xtrack]
00166             = brightness(radmean / n, numean / n);
00167         else
00168           pert_4mu->bt[track0 + track][xtrack] = GSL_NAN;
00169       }
00170
00171     /* Get 15 micron brightness temperature (low altitudes)... */
00172     for (track = 0; track < iasi_rad->ntrack; track++)
00173       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00174         n = 0;
00175         numean = radmean = 0;
00176         for (i = 0; i < N15_LOW; i++)
00177           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_15mu_low[i]])) {
00178             radmean += iasi_rad->Rad[track][xtrack][list_15mu_low[i]];
00179             numean += iasi_rad->freq[list_15mu_low[i]];
00180             n++;
00181           }
00182         if (n > 0.9 * N15_LOW)
00183           pert_15mu_low->bt[track0 + track][xtrack]
00184             = brightness(radmean / n, numean / n);
00185         else
00186           pert_15mu_low->bt[track0 + track][xtrack] = GSL_NAN;
00187       }
00188
00189     /* Get 15 micron brightness temperature (high altitudes)... */
00190     for (track = 0; track < iasi_rad->ntrack; track++)
00191       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00192         n = 0;
00193         numean = radmean = 0;
00194         for (i = 0; i < N15_HIGH; i++)
00195           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_15mu_high[i]])) {
00196             radmean += iasi_rad->Rad[track][xtrack][list_15mu_high[i]];
00197             numean += iasi_rad->freq[list_15mu_high[i]];
00198             n++;
00199           }
```

```
00200         if (n > 0.9 * N15_HIGH)
00201           pert_15mu_high->bt[track0 + track][xtrack]
00202             = brightness(radmean / n, numean / n);
00203         else
00204           pert_15mu_high->bt[track0 + track][xtrack] = GSL_NAN;
00205       }
00206
00207     /* Increment track counter... */
00208     track0 += iasi_rad->ntrack;
00209   }
00210
00211   /* ------------------------------------------------------------
00212      Calculate perturbations and variances...
00213      ------------------------------------------------------------ */
00214
00215   /* Convert to wave analysis struct... */
00216   pert2wave(pert_4mu, &wave,
00217             0, pert_4mu->ntrack - 1, 0, pert_4mu->nxtrack - 1);
00218
00219   /* Estimate background... */
00220   background_poly(&wave, 5, 0);
00221
00222   /* Compute variance... */
00223   variance(&wave, var_dh);
00224
00225   /* Copy data... */
00226   for (ix = 0; ix < wave.nx; ix++)
00227     for (iy = 0; iy < wave.ny; iy++) {
00228       pert_4mu->pt[iy][ix] = wave.pt[ix][iy];
00229       pert_4mu->var[iy][ix] = wave.var[ix][iy];
00230     }
00231
00232   /* Convert to wave analysis struct... */
00233   pert2wave(pert_15mu_low, &wave,
00234             0, pert_15mu_low->ntrack - 1, 0, pert_15mu_low->nxtrack - 1);
00235
00236   /* Estimate background... */
00237   background_poly(&wave, 5, 0);
00238
00239   /* Compute variance... */
00240   variance(&wave, var_dh);
00241
00242   /* Copy data... */
00243   for (ix = 0; ix < wave.nx; ix++)
00244     for (iy = 0; iy < wave.ny; iy++) {
00245       pert_15mu_low->pt[iy][ix] = wave.pt[ix][iy];
00246       pert_15mu_low->var[iy][ix] = wave.var[ix][iy];
00247     }
00248
00249   /* Convert to wave analysis struct... */
00250   pert2wave(pert_15mu_high, &wave,
00251             0, pert_15mu_high->ntrack - 1, 0, pert_15mu_high->nxtrack - 1);
00252
00253   /* Estimate background... */
00254   background_poly(&wave, 5, 0);
00255
00256   /* Compute variance... */
00257   variance(&wave, var_dh);
00258
00259   /* Copy data... */
00260   for (ix = 0; ix < wave.nx; ix++)
00261     for (iy = 0; iy < wave.ny; iy++) {
00262       pert_15mu_high->pt[iy][ix] = wave.pt[ix][iy];
00263       pert_15mu_high->var[iy][ix] = wave.var[ix][iy];
00264     }
00265
00266   /* ------------------------------------------------------------
00267      Write to netCDF file...
00268      ------------------------------------------------------------ */
00269
00270   /* Create netCDF file... */
00271   printf("Write perturbation data file: %s\n", argv[1]);
00272   NC(nc_create(argv[1], NC_CLOBBER, &ncid));
00273
00274   /* Set dimensions... */
00275   NC(nc_def_dim(ncid, "NTRACK", NC_UNLIMITED, &dimid[0]));
00276   NC(nc_def_dim(ncid, "NXTRACK", L1_NXTRACK, &dimid[1]));
00277
00278   /* Add variables... */
00279   NC(nc_def_var(ncid, "time", NC_DOUBLE, 2, dimid, &time_varid));
00280   addatt(ncid, time_varid, "s", "time (seconds since 2000-01-01T00:00Z)");
00281   NC(nc_def_var(ncid, "lon", NC_DOUBLE, 2, dimid, &lon_varid));
00282   addatt(ncid, lon_varid, "deg", "footprint longitude");
00283   NC(nc_def_var(ncid, "lat", NC_DOUBLE, 2, dimid, &lat_varid));
00284   addatt(ncid, lat_varid, "deg", "footprint latitude");
00285
00286   NC(nc_def_var(ncid, "bt_8mu", NC_FLOAT, 2, dimid, &bt_8mu_varid));
```

```
00287   addatt(ncid, bt_8mu_varid, "K", "brightness temperature at 8.1 micron");
00288
00289   NC(nc_def_var(ncid, "bt_4mu", NC_FLOAT, 2, dimid, &bt_4mu_varid));
00290   addatt(ncid, bt_4mu_varid, "K", "brightness temperature" " at 4.3 micron");
00291   NC(nc_def_var(ncid, "bt_4mu_pt", NC_FLOAT, 2, dimid, &bt_4mu_pt_varid));
00292   addatt(ncid, bt_4mu_pt_varid, "K", "brightness temperature perturbation"
00293          " at 4.3 micron");
00294   NC(nc_def_var(ncid, "bt_4mu_var", NC_FLOAT, 2, dimid, &bt_4mu_var_varid));
00295   addatt(ncid, bt_4mu_var_varid, "K^2", "brightness temperature variance"
00296          " at 4.3 micron");
00297
00298   NC(nc_def_var(ncid, "bt_15mu_low", NC_FLOAT, 2, dimid, &bt_15mu_low_varid));
00299   addatt(ncid, bt_15mu_low_varid, "K", "brightness temperature"
00300          " at 15 micron (low altitudes)");
00301   NC(nc_def_var(ncid, "bt_15mu_low_pt", NC_FLOAT, 2, dimid,
00302                 &bt_15mu_low_pt_varid));
00303   addatt(ncid, bt_15mu_low_pt_varid, "K",
00304          "brightness temperature perturbation"
00305          " at 15 micron (low altitudes)");
00306   NC(nc_def_var
00307      (ncid, "bt_15mu_low_var", NC_FLOAT, 2, dimid, &bt_15mu_low_var_varid));
00308   addatt(ncid, bt_15mu_low_var_varid, "K^2",
00309          "brightness temperature variance" " at 15 micron (low altitudes)");
00310
00311   NC(nc_def_var(ncid, "bt_15mu_high", NC_FLOAT, 2, dimid,
00312                 &bt_15mu_high_varid));
00313   addatt(ncid, bt_15mu_high_varid, "K", "brightness temperature"
00314          " at 15 micron (high altitudes)");
00315   NC(nc_def_var(ncid, "bt_15mu_high_pt", NC_FLOAT, 2, dimid,
00316                 &bt_15mu_high_pt_varid));
00317   addatt(ncid, bt_15mu_high_pt_varid, "K",
00318          "brightness temperature perturbation"
00319          " at 15 micron (high altitudes)");
00320   NC(nc_def_var
00321      (ncid, "bt_15mu_high_var", NC_FLOAT, 2, dimid, &bt_15mu_high_var_varid));
00322   addatt(ncid, bt_15mu_high_var_varid, "K^2",
00323          "brightness temperature variance" " at 15 micron (high altitudes)");
00324
00325   /* Leave define mode... */
00326   NC(nc_enddef(ncid));
00327
00328   /* Loop over tracks... */
00329   for (track = 0; track < pert_4mu->ntrack; track++) {
00330
00331     /* Set array sizes... */
00332     start[0] = (size_t) track;
00333     start[1] = 0;
00334     count[0] = 1;
00335     count[1] = (size_t) pert_4mu->nxtrack;
00336
00337     /* Write data... */
00338     NC(nc_put_vara_double(ncid, time_varid, start, count,
00339                           pert_4mu->time[track]));
00340     NC(nc_put_vara_double(ncid, lon_varid, start, count,
00341                           pert_4mu->lon[track]));
00342     NC(nc_put_vara_double(ncid, lat_varid, start, count,
00343                           pert_4mu->lat[track]));
00344
00345     NC(nc_put_vara_double(ncid, bt_8mu_varid, start, count,
00346                           pert_4mu->dc[track]));
00347
00348     NC(nc_put_vara_double(ncid, bt_4mu_varid, start, count,
00349                           pert_4mu->bt[track]));
00350     NC(nc_put_vara_double(ncid, bt_4mu_pt_varid, start, count,
00351                           pert_4mu->pt[track]));
00352     NC(nc_put_vara_double(ncid, bt_4mu_var_varid, start, count,
00353                           pert_4mu->var[track]));
00354
00355     NC(nc_put_vara_double(ncid, bt_15mu_low_varid, start, count,
00356                           pert_15mu_low->bt[track]));
00357     NC(nc_put_vara_double(ncid, bt_15mu_low_pt_varid, start, count,
00358                           pert_15mu_low->pt[track]));
00359     NC(nc_put_vara_double(ncid, bt_15mu_low_var_varid, start, count,
00360                           pert_15mu_low->var[track]));
00361
00362     NC(nc_put_vara_double(ncid, bt_15mu_high_varid, start, count,
00363                           pert_15mu_high->bt[track]));
00364     NC(nc_put_vara_double(ncid, bt_15mu_high_pt_varid, start, count,
00365                           pert_15mu_high->pt[track]));
00366     NC(nc_put_vara_double(ncid, bt_15mu_high_var_varid, start, count,
00367                           pert_15mu_high->var[track]));
00368   }
00369
00370   /* Close file... */
00371   NC(nc_close(ncid));
00372
00373   /* Free... */
```

```
00374   free(iasi_rad);
00375   free(pert_4mu);
00376   free(pert_15mu_low);
00377   free(pert_15mu_high);
00378
00379   return EXIT_SUCCESS;
00380 }
```

Here is the call graph for this function:



5.16 perturbation.c

```
00001 #include "libiasi.h"
00002
00003 /* ------------------------------------------------------------
00004    Constants...
00005    ------------------------------------------------------------ */
00006
00007 /* Number of 4 micron channels: */
00008 #define N4 152
00009
00010 /* Number of 15 micron channels (low altitudes): */
00011 #define N15_LOW 21
00012
00013 /* Number of 15 micron channels (high altitudes): */
00014 #define N15_HIGH 2
00015
00016 /* ------------------------------------------------------------
00017    Functions...
00018    ------------------------------------------------------------ */
00019
00020 /* Add variable defintions to netCDF file. */
00021 void addatt(
00022   int ncid,
00023   int varid,
00024   const char *unit,
00025   const char *long_name);
00026
00027 /* ------------------------------------------------------------
00028    Main...
00029    ------------------------------------------------------------ */
00030
00031 int main(
00032   int argc,
00033   char *argv[]) {
```

```
00034
00035   static iasi_rad_t *iasi_rad;
00036
00037   static pert_t *pert_4mu, *pert_15mu_low, *pert_15mu_high;
00038
00039   static wave_t wave;
00040
00041   static double numean, radmean, var_dh = 100.;
00042
00043   static int list_4mu[N4]
00044     = { 6711, 6712, 6713, 6714, 6715, 6716, 6717, 6718, 6719, 6720,
00045     6721, 6722, 6723, 6724, 6725, 6726, 6727, 6728, 6729, 6730, 6731,
00046     6732, 6733, 6734, 6735, 6736, 6737, 6738, 6739, 6740, 6741, 6742,
00047     6743, 6744, 6745, 6746, 6747, 6748, 6749, 6750, 6751, 6752, 6753,
00048     6754, 6755, 6756, 6757, 6758, 6759, 6760, 6761, 6762, 6763, 6764,
00049     6765, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775,
00050     6776, 6777, 6778, 6779, 6780, 6781, 6782, 6783, 6784, 6785, 6786,
00051     6787, 6788, 6789, 6790, 6791, 6792, 6793, 6794, 6795, 6796, 6797,
00052     6798, 6799, 6800, 6801, 6802, 6803, 6804, 6830, 6831, 6832, 6833,
00053     6834, 6835, 6836, 6837, 6838, 6839, 6840, 6841, 6842, 6843, 6844,
00054     6845, 6846, 6847, 6848, 6849, 6850, 6851, 6852, 6853, 6854, 6855,
00055     6856, 6857, 6858, 6859, 6860, 6861, 6862, 6863, 6864, 6865, 6866,
00056     6867, 6868, 6869, 6870, 6871, 6872, 6873, 6874, 6875, 6876, 6877,
00057     6878, 6879, 6880, 6881, 6882, 6883, 6884, 6885, 6886, 6887
00058   };
00059
00060   static int list_15mu_low[N15_LOW]
00061     = { 22, 28, 34, 40, 46, 52, 58, 72, 100, 105, 112, 118, 119,
00062     124, 125, 130, 131, 136, 137, 143, 144
00063   };
00064
00065   static int list_15mu_high[N15_HIGH]
00066   = { 91, 92 };
00067
00068   static int ix, iy, dimid[2], i, n, ncid, track, track0, xtrack,
00069     time_varid, lon_varid, lat_varid, bt_4mu_varid, bt_4mu_pt_varid,
00070     bt_4mu_var_varid, bt_8mu_varid, bt_15mu_low_varid, bt_15mu_low_pt_varid,
00071     bt_15mu_low_var_varid, bt_15mu_high_varid, bt_15mu_high_pt_varid,
00072     bt_15mu_high_var_varid, iarg;
00073
00074   static size_t start[2], count[2];
00075
00076   /* Check arguments... */
00077   if (argc < 3)
00078     ERRMSG("Give parameters: <out.nc> <l1b_file1> [<l1b_file2> ...]");
00079
00080   /* Allocate... */
00081   ALLOC(iasi_rad, iasi_rad_t, 1);
00082   ALLOC(pert_4mu, pert_t, 1);
00083   ALLOC(pert_15mu_low, pert_t, 1);
00084   ALLOC(pert_15mu_high, pert_t, 1);
00085
00086   /* ------------------------------------------------------------
00087      Read HDF files...
00088      ------------------------------------------------------------ */
00089
00090   /* Loop over HDF files... */
00091   for (iarg = 2; iarg < argc; iarg++) {
00092
00093     /* Read IASI data... */
00094     printf("Read IASI Level-1C data file: %s\n", argv[iarg]);
00095     iasi_read(argv[iarg], iasi_rad);
00096
00097     /* Save geolocation... */
00098     pert_4mu->ntrack += iasi_rad->ntrack;
00099     if (pert_4mu->ntrack > PERT_NTRACK)
00100       ERRMSG("Too many granules!");
00101     pert_4mu->nxtrack = L1_NXTRACK;
00102     if (pert_4mu->nxtrack > PERT_NXTRACK)
00103       ERRMSG("Too many tracks!");
00104     for (track = 0; track < iasi_rad->ntrack; track++)
00105       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00106         pert_4mu->time[track0 + track][xtrack]
00107           = iasi_rad->Time[track][xtrack];
00108         pert_4mu->lon[track0 + track][xtrack]
00109           = iasi_rad->Longitude[track][xtrack];
00110         pert_4mu->lat[track0 + track][xtrack]
00111           = iasi_rad->Latitude[track][xtrack];
00112       }
00113
00114     pert_15mu_low->ntrack += iasi_rad->ntrack;
00115     if (pert_15mu_low->ntrack > PERT_NTRACK)
00116       ERRMSG("Too many granules!");
00117     pert_15mu_low->nxtrack = L1_NXTRACK;
00118     if (pert_15mu_low->nxtrack > PERT_NXTRACK)
00119       ERRMSG("Too many tracks!");
00120     for (track = 0; track < iasi_rad->ntrack; track++)
```

```
00121       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00122         pert_15mu_low->time[track0 + track][xtrack]
00123           = iasi_rad->Time[track][xtrack];
00124         pert_15mu_low->lon[track0 + track][xtrack]
00125           = iasi_rad->Longitude[track][xtrack];
00126         pert_15mu_low->lat[track0 + track][xtrack]
00127           = iasi_rad->Latitude[track][xtrack];
00128       }
00129
00130     pert_15mu_high->ntrack += iasi_rad->ntrack;
00131     if (pert_15mu_high->ntrack > PERT_NTRACK)
00132       ERRMSG("Too many granules!");
00133     pert_15mu_high->nxtrack = L1_NXTRACK;
00134     if (pert_15mu_high->nxtrack > PERT_NXTRACK)
00135       ERRMSG("Too many tracks!");
00136     for (track = 0; track < iasi_rad->ntrack; track++)
00137       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00138         pert_15mu_high->time[track0 + track][xtrack]
00139           = iasi_rad->Time[track][xtrack];
00140         pert_15mu_high->lon[track0 + track][xtrack]
00141           = iasi_rad->Longitude[track][xtrack];
00142         pert_15mu_high->lat[track0 + track][xtrack]
00143           = iasi_rad->Latitude[track][xtrack];
00144       }
00145
00146     /* Get 8.1 micron brightness temperature... */
00147     for (track = 0; track < iasi_rad->ntrack; track++)
00148       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++)
00149         pert_4mu->dc[track0 + track][xtrack]
00150           = brightness(iasi_rad->Rad[track][xtrack][2345],
00151                        iasi_rad->freq[2345]);
00152
00153     /* Get 4.3 micron brightness temperature... */
00154     for (track = 0; track < iasi_rad->ntrack; track++)
00155       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00156         n = 0;
00157         numean = radmean = 0;
00158         for (i = 0; i < N4; i++)
00159           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_4mu[i]])) {
00160             radmean += iasi_rad->Rad[track][xtrack][list_4mu[i]];
00161             numean += iasi_rad->freq[list_4mu[i]];
00162             n++;
00163           }
00164         if (n > 0.9 * N4)
00165           pert_4mu->bt[track0 + track][xtrack]
00166             = brightness(radmean / n, numean / n);
00167         else
00168           pert_4mu->bt[track0 + track][xtrack] = GSL_NAN;
00169       }
00170
00171     /* Get 15 micron brightness temperature (low altitudes)... */
00172     for (track = 0; track < iasi_rad->ntrack; track++)
00173       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00174         n = 0;
00175         numean = radmean = 0;
00176         for (i = 0; i < N15_LOW; i++)
00177           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_15mu_low[i]])) {
00178             radmean += iasi_rad->Rad[track][xtrack][list_15mu_low[i]];
00179             numean += iasi_rad->freq[list_15mu_low[i]];
00180             n++;
00181           }
00182         if (n > 0.9 * N15_LOW)
00183           pert_15mu_low->bt[track0 + track][xtrack]
00184             = brightness(radmean / n, numean / n);
00185         else
00186           pert_15mu_low->bt[track0 + track][xtrack] = GSL_NAN;
00187       }
00188
00189     /* Get 15 micron brightness temperature (high altitudes)... */
00190     for (track = 0; track < iasi_rad->ntrack; track++)
00191       for (xtrack = 0; xtrack < L1_NXTRACK; xtrack++) {
00192         n = 0;
00193         numean = radmean = 0;
00194         for (i = 0; i < N15_HIGH; i++)
00195           if (gsl_finite(iasi_rad->Rad[track][xtrack][list_15mu_high[i]])) {
00196             radmean += iasi_rad->Rad[track][xtrack][list_15mu_high[i]];
00197             numean += iasi_rad->freq[list_15mu_high[i]];
00198             n++;
00199           }
00200         if (n > 0.9 * N15_HIGH)
00201           pert_15mu_high->bt[track0 + track][xtrack]
00202             = brightness(radmean / n, numean / n);
00203         else
00204           pert_15mu_high->bt[track0 + track][xtrack] = GSL_NAN;
00205       }
00206
00207     /* Increment track counter... */
```

```
00208     track0 += iasi_rad->ntrack;
00209   }
00210
00211   /* ------------------------------------------------------------
00212      Calculate perturbations and variances...
00213      ------------------------------------------------------------ */
00214
00215   /* Convert to wave analysis struct... */
00216   pert2wave(pert_4mu, &wave,
00217             0, pert_4mu->ntrack - 1, 0, pert_4mu->nxtrack - 1);
00218
00219   /* Estimate background... */
00220   background_poly(&wave, 5, 0);
00221
00222   /* Compute variance... */
00223   variance(&wave, var_dh);
00224
00225   /* Copy data... */
00226   for (ix = 0; ix < wave.nx; ix++)
00227     for (iy = 0; iy < wave.ny; iy++) {
00228       pert_4mu->pt[iy][ix] = wave.pt[ix][iy];
00229       pert_4mu->var[iy][ix] = wave.var[ix][iy];
00230     }
00231
00232   /* Convert to wave analysis struct... */
00233   pert2wave(pert_15mu_low, &wave,
00234             0, pert_15mu_low->ntrack - 1, 0, pert_15mu_low->nxtrack - 1);
00235
00236   /* Estimate background... */
00237   background_poly(&wave, 5, 0);
00238
00239   /* Compute variance... */
00240   variance(&wave, var_dh);
00241
00242   /* Copy data... */
00243   for (ix = 0; ix < wave.nx; ix++)
00244     for (iy = 0; iy < wave.ny; iy++) {
00245       pert_15mu_low->pt[iy][ix] = wave.pt[ix][iy];
00246       pert_15mu_low->var[iy][ix] = wave.var[ix][iy];
00247     }
00248
00249   /* Convert to wave analysis struct... */
00250   pert2wave(pert_15mu_high, &wave,
00251             0, pert_15mu_high->ntrack - 1, 0, pert_15mu_high->nxtrack - 1);
00252
00253   /* Estimate background... */
00254   background_poly(&wave, 5, 0);
00255
00256   /* Compute variance... */
00257   variance(&wave, var_dh);
00258
00259   /* Copy data... */
00260   for (ix = 0; ix < wave.nx; ix++)
00261     for (iy = 0; iy < wave.ny; iy++) {
00262       pert_15mu_high->pt[iy][ix] = wave.pt[ix][iy];
00263       pert_15mu_high->var[iy][ix] = wave.var[ix][iy];
00264     }
00265
00266   /* ------------------------------------------------------------
00267      Write to netCDF file...
00268      ------------------------------------------------------------ */
00269
00270   /* Create netCDF file... */
00271   printf("Write perturbation data file: %s\n", argv[1]);
00272   NC(nc_create(argv[1], NC_CLOBBER, &ncid));
00273
00274   /* Set dimensions... */
00275   NC(nc_def_dim(ncid, "NTRACK", NC_UNLIMITED, &dimid[0]));
00276   NC(nc_def_dim(ncid, "NXTRACK", L1_NXTRACK, &dimid[1]));
00277
00278   /* Add variables... */
00279   NC(nc_def_var(ncid, "time", NC_DOUBLE, 2, dimid, &time_varid));
00280   addatt(ncid, time_varid, "s", "time (seconds since 2000-01-01T00:00Z)");
00281   NC(nc_def_var(ncid, "lon", NC_DOUBLE, 2, dimid, &lon_varid));
00282   addatt(ncid, lon_varid, "deg", "footprint longitude");
00283   NC(nc_def_var(ncid, "lat", NC_DOUBLE, 2, dimid, &lat_varid));
00284   addatt(ncid, lat_varid, "deg", "footprint latitude");
00285
00286   NC(nc_def_var(ncid, "bt_8mu", NC_FLOAT, 2, dimid, &bt_8mu_varid));
00287   addatt(ncid, bt_8mu_varid, "K", "brightness temperature at 8.1 micron");
00288
00289   NC(nc_def_var(ncid, "bt_4mu", NC_FLOAT, 2, dimid, &bt_4mu_varid));
00290   addatt(ncid, bt_4mu_varid, "K", "brightness temperature" " at 4.3 micron");
00291   NC(nc_def_var(ncid, "bt_4mu_pt", NC_FLOAT, 2, dimid, &bt_4mu_pt_varid));
00292   addatt(ncid, bt_4mu_pt_varid, "K", "brightness temperature perturbation"
00293          " at 4.3 micron");
00294   NC(nc_def_var(ncid, "bt_4mu_var", NC_FLOAT, 2, dimid, &bt_4mu_var_varid));
```

```
00295   addatt(ncid, bt_4mu_var_varid, "K^2", "brightness temperature variance"
00296          " at 4.3 micron");
00297
00298   NC(nc_def_var(ncid, "bt_15mu_low", NC_FLOAT, 2, dimid, &bt_15mu_low_varid));
00299   addatt(ncid, bt_15mu_low_varid, "K", "brightness temperature"
00300          " at 15 micron (low altitudes)");
00301   NC(nc_def_var(ncid, "bt_15mu_low_pt", NC_FLOAT, 2, dimid,
00302                 &bt_15mu_low_pt_varid));
00303   addatt(ncid, bt_15mu_low_pt_varid, "K",
00304          "brightness temperature perturbation"
00305          " at 15 micron (low altitudes)");
00306   NC(nc_def_var
00307      (ncid, "bt_15mu_low_var", NC_FLOAT, 2, dimid, &bt_15mu_low_var_varid));
00308   addatt(ncid, bt_15mu_low_var_varid, "K^2",
00309          "brightness temperature variance" " at 15 micron (low altitudes)");
00310
00311   NC(nc_def_var(ncid, "bt_15mu_high", NC_FLOAT, 2, dimid,
00312                 &bt_15mu_high_varid));
00313   addatt(ncid, bt_15mu_high_varid, "K", "brightness temperature"
00314          " at 15 micron (high altitudes)");
00315   NC(nc_def_var(ncid, "bt_15mu_high_pt", NC_FLOAT, 2, dimid,
00316                 &bt_15mu_high_pt_varid));
00317   addatt(ncid, bt_15mu_high_pt_varid, "K",
00318          "brightness temperature perturbation"
00319          " at 15 micron (high altitudes)");
00320   NC(nc_def_var
00321      (ncid, "bt_15mu_high_var", NC_FLOAT, 2, dimid, &bt_15mu_high_var_varid));
00322   addatt(ncid, bt_15mu_high_var_varid, "K^2",
00323          "brightness temperature variance" " at 15 micron (high altitudes)");
00324
00325   /* Leave define mode... */
00326   NC(nc_enddef(ncid));
00327
00328   /* Loop over tracks... */
00329   for (track = 0; track < pert_4mu->ntrack; track++) {
00330
00331     /* Set array sizes... */
00332     start[0] = (size_t) track;
00333     start[1] = 0;
00334     count[0] = 1;
00335     count[1] = (size_t) pert_4mu->nxtrack;
00336
00337     /* Write data... */
00338     NC(nc_put_vara_double(ncid, time_varid, start, count,
00339                           pert_4mu->time[track]));
00340     NC(nc_put_vara_double(ncid, lon_varid, start, count,
00341                           pert_4mu->lon[track]));
00342     NC(nc_put_vara_double(ncid, lat_varid, start, count,
00343                           pert_4mu->lat[track]));
00344
00345     NC(nc_put_vara_double(ncid, bt_8mu_varid, start, count,
00346                           pert_4mu->dc[track]));
00347
00348     NC(nc_put_vara_double(ncid, bt_4mu_varid, start, count,
00349                           pert_4mu->bt[track]));
00350     NC(nc_put_vara_double(ncid, bt_4mu_pt_varid, start, count,
00351                           pert_4mu->pt[track]));
00352     NC(nc_put_vara_double(ncid, bt_4mu_var_varid, start, count,
00353                           pert_4mu->var[track]));
00354
00355     NC(nc_put_vara_double(ncid, bt_15mu_low_varid, start, count,
00356                           pert_15mu_low->bt[track]));
00357     NC(nc_put_vara_double(ncid, bt_15mu_low_pt_varid, start, count,
00358                           pert_15mu_low->pt[track]));
00359     NC(nc_put_vara_double(ncid, bt_15mu_low_var_varid, start, count,
00360                           pert_15mu_low->var[track]));
00361
00362     NC(nc_put_vara_double(ncid, bt_15mu_high_varid, start, count,
00363                           pert_15mu_high->bt[track]));
00364     NC(nc_put_vara_double(ncid, bt_15mu_high_pt_varid, start, count,
00365                           pert_15mu_high->pt[track]));
00366     NC(nc_put_vara_double(ncid, bt_15mu_high_var_varid, start, count,
00367                           pert_15mu_high->var[track]));
00368   }
00369
00370   /* Close file... */
00371   NC(nc_close(ncid));
00372
00373   /* Free... */
00374   free(iasi_rad);
00375   free(pert_4mu);
00376   free(pert_15mu_low);
00377   free(pert_15mu_high);
00378
00379   return EXIT_SUCCESS;
00380 }
00381
```

```
00382 /*****************************************************************************/
00383
00384 void addatt(
00385   int ncid,
00386   int varid,
00387   const char *unit,
00388   const char *long_name) {
00389
00390   /* Set long name... */
00391   NC(nc_put_att_text(ncid, varid, "long_name", strlen(long_name), long_name));
00392
00393   /* Set units... */
00394   NC(nc_put_att_text(ncid, varid, "units", strlen(unit), unit));
00395 }
```

5.17 retrieval.c File Reference

Data Structures

- struct ncd_t

  Buffer for netCDF data.
- struct ret_t

  Retrieval control parameters.

Functions

- void add_var (int ncid, const char *varname, const char *unit, const char *longname, int type, int dimid[ ], int *varid, int ndims)

  Create variable in netCDF file.
- void buffer_nc (atm_t *atm, double chisq, ncd_t *ncd, int track, int xtrack, int np0, int np1)

  Buffer netCDF data.
- double cost_function (gsl_vector *dx, gsl_vector *dy, gsl_matrix *s_a_inv, gsl_vector *sig_eps_inv)

  Compute cost function.
- void init_l2 (ncd_t *ncd, int track, int xtrack, ctl_t *ctl, atm_t *atm)

  Initialize with IASI Level-2 data.
- void matrix_invert (gsl_matrix *a)

  Invert symmetric matrix.
- void matrix_product (gsl_matrix *a, gsl_vector *b, int transpose, gsl_matrix *c)

  Compute matrix product A^TBA or ABA^T for diagonal matrix B.
- void optimal_estimation (ret_t *ret, ctl_t *ctl, obs_t *obs_meas, obs_t *obs_i, atm_t *atm_apr, atm_t *atm_i, double *chisq)

  Carry out optimal estimation retrieval.
- void read_nc (char *filename, ncd_t *ncd)

  Read netCDF file.
- void read_ret_ctl (int argc, char *argv[ ], ctl_t *ctl, ret_t *ret)

  Read retrieval control parameters.
- void set_cov_apr (ret_t *ret, ctl_t *ctl, atm_t *atm, int *iqa, int *ipa, gsl_matrix *s_a)

  Set a priori covariance.
- void set_cov_meas (ret_t *ret, ctl_t *ctl, obs_t *obs, gsl_vector *sig_noise, gsl_vector *sig_formod, gsl_vector *sig_eps_inv)

  Set measurement errors.
- double sza (double sec, double lon, double lat)

  Calculate solar zenith angle.
- void write_nc (char *filename, ncd_t *ncd)

  Write to netCDF file...
- int main (int argc, char *argv[ ])

5.17.1 Function Documentation

5.17.1.1 void add_var (int *ncid*, const char * *varname*, const char * *unit*, const char * *longname*, int *type*, int *dimid[ ]*, int * *varid*, int *ndims*)

Create variable in netCDF file.

Add variable to netCDF file.

Definition at line 486 of file retrieval.c.

```
00494             {
00495
00496   /* Check if variable exists... */
00497   if (nc_inq_varid(ncid, varname, varid) != NC_NOERR) {
00498
00499     /* Define variable... */
00500     NC(nc_def_var(ncid, varname, type, ndims, dimid, varid));
00501
00502     /* Set long name... */
00503     NC(nc_put_att_text
00504        (ncid, *varid, "long_name", strlen(longname), longname));
00505
00506     /* Set units... */
00507     NC(nc_put_att_text(ncid, *varid, "units", strlen(unit), unit));
00508   }
00509 }
```

5.17.1.2 void buffer_nc (atm_t * *atm*, double *chisq*, ncd_t * *ncd*, int *track*, int *xtrack*, int *np0*, int *np1*)

Buffer netCDF data.

Definition at line 513 of file retrieval.c.

```
00520           {
00521
00522   int ip;
00523
00524   /* Set number of data points... */
00525   ncd->np = np1 - np0 + 1;
00526
00527   /* Save retrieval data... */
00528   ncd->ret_chisq[track * L1_NXTRACK + xtrack] = (float) chisq;
00529   ncd->ret_p[track * L1_NXTRACK + xtrack] = (float) atm->p[np0];
00530   for (ip = np0; ip <= np1; ip++) {
00531     ncd->ret_z[ip - np0] = (float) atm->z[ip];
00532     ncd->ret_t[(track * L1_NXTRACK + xtrack) * ncd->np + ip - np0] =
00533       (gsl_finite(chisq) ? (float) atm->t[ip] : GSL_NAN);
00534   }
00535 }
```

5.17.1.3 double cost_function (gsl_vector * *dx*, gsl_vector * *dy*, gsl_matrix * *s_a_inv*, gsl_vector * *sig_eps_inv*)

Compute cost function.

Definition at line 539 of file retrieval.c.

```
00543                           {
00544
00545   gsl_vector *x_aux, *y_aux;
00546
00547   double chisq_a, chisq_m = 0;
00548
00549   size_t i, m, n;
00550
00551   /* Get sizes... */
00552   m = dy->size;
00553   n = dx->size;
00554
00555   /* Allocate... */
00556   x_aux = gsl_vector_alloc(n);
00557   y_aux = gsl_vector_alloc(m);
00558
00559   /* Determine normalized cost function...
00560      (chi^2 = 1/m * [dy^T * S_eps^{-1} * dy + dx^T * S_a^{-1} * dx]) */
00561   for (i = 0; i < m; i++)
00562     chisq_m +=
00563       gsl_pow_2(gsl_vector_get(dy, i) * gsl_vector_get(sig_eps_inv, i));
00564   gsl_blas_dgemv(CblasNoTrans, 1.0, s_a_inv, dx, 0.0, x_aux);
00565   gsl_blas_ddot(dx, x_aux, &chisq_a);
00566
00567   /* Free... */
00568   gsl_vector_free(x_aux);
00569   gsl_vector_free(y_aux);
00570
00571   /* Return cost function value... */
00572   return (chisq_m + chisq_a) / (double) m;
00573 }
```

5.17.1.4 void init_l2 (ncd_t * *ncd*, int *track*, int *xtrack*, ctl_t * *ctl*, atm_t * *atm*)

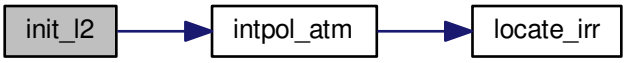
Initialize with IASI Level-2 data.

Definition at line 577 of file retrieval.c.

```
00582               {
00583
00584   static atm_t atm_iasi;
00585
00586   double k[NW], p, q[NG], t, w, zmax = 0, zmin = 1000;
00587
00588   int ip, lay;
00589
00590   /* Store IASI data in atmospheric data struct... */
00591   atm_iasi.np = 0;
00592   for (lay = 0; lay < L2_NLAY; lay++)
00593     if (gsl_finite(ncd->l2_z[track][xtrack][lay])
00594         && ncd->l2_z[track][xtrack][lay] <= 60.) {
00595       atm_iasi.z[atm_iasi.np] = ncd->l2_z[track][xtrack][lay];
00596       atm_iasi.p[atm_iasi.np] = ncd->l2_p[lay];
00597       atm_iasi.t[atm_iasi.np] = ncd->l2_t[track][xtrack][lay];
00598       if ((++atm_iasi.np) > NP)
00599         ERRMSG("Too many layers!");
00600     }
00601
00602   /* Check number of levels... */
00603   if (atm_iasi.np < 2)
00604     return;
00605
00606   /* Get height range of IASI data... */
00607   for (ip = 0; ip < atm_iasi.np; ip++) {
00608     zmax = GSL_MAX(zmax, atm_iasi.z[ip]);
00609     zmin = GSL_MIN(zmin, atm_iasi.z[ip]);
00610   }
00611
00612   /* Merge IASI data... */
00613   for (ip = 0; ip < atm->np; ip++) {
00614
00615     /* Interpolate IASI data... */
00616     intpol_atm(ctl, &atm_iasi, atm->z[ip], &p, &t, q, k);
00617
00618     /* Weighting factor... */
00619     w = 1;
00620     if (atm->z[ip] > zmax)
00621       w = GSL_MAX(1 - (atm->z[ip] - zmax) / 50, 0);
00622     if (atm->z[ip] < zmin)
00623       w = GSL_MAX(1 - (zmin - atm->z[ip]) / 50, 0);
```

```
00624
00625     /* Merge... */
00626     atm->t[ip] = w * t + (1 - w) * atm->t[ip];
00627     atm->p[ip] = w * p + (1 - w) * atm->p[ip];
00628   }
00629 }
```

Here is the call graph for this function:



5.17.1.5 void matrix_invert (gsl_matrix * *a*)

Invert symmetric matrix.

Definition at line 633 of file retrieval.c.

```
00634                   {
00635
00636   size_t diag = 1, i, j, n;
00637
00638   /* Get size... */
00639   n = a->size1;
00640
00641   /* Check if matrix is diagonal... */
00642   for (i = 0; i < n && diag; i++)
00643     for (j = i + 1; j < n; j++)
00644       if (gsl_matrix_get(a, i, j) != 0) {
00645         diag = 0;
00646         break;
00647       }
00648
00649   /* Quick inversion of diagonal matrix... */
00650   if (diag)
00651     for (i = 0; i < n; i++)
00652       gsl_matrix_set(a, i, i, 1 / gsl_matrix_get(a, i, i));
00653
00654   /* Matrix inversion by means of Cholesky decomposition... */
00655   else {
00656     gsl_linalg_cholesky_decomp(a);
00657     gsl_linalg_cholesky_invert(a);
00658   }
00659 }
```

5.17.1.6 void matrix_product (gsl_matrix * *a*, gsl_vector * *b*, int *transpose*, gsl_matrix * *c*)

Compute matrix product A^TBA or ABA^T for diagonal matrix B.

Definition at line 663 of file retrieval.c.

```
00667                   {
00668
00669   gsl_matrix *aux;
00670
00671   size_t i, j, m, n;
00672
00673   /* Set sizes... */
00674   m = a->size1;
```

```
00675   n = a->size2;
00676
00677   /* Allocate... */
00678   aux = gsl_matrix_alloc(m, n);
00679
00680   /* Compute A^T B A... */
00681   if (transpose == 1) {
00682
00683     /* Compute B^1/2 A... */
00684     for (i = 0; i < m; i++)
00685       for (j = 0; j < n; j++)
00686         gsl_matrix_set(aux, i, j,
00687                        gsl_vector_get(b, i) * gsl_matrix_get(a, i, j));
00688
00689     /* Compute A^T B A = (B^1/2 A)^T (B^1/2 A)... */
00690     gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, aux, aux, 0.0, c);
00691   }
00692
00693   /* Compute A B A^T... */
00694   else if (transpose == 2) {
00695
00696     /* Compute A B^1/2... */
00697     for (i = 0; i < m; i++)
00698       for (j = 0; j < n; j++)
00699         gsl_matrix_set(aux, i, j,
00700                        gsl_matrix_get(a, i, j) * gsl_vector_get(b, j));
00701
00702     /* Compute A B A^T = (A B^1/2) (A B^1/2)^T... */
00703     gsl_blas_dgemm(CblasNoTrans, CblasTrans, 1.0, aux, aux, 0.0, c);
00704   }
00705
00706   /* Free... */
00707   gsl_matrix_free(aux);
00708 }
```

5.17.1.7 void optimal_estimation (ret_t * *ret*, ctl_t * *ctl*, obs_t * *obs_meas*, obs_t * *obs_i*, atm_t * *atm_apr*, atm_t * *atm_i*, double * *chisq*)

Carry out optimal estimation retrieval.

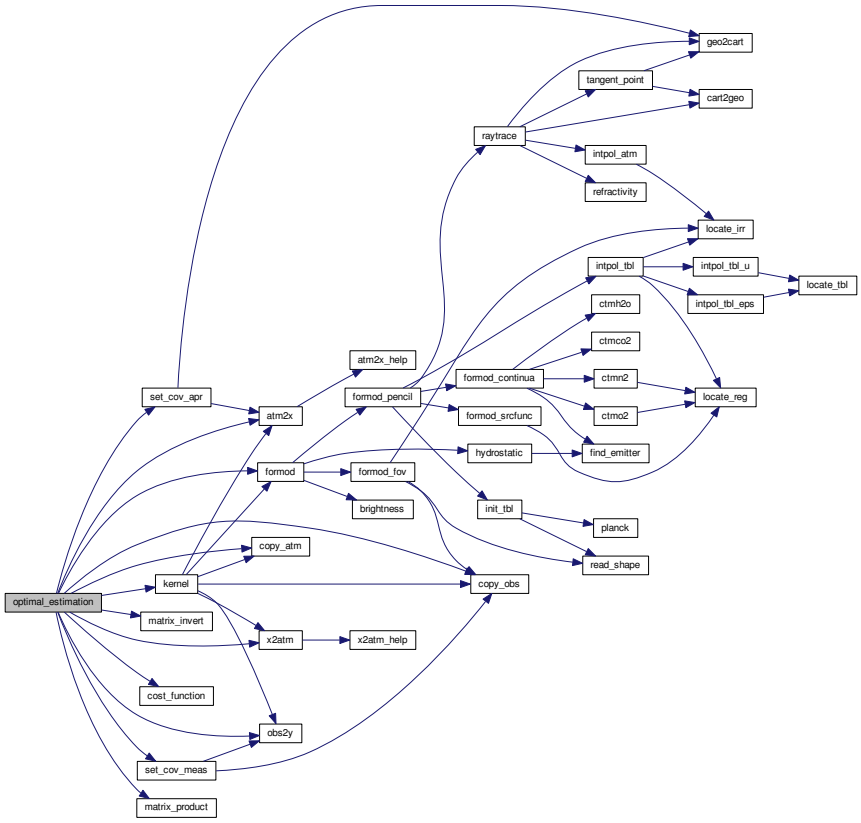
Definition at line 712 of file retrieval.c.

```
00719                 {
00720
00721   static int ipa[N], iqa[N];
00722
00723   gsl_matrix *a, *cov, *k_i, *s_a_inv;
00724
00725   gsl_vector *b, *dx, *dy, *sig_eps_inv, *sig_formod, *sig_noise,
00726     *x_a, *x_i, *x_step, *y_aux, *y_i, *y_m;
00727
00728   double chisq_old, disq = 0, lmpar = 0.001;
00729
00730   int ig, ip, it = 0, it2, iw;
00731
00732   size_t i, m, n;
00733
00734   /* ------------------------------------------------------------
00735      Initialize...
00736      ------------------------------------------------------------ */
00737
00738   /* Get sizes... */
00739   m = obs2y(ctl, obs_meas, NULL, NULL, NULL);
00740   n = atm2x(ctl, atm_apr, NULL, iqa, ipa);
00741   if (m <= 0 || n <= 0) {
00742     *chisq = GSL_NAN;
00743     return;
00744   }
00745
00746   /* Allocate... */
00747   a = gsl_matrix_alloc(n, n);
00748   cov = gsl_matrix_alloc(n, n);
00749   k_i = gsl_matrix_alloc(m, n);
00750   s_a_inv = gsl_matrix_alloc(n, n);
00751
00752   b = gsl_vector_alloc(n);
00753   dx = gsl_vector_alloc(n);
00754   dy = gsl_vector_alloc(m);
00755   sig_eps_inv = gsl_vector_alloc(m);
00756   sig_formod = gsl_vector_alloc(m);
```

```
00757   sig_noise = gsl_vector_alloc(m);
00758   x_a = gsl_vector_alloc(n);
00759   x_i = gsl_vector_alloc(n);
00760   x_step = gsl_vector_alloc(n);
00761   y_aux = gsl_vector_alloc(m);
00762   y_i = gsl_vector_alloc(m);
00763   y_m = gsl_vector_alloc(m);
00764
00765   /* Set initial state... */
00766   copy_atm(ctl, atm_i, atm_apr, 0);
00767   copy_obs(ctl, obs_i, obs_meas, 0);
00768   formod(ctl, atm_i, obs_i);
00769
00770   /* Set state vectors and observation vectors... */
00771   atm2x(ctl, atm_apr, x_a, NULL, NULL);
00772   atm2x(ctl, atm_i, x_i, NULL, NULL);
00773   obs2y(ctl, obs_meas, y_m, NULL, NULL);
00774   obs2y(ctl, obs_i, y_i, NULL, NULL);
00775
00776   /* Set inverse a priori covariance S_a^-1... */
00777   set_cov_apr(ret, ctl, atm_apr, iqa, ipa, s_a_inv);
00778   matrix_invert(s_a_inv);
00779
00780   /* Get measurement errors... */
00781   set_cov_meas(ret, ctl, obs_meas, sig_noise, sig_formod, sig_eps_inv);
00782
00783   /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00784   gsl_vector_memcpy(dx, x_i);
00785   gsl_vector_sub(dx, x_a);
00786   gsl_vector_memcpy(dy, y_m);
00787   gsl_vector_sub(dy, y_i);
00788
00789   /* Compute cost function... */
00790   *chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00791
00792   /* Compute initial kernel... */
00793   kernel(ctl, atm_i, obs_i, k_i);
00794
00795   /* ------------------------------------------------------------
00796      Levenberg-Marquardt minimization...
00797      ------------------------------------------------------------ */
00798
00799   /* Outer loop... */
00800   for (it = 1; it <= ret->conv_itmax; it++) {
00801
00802     /* Store current cost function value... */
00803     chisq_old = *chisq;
00804
00805     /* Compute kernel matrix K_i... */
00806     if (it > 1 && it % ret->kernel_recomp == 0)
00807       kernel(ctl, atm_i, obs_i, k_i);
00808
00809     /* Compute K_i^T * S_eps^{-1} * K_i ... */
00810     if (it == 1 || it % ret->kernel_recomp == 0)
00811       matrix_product(k_i, sig_eps_inv, 1, cov);
00812
00813     /* Determine b = K_i^T * S_eps^{-1} * dy - S_a^{-1} * dx ... */
00814     for (i = 0; i < m; i++)
00815       gsl_vector_set(y_aux, i, gsl_vector_get(dy, i)
00816                      * gsl_pow_2(gsl_vector_get(sig_eps_inv, i)));
00817     gsl_blas_dgemv(CblasTrans, 1.0, k_i, y_aux, 0.0, b);
00818     gsl_blas_dgemv(CblasNoTrans, -1.0, s_a_inv, dx, 1.0, b);
00819
00820     /* Inner loop... */
00821     for (it2 = 0; it2 < 20; it2++) {
00822
00823       /* Compute A = (1 + lmpar) * S_a^{-1} + K_i^T * S_eps^{-1} * K_i ... */
00824       gsl_matrix_memcpy(a, s_a_inv);
00825       gsl_matrix_scale(a, 1 + lmpar);
00826       gsl_matrix_add(a, cov);
00827
00828       /* Solve A * x_step = b by means of Cholesky decomposition... */
00829       gsl_linalg_cholesky_decomp(a);
00830       gsl_linalg_cholesky_solve(a, b, x_step);
00831
00832       /* Update atmospheric state... */
00833       gsl_vector_add(x_i, x_step);
00834       copy_atm(ctl, atm_i, atm_apr, 0);
00835       copy_obs(ctl, obs_i, obs_meas, 0);
00836       x2atm(ctl, x_i, atm_i);
00837
00838       /* Check atmospheric state... */
00839       for (ip = 0; ip < atm_i->np; ip++) {
00840         atm_i->p[ip] = GSL_MIN(GSL_MAX(atm_i->p[ip], 5e-7), 5e4);
00841         atm_i->t[ip] = GSL_MIN(GSL_MAX(atm_i->t[ip], 100), 400);
00842         for (ig = 0; ig < ctl->ng; ig++)
00843           atm_i->q[ig][ip] = GSL_MIN(GSL_MAX(atm_i->q[ig][ip], 0), 1);
```

```
00844         for (iw = 0; iw < ctl->nw; iw++)
00845           atm_i->k[iw][ip] = GSL_MAX(atm_i->k[iw][ip], 0);
00846       }
00847
00848       /* Forward calculation... */
00849       formod(ctl, atm_i, obs_i);
00850       obs2y(ctl, obs_i, y_i, NULL, NULL);
00851
00852       /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00853       gsl_vector_memcpy(dx, x_i);
00854       gsl_vector_sub(dx, x_a);
00855       gsl_vector_memcpy(dy, y_m);
00856       gsl_vector_sub(dy, y_i);
00857
00858       /* Compute cost function... */
00859       *chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00860
00861       /* Modify Levenberg-Marquardt parameter... */
00862       if (*chisq > chisq_old) {
00863         lmpar *= 10;
00864         gsl_vector_sub(x_i, x_step);
00865       } else {
00866         lmpar /= 10;
00867         break;
00868       }
00869     }
00870
00871     /* Get normalized step size in state space... */
00872     gsl_blas_ddot(x_step, b, &disq);
00873     disq /= (double) n;
00874
00875     /* Convergence test... */
00876     if ((it == 1 || it % ret->kernel_recomp == 0) && disq < ret->
      conv_dmin)
00877       break;
00878   }
00879
00880   /* ------------------------------------------------------------
00881      Finalize...
00882      ------------------------------------------------------------ */
00883
00884   gsl_matrix_free(a);
00885   gsl_matrix_free(cov);
00886   gsl_matrix_free(k_i);
00887   gsl_matrix_free(s_a_inv);
00888
00889   gsl_vector_free(b);
00890   gsl_vector_free(dx);
00891   gsl_vector_free(dy);
00892   gsl_vector_free(sig_eps_inv);
00893   gsl_vector_free(sig_formod);
00894   gsl_vector_free(sig_noise);
00895   gsl_vector_free(x_a);
00896   gsl_vector_free(x_i);
00897   gsl_vector_free(x_step);
00898   gsl_vector_free(y_aux);
00899   gsl_vector_free(y_i);
00900   gsl_vector_free(y_m);
00901 }
```

Here is the call graph for this function:



5.17.1.8 void read_nc (char * *filename*, ncd_t * *ncd*)

Read netCDF file.

Definition at line 905 of file retrieval.c.

```
00907               {
00908
00909   int dimid, varid;
00910
00911   size_t len;
00912
00913   /* Open netCDF file... */
00914   printf("Read netCDF file: %s\n", filename);
00915   NC(nc_open(filename, NC_WRITE, &ncd->ncid));
00916
00917   /* Read number of tracks... */
00918   NC(nc_inq_dimid(ncd->ncid, "L1_NTRACK", &dimid));
00919   NC(nc_inq_dimlen(ncd->ncid, dimid, &len));
00920   ncd->ntrack = (int) len;
00921
00922   /* Read Level-1 data... */
00923   NC(nc_inq_varid(ncd->ncid, "l1_time", &varid));
00924   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_time[0]));
00925   NC(nc_inq_varid(ncd->ncid, "l1_lon", &varid));
00926   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_lon[0]));
00927   NC(nc_inq_varid(ncd->ncid, "l1_lat", &varid));
00928   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_lat[0]));
00929   NC(nc_inq_varid(ncd->ncid, "l1_sat_z", &varid));
00930   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_z));
00931   NC(nc_inq_varid(ncd->ncid, "l1_sat_lon", &varid));
00932   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_lon));
00933   NC(nc_inq_varid(ncd->ncid, "l1_sat_lat", &varid));
```

```
00934   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_lat));
00935   NC(nc_inq_varid(ncd->ncid, "l1_nu", &varid));
00936   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_nu));
00937   NC(nc_inq_varid(ncd->ncid, "l1_rad", &varid));
00938   NC(nc_get_var_float(ncd->ncid, varid, ncd->l1_rad[0][0]));
00939
00940   /* Read Level-2 data... */
00941   NC(nc_inq_varid(ncd->ncid, "l2_z", &varid));
00942   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_z[0][0]));
00943   NC(nc_inq_varid(ncd->ncid, "l2_press", &varid));
00944   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_p));
00945   NC(nc_inq_varid(ncd->ncid, "l2_temp", &varid));
00946   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_t[0][0]));
00947 }
```

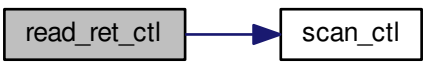
5.17.1.9 void read_ret_ctl (int *argc*, char * *argv[]*, ctl_t * *ctl*, ret_t * *ret*)

Read retrieval control parameters.

Definition at line 951 of file retrieval.c.

```
00955                {
00956
00957   int id, ig, iw;
00958
00959   /* Iteration control... */
00960   ret->kernel_recomp =
00961     (int) scan_ctl(argc, argv, "KERNEL_RECOMP", -1, "3", NULL);
00962   ret->conv_itmax = (int) scan_ctl(argc, argv, "CONV_ITMAX", -1, "30", NULL);
00963   ret->conv_dmin = scan_ctl(argc, argv, "CONV_DMIN", -1, "0.1", NULL);
00964
00965   for (id = 0; id < ctl->nd; id++)
00966     ret->err_formod[id] = scan_ctl(argc, argv, "ERR_FORMOD", id, "0", NULL);
00967
00968   for (id = 0; id < ctl->nd; id++)
00969     ret->err_noise[id] = scan_ctl(argc, argv, "ERR_NOISE", id, "0", NULL);
00970
00971   ret->err_press = scan_ctl(argc, argv, "ERR_PRESS", -1, "0", NULL);
00972   ret->err_press_cz = scan_ctl(argc, argv, "ERR_PRESS_CZ", -1, "-999", NULL);
00973   ret->err_press_ch = scan_ctl(argc, argv, "ERR_PRESS_CH", -1, "-999", NULL);
00974
00975   ret->err_temp = scan_ctl(argc, argv, "ERR_TEMP", -1, "0", NULL);
00976   ret->err_temp_cz = scan_ctl(argc, argv, "ERR_TEMP_CZ", -1, "-999", NULL);
00977   ret->err_temp_ch = scan_ctl(argc, argv, "ERR_TEMP_CH", -1, "-999", NULL);
00978
00979   for (ig = 0; ig < ctl->ng; ig++) {
00980     ret->err_q[ig] = scan_ctl(argc, argv, "ERR_Q", ig, "0", NULL);
00981     ret->err_q_cz[ig] = scan_ctl(argc, argv, "ERR_Q_CZ", ig, "-999", NULL);
00982     ret->err_q_ch[ig] = scan_ctl(argc, argv, "ERR_Q_CH", ig, "-999", NULL);
00983   }
00984
00985   for (iw = 0; iw < ctl->nw; iw++) {
00986     ret->err_k[iw] = scan_ctl(argc, argv, "ERR_K", iw, "0", NULL);
00987     ret->err_k_cz[iw] = scan_ctl(argc, argv, "ERR_K_CZ", iw, "-999", NULL);
00988     ret->err_k_ch[iw] = scan_ctl(argc, argv, "ERR_K_CH", iw, "-999", NULL);
00989   }
00990 }
```

Here is the call graph for this function:



5.17.1.10 void set_cov_apr (ret_t * *ret*, ctl_t * *ctl*, atm_t * *atm*, int * *iqa*, int * *ipa*, gsl_matrix * *s_a*)

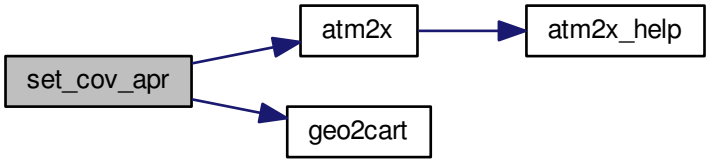
Set a priori covariance.

Definition at line 994 of file retrieval.c.

```
01000                   {
01001
01002   gsl_vector *x_a;
01003
01004   double ch, cz, rho, x0[3], x1[3];
01005
01006   int ig, iw;
01007
01008   size_t i, j, n;
01009
01010   /* Get sizes... */
01011   n = s_a->size1;
01012
01013   /* Allocate... */
01014   x_a = gsl_vector_alloc(n);
01015
01016   /* Get sigma vector... */
01017   atm2x(ctl, atm, x_a, NULL, NULL);
01018   for (i = 0; i < n; i++) {
01019     if (iqa[i] == IDXP)
01020       gsl_vector_set(x_a, i, ret->err_press / 100 * gsl_vector_get(x_a, i));
01021     if (iqa[i] == IDXT)
01022       gsl_vector_set(x_a, i, ret->err_temp);
01023     for (ig = 0; ig < ctl->ng; ig++)
01024       if (iqa[i] == IDXQ(ig))
01025         gsl_vector_set(x_a, i, ret->err_q[ig] / 100 * gsl_vector_get(x_a, i));
01026     for (iw = 0; iw < ctl->nw; iw++)
01027       if (iqa[i] == IDXK(iw))
01028         gsl_vector_set(x_a, i, ret->err_k[iw]);
01029   }
01030
01031   /* Check standard deviations... */
01032   for (i = 0; i < n; i++)
01033     if (gsl_pow_2(gsl_vector_get(x_a, i)) <= 0)
01034       ERRMSG("Check a priori data (zero standard deviation)!");
01035
01036   /* Initialize diagonal covariance... */
01037   gsl_matrix_set_zero(s_a);
01038   for (i = 0; i < n; i++)
01039     gsl_matrix_set(s_a, i, i, gsl_pow_2(gsl_vector_get(x_a, i)));
01040
01041   /* Loop over matrix elements... */
01042   for (i = 0; i < n; i++)
01043     for (j = 0; j < n; j++)
01044       if (i != j && iqa[i] == iqa[j]) {
01045
01046         /* Initialize... */
01047         cz = ch = 0;
01048
01049         /* Set correlation lengths for pressure... */
01050         if (iqa[i] == IDXP) {
01051           cz = ret->err_press_cz;
01052           ch = ret->err_press_ch;
01053         }
01054
01055         /* Set correlation lengths for temperature... */
01056         if (iqa[i] == IDXT) {
01057           cz = ret->err_temp_cz;
01058           ch = ret->err_temp_ch;
01059         }
01060
01061         /* Set correlation lengths for volume mixing ratios... */
01062         for (ig = 0; ig < ctl->ng; ig++)
01063           if (iqa[i] == IDXQ(ig)) {
01064             cz = ret->err_q_cz[ig];
01065             ch = ret->err_q_ch[ig];
01066           }
01067
01068         /* Set correlation lengths for extinction... */
01069         for (iw = 0; iw < ctl->nw; iw++)
01070           if (iqa[i] == IDXK(iw)) {
01071             cz = ret->err_k_cz[iw];
01072             ch = ret->err_k_ch[iw];
01073           }
01074
01075         /* Compute correlations... */
01076         if (cz > 0 && ch > 0) {
```

```
01077
01078           /* Get Cartesian coordinates... */
01079           geo2cart(0, atm->lon[ipa[i]], atm->lat[ipa[i]], x0);
01080           geo2cart(0, atm->lon[ipa[j]], atm->lat[ipa[j]], x1);
01081
01082           /* Compute correlations... */
01083           rho =
01084             exp(-DIST(x0, x1) / ch -
01085                 fabs(atm->z[ipa[i]] - atm->z[ipa[j]]) / cz);
01086
01087           /* Set covariance... */
01088           gsl_matrix_set(s_a, i, j, gsl_vector_get(x_a, i)
01089                          * gsl_vector_get(x_a, j) * rho);
01090         }
01091       }
01092
01093   /* Free... */
01094   gsl_vector_free(x_a);
01095 }
```

Here is the call graph for this function:



5.17.1.11 void set_cov_meas (ret_t * *ret*, ctl_t * *ctl*, obs_t * *obs*, gsl_vector * *sig_noise*, gsl_vector * *sig_formod*, gsl_vector * *sig_eps_inv*)

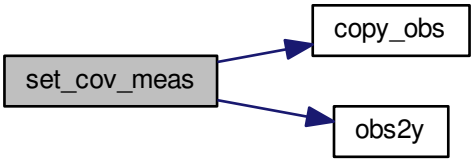
Set measurement errors.

Definition at line 1099 of file retrieval.c.

```
01105                           {
01106
01107   static obs_t obs_err;
01108
01109   int id, ir;
01110
01111   size_t i, m;
01112
01113   /* Get size... */
01114   m = sig_eps_inv->size;
01115
01116   /* Noise error (always considered in retrieval fit)... */
01117   copy_obs(ctl, &obs_err, obs, 1);
01118   for (ir = 0; ir < obs_err.nr; ir++)
01119     for (id = 0; id < ctl->nd; id++)
01120       obs_err.rad[id][ir]
01121         = (gsl_finite(obs->rad[id][ir]) ? ret->err_noise[id] : GSL_NAN);
01122   obs2y(ctl, &obs_err, sig_noise, NULL, NULL);
01123
01124   /* Forward model error (always considered in retrieval fit)... */
01125   copy_obs(ctl, &obs_err, obs, 1);
01126   for (ir = 0; ir < obs_err.nr; ir++)
01127     for (id = 0; id < ctl->nd; id++)
01128       obs_err.rad[id][ir]
01129         = fabs(ret->err_formod[id] / 100 * obs->rad[id][ir]);
01130   obs2y(ctl, &obs_err, sig_formod, NULL, NULL);
01131
01132   /* Total error... */
```

```
01133   for (i = 0; i < m; i++)
01134     gsl_vector_set(sig_eps_inv, i,
01135                    1 / sqrt(gsl_pow_2(gsl_vector_get(sig_noise, i))
01136                             + gsl_pow_2(gsl_vector_get(sig_formod, i))));
01137
01138   /* Check standard deviations... */
01139   for (i = 0; i < m; i++)
01140     if (gsl_vector_get(sig_eps_inv, i) <= 0)
01141       ERRMSG("Check measurement errors (zero standard deviation)!");
01142 }
```

Here is the call graph for this function:



5.17.1.12 double sza (double *sec*, double *lon*, double *lat*)

Calculate solar zenith angle.

Definition at line 1146 of file retrieval.c.

```
01149              {
01150
01151   double D, dec, e, g, GMST, h, L, LST, q, ra;
01152
01153   /* Number of days and fraction with respect to 2000-01-01T12:00Z... */
01154   D = sec / 86400 - 0.5;
01155
01156   /* Geocentric apparent ecliptic longitude [rad]... */
01157   g = (357.529 + 0.98560028 * D) * M_PI / 180;
01158   q = 280.459 + 0.98564736 * D;
01159   L = (q + 1.915 * sin(g) + 0.020 * sin(2 * g)) * M_PI / 180;
01160
01161   /* Mean obliquity of the ecliptic [rad]... */
01162   e = (23.439 - 0.00000036 * D) * M_PI / 180;
01163
01164   /* Declination [rad]... */
01165   dec = asin(sin(e) * sin(L));
01166
01167   /* Right ascension [rad]... */
01168   ra = atan2(cos(e) * sin(L), cos(L));
01169
01170   /* Greenwich Mean Sidereal Time [h]... */
01171   GMST = 18.697374558 + 24.06570982441908 * D;
01172
01173   /* Local Sidereal Time [h]... */
01174   LST = GMST + lon / 15;
01175
01176   /* Hour angle [rad]... */
01177   h = LST / 12 * M_PI - ra;
01178
01179   /* Convert latitude... */
01180   lat *= M_PI / 180;
01181
01182   /* Return solar zenith angle [deg]... */
01183   return acos(sin(lat) * sin(dec) +
01184               cos(lat) * cos(dec) * cos(h)) * 180 / M_PI;
01185 }
```

5.17.1.13 void write_nc (char * *filename*, ncd_t * *ncd*)

Write to netCDF file...

Definition at line 1189 of file retrieval.c.

```
01191                {
01192
01193   int dimid[10], c_id, p_id, t_id, z_id;
01194
01195   /* Create netCDF file... */
01196   printf("Write netCDF file: %s\n", filename);
01197
01198   /* Read existing dimensions... */
01199   NC(nc_inq_dimid(ncd->ncid, "L1_NTRACK", &dimid[0]));
01200   NC(nc_inq_dimid(ncd->ncid, "L1_NXTRACK", &dimid[1]));
01201
01202   /* Set define mode... */
01203   NC(nc_redef(ncd->ncid));
01204
01205   /* Set new dimensions... */
01206   if (nc_inq_dimid(ncd->ncid, "RET_NP", &dimid[2]) != NC_NOERR)
01207     NC(nc_def_dim(ncd->ncid, "RET_NP", (size_t) ncd->np, &dimid[2]));
01208
01209   /* Set new variables... */
01210   add_var(ncd->ncid, "ret_z", "km", "altitude", NC_FLOAT, &dimid[2], &z_id,
01211           1);
01212   add_var(ncd->ncid, "ret_press", "hPa", "pressure", NC_FLOAT, dimid, &p_id,
01213           2);
01214   add_var(ncd->ncid, "ret_temp", "K", "temperature", NC_FLOAT, dimid, &t_id,
01215           3);
01216   add_var(ncd->ncid, "ret_chisq", "1", "chi^2 value of fit", NC_FLOAT, dimid,
01217           &c_id, 2);
01218
01219   /* Leave define mode... */
01220   NC(nc_enddef(ncd->ncid));
01221
01222   /* Write data... */
01223   NC(nc_put_var_float(ncd->ncid, z_id, ncd->ret_z));
01224   NC(nc_put_var_float(ncd->ncid, p_id, ncd->ret_p));
01225   NC(nc_put_var_float(ncd->ncid, t_id, ncd->ret_t));
01226   NC(nc_put_var_float(ncd->ncid, c_id, ncd->ret_chisq));
01227
01228   /* Close netCDF file... */
01229   NC(nc_close(ncd->ncid));
01230 }
```

Here is the call graph for this function:

5.17.1.14 int main (int *argc*, char * *argv[ ]*)

Definition at line 263 of file retrieval.c.

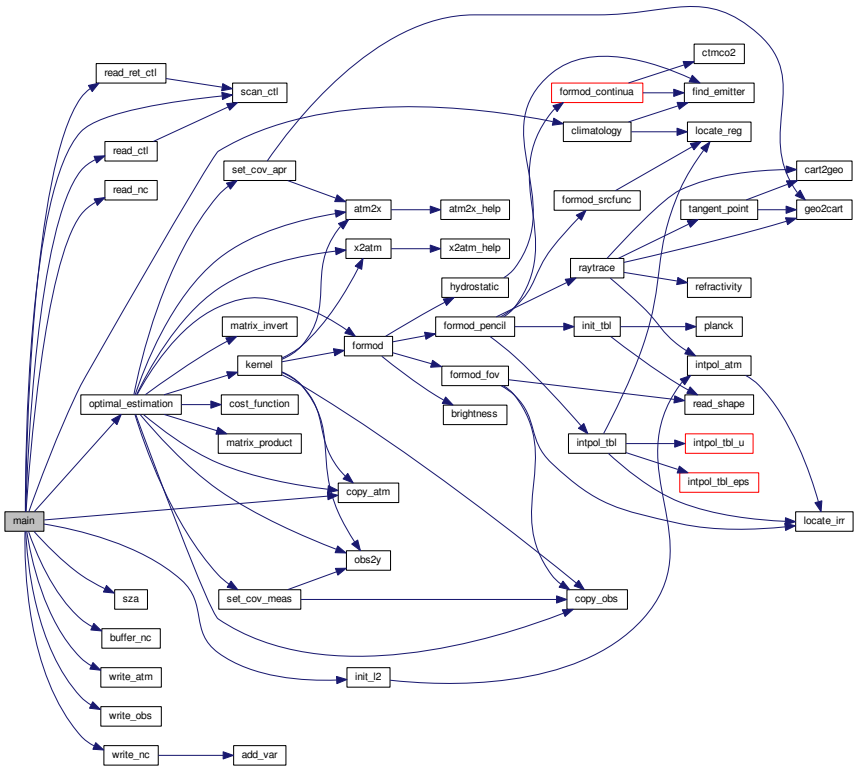
```
00265                 {
00266
00267   static ctl_t ctl;
00268   static atm_t atm_apr, atm_clim, atm_i;
00269   static obs_t obs_i, obs_meas;
00270   static ncd_t ncd;
```

```
00271   static ret_t ret;
00272
00273   FILE *in;
00274
00275   char filename[LEN], filename2[LEN];
00276
00277   double chisq, sza_thresh, z[NP], t0;
00278
00279   int channel[ND], i, id, ip, iz, nz, ntask = -1, rank, size,
00280     np0, np1, track, track0, track1, xtrack, xtrack0, xtrack1,
00281     task0, task1, debug;
00282
00283   /* ------------------------------------------------------------
00284      Init...
00285      ------------------------------------------------------------ */
00286
00287   /* MPI... */
00288   MPI_Init(&argc, &argv);
00289   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
00290   MPI_Comm_size(MPI_COMM_WORLD, &size);
00291
00292   /* Measure CPU time... */
00293   TIMER("total", 1);
00294
00295   /* Check arguments... */
00296   if (argc < 3)
00297     ERRMSG("Give parameters: <ctl> <filelist>");
00298
00299   /* Read control parameters... */
00300   read_ctl(argc, argv, &ctl);
00301   read_ret_ctl(argc, argv, &ctl, &ret);
00302   debug = (int) scan_ctl(argc, argv, "DEBUG", -1, "1", NULL);
00303
00304   /* Read retrieval grid... */
00305   nz = (int) scan_ctl(argc, argv, "NZ", -1, "", NULL);
00306   if (nz > NP)
00307     ERRMSG("Too many altitudes!");
00308   for (iz = 0; iz < nz; iz++)
00309     z[iz] = scan_ctl(argc, argv, "Z", iz, "", NULL);
00310
00311   /* Read task range... */
00312   task0 = (int) scan_ctl(argc, argv, "TASK_MIN", -1, "0", NULL);
00313   task1 = (int) scan_ctl(argc, argv, "TASK_MAX", -1, "99999", NULL);
00314
00315   /* Read track range... */
00316   track0 = (int) scan_ctl(argc, argv, "TRACK_MIN", -1, "0", NULL);
00317   track1 = (int) scan_ctl(argc, argv, "TRACK_MAX", -1, "99999", NULL);
00318
00319   /* Read xtrack range... */
00320   xtrack0 = (int) scan_ctl(argc, argv, "XTRACK_MIN", -1, "0", NULL);
00321   xtrack1 = (int) scan_ctl(argc, argv, "XTRACK_MAX", -1, "59", NULL);
00322
00323   /* Read height range... */
00324   np0 = (int) scan_ctl(argc, argv, "NP_MIN", -1, "0", NULL);
00325   np1 = (int) scan_ctl(argc, argv, "NP_MAX", -1, "100", NULL);
00326   np1 = GSL_MIN(np1, nz - 1);
00327
00328   /* SZA threshold... */
00329   sza_thresh = scan_ctl(argc, argv, "SZA", -1, "96", NULL);
00330
00331   /* ------------------------------------------------------------
00332      Distribute granules...
00333      ------------------------------------------------------------ */
00334
00335   /* Open filelist... */
00336   printf("Read filelist: %s\n", argv[2]);
00337   if (!(in = fopen(argv[2], "r")))
00338     ERRMSG("Cannot open filelist!");
00339
00340   /* Loop over netCDF files... */
00341   while (fscanf(in, "%s", filename) != EOF) {
00342
00343     /* Distribute files with MPI... */
00344     if ((++ntask) % size != rank)
00345       continue;
00346
00347     /* Check task range... */
00348     if (ntask < task0 || ntask > task1)
00349       continue;
00350
00351     /* Write info... */
00352     printf("Retrieve file %s on rank %d of %d (with %d threads)...\n",
00353            filename, rank + 1, size, omp_get_max_threads());
00354
00355     /* ------------------------------------------------------------
00356        Initialize retrieval...
00357        ------------------------------------------------------------ */
```

```
00358
00359     /* Read netCDF file... */
00360     read_nc(filename, &ncd);
00361
00362     /* Adjust number of tracks... */
00363     if (track1 >= ncd.ntrack)
00364       track1 = ncd.ntrack - 1;
00365
00366     /* Identify radiance channels... */
00367     for (id = 0; id < ctl.nd; id++) {
00368       channel[id] = -999;
00369       for (i = 0; i < L1_NCHAN; i++)
00370         if (fabs(ctl.nu[id] - ncd.l1_nu[i]) < 0.1)
00371           channel[id] = i;
00372       if (channel[id] < 0)
00373         ERRMSG("Cannot identify radiance channel!");
00374     }
00375
00376     /* Set climatological data for center of granule... */
00377     atm_clim.np = nz;
00378     for (iz = 0; iz < nz; iz++)
00379       atm_clim.z[iz] = z[iz];
00380     climatology(&ctl, &atm_clim);
00381
00382     /* ------------------------------------------------------------
00383        Retrieval...
00384        ------------------------------------------------------------ */
00385
00386     /* Loop over swaths... */
00387     for (track = track0; track <= track1; track++) {
00388
00389       /* Loop over scan... */
00390       for (xtrack = xtrack0; xtrack <= xtrack1; xtrack++) {
00391
00392         /* Init timer... */
00393         t0 = omp_get_wtime();
00394
00395         /* Store observation data... */
00396         obs_meas.nr = 1;
00397         obs_meas.time[0] = ncd.l1_time[track][xtrack];
00398         obs_meas.obsz[0] = ncd.l1_sat_z[track];
00399         obs_meas.obslon[0] = ncd.l1_sat_lon[track];
00400         obs_meas.obslat[0] = ncd.l1_sat_lat[track];
00401         obs_meas.vplon[0] = ncd.l1_lon[track][xtrack];
00402         obs_meas.vplat[0] = ncd.l1_lat[track][xtrack];
00403         for (id = 0; id < ctl.nd; id++)
00404           obs_meas.rad[id][0] = ncd.l1_rad[track][xtrack][channel[id]];
00405
00406         /* Flag out 4 micron channels for daytime measurements... */
00407         if (sza(obs_meas.time[0], obs_meas.obslon[0], obs_meas.
    obslat[0])
00408             < sza_thresh)
00409           for (id = 0; id < ctl.nd; id++)
00410             if (ctl.nu[id] >= 2000)
00411               obs_meas.rad[id][0] = GSL_NAN;
00412
00413         /* Prepare atmospheric data... */
00414         copy_atm(&ctl, &atm_apr, &atm_clim, 0);
00415         for (ip = 0; ip < atm_apr.np; ip++) {
00416           atm_apr.time[ip] = obs_meas.time[0];
00417           atm_apr.lon[ip] = obs_meas.vplon[0];
00418           atm_apr.lat[ip] = obs_meas.vplat[0];
00419         }
00420
00421         /* Merge Level-2 data... */
00422         init_l2(&ncd, track, xtrack, &ctl, &atm_apr);
00423
00424         /* Retrieval... */
00425         optimal_estimation(&ret, &ctl, &obs_meas, &obs_i,
00426                            &atm_apr, &atm_i, &chisq);
00427
00428         /* Buffer results... */
00429         buffer_nc(&atm_i, chisq, &ncd, track, xtrack, np0, np1);
00430
00431         /* Write debug information... */
00432         if (debug >= 1)
00433           printf
00434             ("  task= %4d | track= %5d | xtrack= %3d | chi^2= %8.3f | time= %8.3f s\n",
00435              ntask, track, xtrack, chisq, omp_get_wtime() - t0);
00436         if (debug >= 2) {
00437           sprintf(filename2, "atm_apr_%d_%d_%d.tab", ntask, track, xtrack);
00438           write_atm(NULL, filename2, &ctl, &atm_apr);
00439           sprintf(filename2, "atm_i_%d_%d_%d.tab", ntask, track, xtrack);
00440           write_atm(NULL, filename2, &ctl, &atm_i);
00441           sprintf(filename2, "obs_meas_%d_%d_%d.tab", ntask, track, xtrack);
00442           write_obs(NULL, filename2, &ctl, &obs_meas);
00443           sprintf(filename2, "obs_i_%d_%d_%d.tab", ntask, track, xtrack);
```

```
00444           write_obs(NULL, filename2, &ctl, &obs_i);
00445         }
00446       }
00447     }
00448
00449     /* ------------------------------------------------------------
00450        Finalize...
00451        ------------------------------------------------------------ */
00452
00453     /* Write netCDF file... */
00454     write_nc(filename, &ncd);
00455
00456     /* Write info... */
00457     printf("Retrieval finished on rank %d of %d!\n", rank, size);
00458   }
00459
00460   /* Close file list... */
00461   fclose(in);
00462
00463   /* Measure CPU time... */
00464   TIMER("total", 3);
00465
00466   /* Report memory usage... */
00467   printf("MEMORY_ATM = %g MByte\n", 4. * sizeof(atm_t) / 1024. / 1024.);
00468   printf("MEMORY_CTL = %g MByte\n", 1. * sizeof(ctl_t) / 1024. / 1024.);
00469   printf("MEMORY_NCD = %g MByte\n", 1. * sizeof(ncd_t) / 1024. / 1024.);
00470   printf("MEMORY_OBS = %g MByte\n", 3. * sizeof(atm_t) / 1024. / 1024.);
00471   printf("MEMORY_RET = %g MByte\n", 1. * sizeof(ret_t) / 1024. / 1024.);
00472   printf("MEMORY_TBL = %g MByte\n", 1. * sizeof(tbl_t) / 1024. / 1024.);
00473
00474   /* Report problem size... */
00475   printf("SIZE_TASKS = %d\n", size);
00476   printf("SIZE_THREADS = %d\n", omp_get_max_threads());
00477
00478   /* MPI... */
00479   MPI_Finalize();
00480
00481   return EXIT_SUCCESS;
00482 }
```

Here is the call graph for this function:



5.18 retrieval.c

```
00001 #include <mpi.h>
00002 #include <omp.h>
00003 #include <netcdf.h>
00004 #include "jurassic.h"
00005
00006 /* ------------------------------------------------------------
00007    Macros...
00008    ------------------------------------------------------------ */
00009
00011 #define NC(cmd) {                                   \
00012     if((cmd)!=NC_NOERR)                             \
00013       ERRMSG(nc_strerror(cmd));                     \
00014   }
00015
00016 /* ------------------------------------------------------------
00017    Dimensions...
00018    ------------------------------------------------------------ */
00019
00021 #define L1_NCHAN 33
00022
00024 #define L1_NTRACK 1800
00025
00027 #define L1_NXTRACK 60
00028
00030 #define L2_NLAY 27
00031
00033 #define L2_NTRACK 1800
00034
00036 #define L2_NXTRACK 60
00037
00038 /* ------------------------------------------------------------
00039    Structs...
00040    ------------------------------------------------------------ */
00041
00043 typedef struct {
00044
00046   int ncid;
00047
00049   int np;
00050
00052   int ntrack;
00053
00055   double l1_time[L1_NTRACK][L1_NXTRACK];
00056
00058   double l1_lon[L1_NTRACK][L1_NXTRACK];
00059
00061   double l1_lat[L1_NTRACK][L1_NXTRACK];
00062
00064   double l1_sat_z[L1_NTRACK];
00065
00067   double l1_sat_lon[L1_NTRACK];
00068
00070   double l1_sat_lat[L1_NTRACK];
00071
00073   double l1_nu[L1_NCHAN];
00074
00076   float l1_rad[L1_NTRACK][L1_NXTRACK][L1_NCHAN];
00077
00079   double l2_z[L2_NTRACK][L2_NXTRACK][L2_NLAY];
00080
00082   double l2_p[L2_NLAY];
00083
00085   double l2_t[L2_NTRACK][L2_NXTRACK][L2_NLAY];
00086
00088   float ret_z[NP];
00089
00091   float ret_p[L1_NTRACK * L1_NXTRACK];
00092
00094   float ret_t[L1_NTRACK * L1_NXTRACK * NP];
00095
00097   float ret_chisq[L1_NTRACK * L1_NXTRACK];
00098
00099 } ncd_t;
00100
00102 typedef struct {
00103
00105   int kernel_recomp;
00106
00108   int conv_itmax;
00109
00111   double conv_dmin;
00112
00114   double err_formod[ND];
00115
```

```
00117   double err_noise[ND];
00118
00120   double err_press;
00121
00123   double err_press_cz;
00124
00126   double err_press_ch;
00127
00129   double err_temp;
00130
00132   double err_temp_cz;
00133
00135   double err_temp_ch;
00136
00138   double err_q[NG];
00139
00141   double err_q_cz[NG];
00142
00144   double err_q_ch[NG];
00145
00147   double err_k[NW];
00148
00150   double err_k_cz[NW];
00151
00153   double err_k_ch[NW];
00154
00155 } ret_t;
00156
00157 /* ------------------------------------------------------------
00158    Functions...
00159    ------------------------------------------------------------ */
00160
00162 void add_var(
00163   int ncid,
00164   const char *varname,
00165   const char *unit,
00166   const char *longname,
00167   int type,
00168   int dimid[],
00169   int *varid,
00170   int ndims);
00171
00173 void buffer_nc(
00174   atm_t * atm,
00175   double chisq,
00176   ncd_t * ncd,
00177   int track,
00178   int xtrack,
00179   int np0,
00180   int np1);
00181
00183 double cost_function(
00184   gsl_vector * dx,
00185   gsl_vector * dy,
00186   gsl_matrix * s_a_inv,
00187   gsl_vector * sig_eps_inv);
00188
00190 void init_l2(
00191   ncd_t * ncd,
00192   int track,
00193   int xtrack,
00194   ctl_t * ctl,
00195   atm_t * atm);
00196
00198 void matrix_invert(
00199   gsl_matrix * a);
00200
00202 void matrix_product(
00203   gsl_matrix * a,
00204   gsl_vector * b,
00205   int transpose,
00206   gsl_matrix * c);
00207
00209 void optimal_estimation(
00210   ret_t * ret,
00211   ctl_t * ctl,
00212   obs_t * obs_meas,
00213   obs_t * obs_i,
00214   atm_t * atm_apr,
00215   atm_t * atm_i,
00216   double *chisq);
00217
00219 void read_nc(
00220   char *filename,
00221   ncd_t * ncd);
00222
00224 void read_ret_ctl(
```

```
00225   int argc,
00226   char *argv[],
00227   ctl_t * ctl,
00228   ret_t * ret);
00229
00231 void set_cov_apr(
00232   ret_t * ret,
00233   ctl_t * ctl,
00234   atm_t * atm,
00235   int *iqa,
00236   int *ipa,
00237   gsl_matrix * s_a);
00238
00240 void set_cov_meas(
00241   ret_t * ret,
00242   ctl_t * ctl,
00243   obs_t * obs,
00244   gsl_vector * sig_noise,
00245   gsl_vector * sig_formod,
00246   gsl_vector * sig_eps_inv);
00247
00249 double sza(
00250   double sec,
00251   double lon,
00252   double lat);
00253
00255 void write_nc(
00256   char *filename,
00257   ncd_t * ncd);
00258
00259 /* ------------------------------------------------------------
00260    Main...
00261    ------------------------------------------------------------ */
00262
00263 int main(
00264   int argc,
00265   char *argv[]) {
00266
00267   static ctl_t ctl;
00268   static atm_t atm_apr, atm_clim, atm_i;
00269   static obs_t obs_i, obs_meas;
00270   static ncd_t ncd;
00271   static ret_t ret;
00272
00273   FILE *in;
00274
00275   char filename[LEN], filename2[LEN];
00276
00277   double chisq, sza_thresh, z[NP], t0;
00278
00279   int channel[ND], i, id, ip, iz, nz, ntask = -1, rank, size,
00280     np0, np1, track, track0, track1, xtrack, xtrack0, xtrack1,
00281     task0, task1, debug;
00282
00283   /* ------------------------------------------------------------
00284      Init...
00285      ------------------------------------------------------------ */
00286
00287   /* MPI... */
00288   MPI_Init(&argc, &argv);
00289   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
00290   MPI_Comm_size(MPI_COMM_WORLD, &size);
00291
00292   /* Measure CPU time... */
00293   TIMER("total", 1);
00294
00295   /* Check arguments... */
00296   if (argc < 3)
00297     ERRMSG("Give parameters: <ctl> <filelist>");
00298
00299   /* Read control parameters... */
00300   read_ctl(argc, argv, &ctl);
00301   read_ret_ctl(argc, argv, &ctl, &ret);
00302   debug = (int) scan_ctl(argc, argv, "DEBUG", -1, "1", NULL);
00303
00304   /* Read retrieval grid... */
00305   nz = (int) scan_ctl(argc, argv, "NZ", -1, "", NULL);
00306   if (nz > NP)
00307     ERRMSG("Too many altitudes!");
00308   for (iz = 0; iz < nz; iz++)
00309     z[iz] = scan_ctl(argc, argv, "Z", iz, "", NULL);
00310
00311   /* Read task range... */
00312   task0 = (int) scan_ctl(argc, argv, "TASK_MIN", -1, "0", NULL);
00313   task1 = (int) scan_ctl(argc, argv, "TASK_MAX", -1, "99999", NULL);
00314
00315   /* Read track range... */
```

```
00316   track0 = (int) scan_ctl(argc, argv, "TRACK_MIN", -1, "0", NULL);
00317   track1 = (int) scan_ctl(argc, argv, "TRACK_MAX", -1, "99999", NULL);
00318
00319   /* Read xtrack range... */
00320   xtrack0 = (int) scan_ctl(argc, argv, "XTRACK_MIN", -1, "0", NULL);
00321   xtrack1 = (int) scan_ctl(argc, argv, "XTRACK_MAX", -1, "59", NULL);
00322
00323   /* Read height range... */
00324   np0 = (int) scan_ctl(argc, argv, "NP_MIN", -1, "0", NULL);
00325   np1 = (int) scan_ctl(argc, argv, "NP_MAX", -1, "100", NULL);
00326   np1 = GSL_MIN(np1, nz - 1);
00327
00328   /* SZA threshold... */
00329   sza_thresh = scan_ctl(argc, argv, "SZA", -1, "96", NULL);
00330
00331   /* ------------------------------------------------------------
00332      Distribute granules...
00333      ------------------------------------------------------------ */
00334
00335   /* Open filelist... */
00336   printf("Read filelist: %s\n", argv[2]);
00337   if (!(in = fopen(argv[2], "r")))
00338     ERRMSG("Cannot open filelist!");
00339
00340   /* Loop over netCDF files... */
00341   while (fscanf(in, "%s", filename) != EOF) {
00342
00343     /* Distribute files with MPI... */
00344     if ((++ntask) % size != rank)
00345       continue;
00346
00347     /* Check task range... */
00348     if (ntask < task0 || ntask > task1)
00349       continue;
00350
00351     /* Write info... */
00352     printf("Retrieve file %s on rank %d of %d (with %d threads)...\n",
00353            filename, rank + 1, size, omp_get_max_threads());
00354
00355     /* ------------------------------------------------------------
00356        Initialize retrieval...
00357        ------------------------------------------------------------ */
00358
00359     /* Read netCDF file... */
00360     read_nc(filename, &ncd);
00361
00362     /* Adjust number of tracks... */
00363     if (track1 >= ncd.ntrack)
00364       track1 = ncd.ntrack - 1;
00365
00366     /* Identify radiance channels... */
00367     for (id = 0; id < ctl.nd; id++) {
00368       channel[id] = -999;
00369       for (i = 0; i < L1_NCHAN; i++)
00370         if (fabs(ctl.nu[id] - ncd.l1_nu[i]) < 0.1)
00371           channel[id] = i;
00372       if (channel[id] < 0)
00373         ERRMSG("Cannot identify radiance channel!");
00374     }
00375
00376     /* Set climatological data for center of granule... */
00377     atm_clim.np = nz;
00378     for (iz = 0; iz < nz; iz++)
00379       atm_clim.z[iz] = z[iz];
00380     climatology(&ctl, &atm_clim);
00381
00382     /* ------------------------------------------------------------
00383        Retrieval...
00384        ------------------------------------------------------------ */
00385
00386     /* Loop over swaths... */
00387     for (track = track0; track <= track1; track++) {
00388
00389       /* Loop over scan... */
00390       for (xtrack = xtrack0; xtrack <= xtrack1; xtrack++) {
00391
00392         /* Init timer... */
00393         t0 = omp_get_wtime();
00394
00395         /* Store observation data... */
00396         obs_meas.nr = 1;
00397         obs_meas.time[0] = ncd.l1_time[track][xtrack];
00398         obs_meas.obsz[0] = ncd.l1_sat_z[track];
00399         obs_meas.obslon[0] = ncd.l1_sat_lon[track];
00400         obs_meas.obslat[0] = ncd.l1_sat_lat[track];
00401         obs_meas.vplon[0] = ncd.l1_lon[track][xtrack];
00402         obs_meas.vplat[0] = ncd.l1_lat[track][xtrack];
```

```
00403         for (id = 0; id < ctl.nd; id++)
00404           obs_meas.rad[id][0] = ncd.l1_rad[track][xtrack][channel[id]];
00405
00406         /* Flag out 4 micron channels for daytime measurements... */
00407         if (sza(obs_meas.time[0], obs_meas.obslon[0], obs_meas.
      obslat[0])
00408             < sza_thresh)
00409           for (id = 0; id < ctl.nd; id++)
00410             if (ctl.nu[id] >= 2000)
00411               obs_meas.rad[id][0] = GSL_NAN;
00412
00413         /* Prepare atmospheric data... */
00414         copy_atm(&ctl, &atm_apr, &atm_clim, 0);
00415         for (ip = 0; ip < atm_apr.np; ip++) {
00416           atm_apr.time[ip] = obs_meas.time[0];
00417           atm_apr.lon[ip] = obs_meas.vplon[0];
00418           atm_apr.lat[ip] = obs_meas.vplat[0];
00419         }
00420
00421         /* Merge Level-2 data... */
00422         init_l2(&ncd, track, xtrack, &ctl, &atm_apr);
00423
00424         /* Retrieval... */
00425         optimal_estimation(&ret, &ctl, &obs_meas, &obs_i,
00426                            &atm_apr, &atm_i, &chisq);
00427
00428         /* Buffer results... */
00429         buffer_nc(&atm_i, chisq, &ncd, track, xtrack, np0, np1);
00430
00431         /* Write debug information... */
00432         if (debug >= 1)
00433           printf
00434             ("  task= %4d | track= %5d | xtrack= %3d | chi^2= %8.3f | time= %8.3f s\n",
00435              ntask, track, xtrack, chisq, omp_get_wtime() - t0);
00436         if (debug >= 2) {
00437           sprintf(filename2, "atm_apr_%d_%d_%d.tab", ntask, track, xtrack);
00438           write_atm(NULL, filename2, &ctl, &atm_apr);
00439           sprintf(filename2, "atm_i_%d_%d_%d.tab", ntask, track, xtrack);
00440           write_atm(NULL, filename2, &ctl, &atm_i);
00441           sprintf(filename2, "obs_meas_%d_%d_%d.tab", ntask, track, xtrack);
00442           write_obs(NULL, filename2, &ctl, &obs_meas);
00443           sprintf(filename2, "obs_i_%d_%d_%d.tab", ntask, track, xtrack);
00444           write_obs(NULL, filename2, &ctl, &obs_i);
00445         }
00446       }
00447     }
00448
00449     /* ------------------------------------------------------------
00450        Finalize...
00451        ------------------------------------------------------------ */
00452
00453     /* Write netCDF file... */
00454     write_nc(filename, &ncd);
00455
00456     /* Write info... */
00457     printf("Retrieval finished on rank %d of %d!\n", rank, size);
00458   }
00459
00460   /* Close file list... */
00461   fclose(in);
00462
00463   /* Measure CPU time... */
00464   TIMER("total", 3);
00465
00466   /* Report memory usage... */
00467   printf("MEMORY_ATM = %g MByte\n", 4. * sizeof(atm_t) / 1024. / 1024.);
00468   printf("MEMORY_CTL = %g MByte\n", 1. * sizeof(ctl_t) / 1024. / 1024.);
00469   printf("MEMORY_NCD = %g MByte\n", 1. * sizeof(ncd_t) / 1024. / 1024.);
00470   printf("MEMORY_OBS = %g MByte\n", 3. * sizeof(atm_t) / 1024. / 1024.);
00471   printf("MEMORY_RET = %g MByte\n", 1. * sizeof(ret_t) / 1024. / 1024.);
00472   printf("MEMORY_TBL = %g MByte\n", 1. * sizeof(tbl_t) / 1024. / 1024.);
00473
00474   /* Report problem size... */
00475   printf("SIZE_TASKS = %d\n", size);
00476   printf("SIZE_THREADS = %d\n", omp_get_max_threads());
00477
00478   /* MPI... */
00479   MPI_Finalize();
00480
00481   return EXIT_SUCCESS;
00482 }
00483
00484 /*****************************************************************************/
00485
00486 void add_var(
00487   int ncid,
00488   const char *varname,
```

```
00489   const char *unit,
00490   const char *longname,
00491   int type,
00492   int dimid[],
00493   int *varid,
00494   int ndims) {
00495
00496   /* Check if variable exists... */
00497   if (nc_inq_varid(ncid, varname, varid) != NC_NOERR) {
00498
00499     /* Define variable... */
00500     NC(nc_def_var(ncid, varname, type, ndims, dimid, varid));
00501
00502     /* Set long name... */
00503     NC(nc_put_att_text
00504        (ncid, *varid, "long_name", strlen(longname), longname));
00505
00506     /* Set units... */
00507     NC(nc_put_att_text(ncid, *varid, "units", strlen(unit), unit));
00508   }
00509 }
00510
00511 /*****************************************************************************/
00512
00513 void buffer_nc(
00514   atm_t * atm,
00515   double chisq,
00516   ncd_t * ncd,
00517   int track,
00518   int xtrack,
00519   int np0,
00520   int np1) {
00521
00522   int ip;
00523
00524   /* Set number of data points... */
00525   ncd->np = np1 - np0 + 1;
00526
00527   /* Save retrieval data... */
00528   ncd->ret_chisq[track * L1_NXTRACK + xtrack] = (float) chisq;
00529   ncd->ret_p[track * L1_NXTRACK + xtrack] = (float) atm->p[np0];
00530   for (ip = np0; ip <= np1; ip++) {
00531     ncd->ret_z[ip - np0] = (float) atm->z[ip];
00532     ncd->ret_t[(track * L1_NXTRACK + xtrack) * ncd->np + ip - np0] =
00533       (gsl_finite(chisq) ? (float) atm->t[ip] : GSL_NAN);
00534   }
00535 }
00536
00537 /*****************************************************************************/
00538
00539 double cost_function(
00540   gsl_vector * dx,
00541   gsl_vector * dy,
00542   gsl_matrix * s_a_inv,
00543   gsl_vector * sig_eps_inv) {
00544
00545   gsl_vector *x_aux, *y_aux;
00546
00547   double chisq_a, chisq_m = 0;
00548
00549   size_t i, m, n;
00550
00551   /* Get sizes... */
00552   m = dy->size;
00553   n = dx->size;
00554
00555   /* Allocate... */
00556   x_aux = gsl_vector_alloc(n);
00557   y_aux = gsl_vector_alloc(m);
00558
00559   /* Determine normalized cost function...
00560      (chi^2 = 1/m * [dy^T * S_eps^{-1} * dy + dx^T * S_a^{-1} * dx]) */
00561   for (i = 0; i < m; i++)
00562     chisq_m +=
00563       gsl_pow_2(gsl_vector_get(dy, i) * gsl_vector_get(sig_eps_inv, i));
00564   gsl_blas_dgemv(CblasNoTrans, 1.0, s_a_inv, dx, 0.0, x_aux);
00565   gsl_blas_ddot(dx, x_aux, &chisq_a);
00566
00567   /* Free... */
00568   gsl_vector_free(x_aux);
00569   gsl_vector_free(y_aux);
00570
00571   /* Return cost function value... */
00572   return (chisq_m + chisq_a) / (double) m;
00573 }
00574
00575 /*****************************************************************************/
```

```
00576
00577 void init_l2(
00578   ncd_t * ncd,
00579   int track,
00580   int xtrack,
00581   ctl_t * ctl,
00582   atm_t * atm) {
00583
00584   static atm_t atm_iasi;
00585
00586   double k[NW], p, q[NG], t, w, zmax = 0, zmin = 1000;
00587
00588   int ip, lay;
00589
00590   /* Store IASI data in atmospheric data struct... */
00591   atm_iasi.np = 0;
00592   for (lay = 0; lay < L2_NLAY; lay++)
00593     if (gsl_finite(ncd->l2_z[track][xtrack][lay])
00594         && ncd->l2_z[track][xtrack][lay] <= 60.) {
00595       atm_iasi.z[atm_iasi.np] = ncd->l2_z[track][xtrack][lay];
00596       atm_iasi.p[atm_iasi.np] = ncd->l2_p[lay];
00597       atm_iasi.t[atm_iasi.np] = ncd->l2_t[track][xtrack][lay];
00598       if ((++atm_iasi.np) > NP)
00599         ERRMSG("Too many layers!");
00600     }
00601
00602   /* Check number of levels... */
00603   if (atm_iasi.np < 2)
00604     return;
00605
00606   /* Get height range of IASI data... */
00607   for (ip = 0; ip < atm_iasi.np; ip++) {
00608     zmax = GSL_MAX(zmax, atm_iasi.z[ip]);
00609     zmin = GSL_MIN(zmin, atm_iasi.z[ip]);
00610   }
00611
00612   /* Merge IASI data... */
00613   for (ip = 0; ip < atm->np; ip++) {
00614
00615     /* Interpolate IASI data... */
00616     intpol_atm(ctl, &atm_iasi, atm->z[ip], &p, &t, q, k);
00617
00618     /* Weighting factor... */
00619     w = 1;
00620     if (atm->z[ip] > zmax)
00621       w = GSL_MAX(1 - (atm->z[ip] - zmax) / 50, 0);
00622     if (atm->z[ip] < zmin)
00623       w = GSL_MAX(1 - (zmin - atm->z[ip]) / 50, 0);
00624
00625     /* Merge... */
00626     atm->t[ip] = w * t + (1 - w) * atm->t[ip];
00627     atm->p[ip] = w * p + (1 - w) * atm->p[ip];
00628   }
00629 }
00630
00631 /*****************************************************************************/
00632
00633 void matrix_invert(
00634   gsl_matrix * a) {
00635
00636   size_t diag = 1, i, j, n;
00637
00638   /* Get size... */
00639   n = a->size1;
00640
00641   /* Check if matrix is diagonal... */
00642   for (i = 0; i < n && diag; i++)
00643     for (j = i + 1; j < n; j++)
00644       if (gsl_matrix_get(a, i, j) != 0) {
00645         diag = 0;
00646         break;
00647       }
00648
00649   /* Quick inversion of diagonal matrix... */
00650   if (diag)
00651     for (i = 0; i < n; i++)
00652       gsl_matrix_set(a, i, i, 1 / gsl_matrix_get(a, i, i));
00653
00654   /* Matrix inversion by means of Cholesky decomposition... */
00655   else {
00656     gsl_linalg_cholesky_decomp(a);
00657     gsl_linalg_cholesky_invert(a);
00658   }
00659 }
00660
00661 /*****************************************************************************/
00662
```

```
00663 void matrix_product(
00664   gsl_matrix * a,
00665   gsl_vector * b,
00666   int transpose,
00667   gsl_matrix * c) {
00668
00669   gsl_matrix *aux;
00670
00671   size_t i, j, m, n;
00672
00673   /* Set sizes... */
00674   m = a->size1;
00675   n = a->size2;
00676
00677   /* Allocate... */
00678   aux = gsl_matrix_alloc(m, n);
00679
00680   /* Compute A^T B A... */
00681   if (transpose == 1) {
00682
00683     /* Compute B^1/2 A... */
00684     for (i = 0; i < m; i++)
00685       for (j = 0; j < n; j++)
00686         gsl_matrix_set(aux, i, j,
00687                        gsl_vector_get(b, i) * gsl_matrix_get(a, i, j));
00688
00689     /* Compute A^T B A = (B^1/2 A)^T (B^1/2 A)... */
00690     gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, aux, aux, 0.0, c);
00691   }
00692
00693   /* Compute A B A^T... */
00694   else if (transpose == 2) {
00695
00696     /* Compute A B^1/2... */
00697     for (i = 0; i < m; i++)
00698       for (j = 0; j < n; j++)
00699         gsl_matrix_set(aux, i, j,
00700                        gsl_matrix_get(a, i, j) * gsl_vector_get(b, j));
00701
00702     /* Compute A B A^T = (A B^1/2) (A B^1/2)^T... */
00703     gsl_blas_dgemm(CblasNoTrans, CblasTrans, 1.0, aux, aux, 0.0, c);
00704   }
00705
00706   /* Free... */
00707   gsl_matrix_free(aux);
00708 }
00709
00710 /*****************************************************************************/
00711
00712 void optimal_estimation(
00713   ret_t * ret,
00714   ctl_t * ctl,
00715   obs_t * obs_meas,
00716   obs_t * obs_i,
00717   atm_t * atm_apr,
00718   atm_t * atm_i,
00719   double *chisq) {
00720
00721   static int ipa[N], iqa[N];
00722
00723   gsl_matrix *a, *cov, *k_i, *s_a_inv;
00724
00725   gsl_vector *b, *dx, *dy, *sig_eps_inv, *sig_formod, *sig_noise,
00726     *x_a, *x_i, *x_step, *y_aux, *y_i, *y_m;
00727
00728   double chisq_old, disq = 0, lmpar = 0.001;
00729
00730   int ig, ip, it = 0, it2, iw;
00731
00732   size_t i, m, n;
00733
00734   /* ------------------------------------------------------------
00735      Initialize...
00736      ------------------------------------------------------------ */
00737
00738   /* Get sizes... */
00739   m = obs2y(ctl, obs_meas, NULL, NULL, NULL);
00740   n = atm2x(ctl, atm_apr, NULL, iqa, ipa);
00741   if (m <= 0 || n <= 0) {
00742     *chisq = GSL_NAN;
00743     return;
00744   }
00745
00746   /* Allocate... */
00747   a = gsl_matrix_alloc(n, n);
00748   cov = gsl_matrix_alloc(n, n);
00749   k_i = gsl_matrix_alloc(m, n);
```

```
00750   s_a_inv = gsl_matrix_alloc(n, n);
00751
00752   b = gsl_vector_alloc(n);
00753   dx = gsl_vector_alloc(n);
00754   dy = gsl_vector_alloc(m);
00755   sig_eps_inv = gsl_vector_alloc(m);
00756   sig_formod = gsl_vector_alloc(m);
00757   sig_noise = gsl_vector_alloc(m);
00758   x_a = gsl_vector_alloc(n);
00759   x_i = gsl_vector_alloc(n);
00760   x_step = gsl_vector_alloc(n);
00761   y_aux = gsl_vector_alloc(m);
00762   y_i = gsl_vector_alloc(m);
00763   y_m = gsl_vector_alloc(m);
00764
00765   /* Set initial state... */
00766   copy_atm(ctl, atm_i, atm_apr, 0);
00767   copy_obs(ctl, obs_i, obs_meas, 0);
00768   formod(ctl, atm_i, obs_i);
00769
00770   /* Set state vectors and observation vectors... */
00771   atm2x(ctl, atm_apr, x_a, NULL, NULL);
00772   atm2x(ctl, atm_i, x_i, NULL, NULL);
00773   obs2y(ctl, obs_meas, y_m, NULL, NULL);
00774   obs2y(ctl, obs_i, y_i, NULL, NULL);
00775
00776   /* Set inverse a priori covariance S_a^-1... */
00777   set_cov_apr(ret, ctl, atm_apr, iqa, ipa, s_a_inv);
00778   matrix_invert(s_a_inv);
00779
00780   /* Get measurement errors... */
00781   set_cov_meas(ret, ctl, obs_meas, sig_noise, sig_formod, sig_eps_inv);
00782
00783   /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00784   gsl_vector_memcpy(dx, x_i);
00785   gsl_vector_sub(dx, x_a);
00786   gsl_vector_memcpy(dy, y_m);
00787   gsl_vector_sub(dy, y_i);
00788
00789   /* Compute cost function... */
00790   *chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00791
00792   /* Compute initial kernel... */
00793   kernel(ctl, atm_i, obs_i, k_i);
00794
00795   /* ------------------------------------------------------------
00796      Levenberg-Marquardt minimization...
00797      ------------------------------------------------------------ */
00798
00799   /* Outer loop... */
00800   for (it = 1; it <= ret->conv_itmax; it++) {
00801
00802     /* Store current cost function value... */
00803     chisq_old = *chisq;
00804
00805     /* Compute kernel matrix K_i... */
00806     if (it > 1 && it % ret->kernel_recomp == 0)
00807       kernel(ctl, atm_i, obs_i, k_i);
00808
00809     /* Compute K_i^T * S_eps^{-1} * K_i ... */
00810     if (it == 1 || it % ret->kernel_recomp == 0)
00811       matrix_product(k_i, sig_eps_inv, 1, cov);
00812
00813     /* Determine b = K_i^T * S_eps^{-1} * dy - S_a^{-1} * dx ... */
00814     for (i = 0; i < m; i++)
00815       gsl_vector_set(y_aux, i, gsl_vector_get(dy, i)
00816                      * gsl_pow_2(gsl_vector_get(sig_eps_inv, i)));
00817     gsl_blas_dgemv(CblasTrans, 1.0, k_i, y_aux, 0.0, b);
00818     gsl_blas_dgemv(CblasNoTrans, -1.0, s_a_inv, dx, 1.0, b);
00819
00820     /* Inner loop... */
00821     for (it2 = 0; it2 < 20; it2++) {
00822
00823       /* Compute A = (1 + lmpar) * S_a^{-1} + K_i^T * S_eps^{-1} * K_i ... */
00824       gsl_matrix_memcpy(a, s_a_inv);
00825       gsl_matrix_scale(a, 1 + lmpar);
00826       gsl_matrix_add(a, cov);
00827
00828       /* Solve A * x_step = b by means of Cholesky decomposition... */
00829       gsl_linalg_cholesky_decomp(a);
00830       gsl_linalg_cholesky_solve(a, b, x_step);
00831
00832       /* Update atmospheric state... */
00833       gsl_vector_add(x_i, x_step);
00834       copy_atm(ctl, atm_i, atm_apr, 0);
00835       copy_obs(ctl, obs_i, obs_meas, 0);
00836       x2atm(ctl, x_i, atm_i);
```

```
00837
00838       /* Check atmospheric state... */
00839       for (ip = 0; ip < atm_i->np; ip++) {
00840         atm_i->p[ip] = GSL_MIN(GSL_MAX(atm_i->p[ip], 5e-7), 5e4);
00841         atm_i->t[ip] = GSL_MIN(GSL_MAX(atm_i->t[ip], 100), 400);
00842         for (ig = 0; ig < ctl->ng; ig++)
00843           atm_i->q[ig][ip] = GSL_MIN(GSL_MAX(atm_i->q[ig][ip], 0), 1);
00844         for (iw = 0; iw < ctl->nw; iw++)
00845           atm_i->k[iw][ip] = GSL_MAX(atm_i->k[iw][ip], 0);
00846       }
00847
00848       /* Forward calculation... */
00849       formod(ctl, atm_i, obs_i);
00850       obs2y(ctl, obs_i, y_i, NULL, NULL);
00851
00852       /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00853       gsl_vector_memcpy(dx, x_i);
00854       gsl_vector_sub(dx, x_a);
00855       gsl_vector_memcpy(dy, y_m);
00856       gsl_vector_sub(dy, y_i);
00857
00858       /* Compute cost function... */
00859       *chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00860
00861       /* Modify Levenberg-Marquardt parameter... */
00862       if (*chisq > chisq_old) {
00863         lmpar *= 10;
00864         gsl_vector_sub(x_i, x_step);
00865       } else {
00866         lmpar /= 10;
00867         break;
00868       }
00869     }
00870
00871     /* Get normalized step size in state space... */
00872     gsl_blas_ddot(x_step, b, &disq);
00873     disq /= (double) n;
00874
00875     /* Convergence test... */
00876     if ((it == 1 || it % ret->kernel_recomp == 0) && disq < ret->
      conv_dmin)
00877       break;
00878   }
00879
00880   /* ------------------------------------------------------------
00881      Finalize...
00882      ------------------------------------------------------------ */
00883
00884   gsl_matrix_free(a);
00885   gsl_matrix_free(cov);
00886   gsl_matrix_free(k_i);
00887   gsl_matrix_free(s_a_inv);
00888
00889   gsl_vector_free(b);
00890   gsl_vector_free(dx);
00891   gsl_vector_free(dy);
00892   gsl_vector_free(sig_eps_inv);
00893   gsl_vector_free(sig_formod);
00894   gsl_vector_free(sig_noise);
00895   gsl_vector_free(x_a);
00896   gsl_vector_free(x_i);
00897   gsl_vector_free(x_step);
00898   gsl_vector_free(y_aux);
00899   gsl_vector_free(y_i);
00900   gsl_vector_free(y_m);
00901 }
00902
00903 /*****************************************************************************/
00904
00905 void read_nc(
00906   char *filename,
00907   ncd_t * ncd) {
00908
00909   int dimid, varid;
00910
00911   size_t len;
00912
00913   /* Open netCDF file... */
00914   printf("Read netCDF file: %s\n", filename);
00915   NC(nc_open(filename, NC_WRITE, &ncd->ncid));
00916
00917   /* Read number of tracks... */
00918   NC(nc_inq_dimid(ncd->ncid, "L1_NTRACK", &dimid));
00919   NC(nc_inq_dimlen(ncd->ncid, dimid, &len));
00920   ncd->ntrack = (int) len;
00921
00922   /* Read Level-1 data... */
```

```
00923   NC(nc_inq_varid(ncd->ncid, "l1_time", &varid));
00924   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_time[0]));
00925   NC(nc_inq_varid(ncd->ncid, "l1_lon", &varid));
00926   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_lon[0]));
00927   NC(nc_inq_varid(ncd->ncid, "l1_lat", &varid));
00928   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_lat[0]));
00929   NC(nc_inq_varid(ncd->ncid, "l1_sat_z", &varid));
00930   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_z));
00931   NC(nc_inq_varid(ncd->ncid, "l1_sat_lon", &varid));
00932   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_lon));
00933   NC(nc_inq_varid(ncd->ncid, "l1_sat_lat", &varid));
00934   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_sat_lat));
00935   NC(nc_inq_varid(ncd->ncid, "l1_nu", &varid));
00936   NC(nc_get_var_double(ncd->ncid, varid, ncd->l1_nu));
00937   NC(nc_inq_varid(ncd->ncid, "l1_rad", &varid));
00938   NC(nc_get_var_float(ncd->ncid, varid, ncd->l1_rad[0][0]));
00939
00940   /* Read Level-2 data... */
00941   NC(nc_inq_varid(ncd->ncid, "l2_z", &varid));
00942   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_z[0][0]));
00943   NC(nc_inq_varid(ncd->ncid, "l2_press", &varid));
00944   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_p));
00945   NC(nc_inq_varid(ncd->ncid, "l2_temp", &varid));
00946   NC(nc_get_var_double(ncd->ncid, varid, ncd->l2_t[0][0]));
00947 }
00948
00949 /*****************************************************************************/
00950
00951 void read_ret_ctl(
00952   int argc,
00953   char *argv[],
00954   ctl_t * ctl,
00955   ret_t * ret) {
00956
00957   int id, ig, iw;
00958
00959   /* Iteration control... */
00960   ret->kernel_recomp =
00961     (int) scan_ctl(argc, argv, "KERNEL_RECOMP", -1, "3", NULL);
00962   ret->conv_itmax = (int) scan_ctl(argc, argv, "CONV_ITMAX", -1, "30", NULL);
00963   ret->conv_dmin = scan_ctl(argc, argv, "CONV_DMIN", -1, "0.1", NULL);
00964
00965   for (id = 0; id < ctl->nd; id++)
00966     ret->err_formod[id] = scan_ctl(argc, argv, "ERR_FORMOD", id, "0", NULL);
00967
00968   for (id = 0; id < ctl->nd; id++)
00969     ret->err_noise[id] = scan_ctl(argc, argv, "ERR_NOISE", id, "0", NULL);
00970
00971   ret->err_press = scan_ctl(argc, argv, "ERR_PRESS", -1, "0", NULL);
00972   ret->err_press_cz = scan_ctl(argc, argv, "ERR_PRESS_CZ", -1, "-999", NULL);
00973   ret->err_press_ch = scan_ctl(argc, argv, "ERR_PRESS_CH", -1, "-999", NULL);
00974
00975   ret->err_temp = scan_ctl(argc, argv, "ERR_TEMP", -1, "0", NULL);
00976   ret->err_temp_cz = scan_ctl(argc, argv, "ERR_TEMP_CZ", -1, "-999", NULL);
00977   ret->err_temp_ch = scan_ctl(argc, argv, "ERR_TEMP_CH", -1, "-999", NULL);
00978
00979   for (ig = 0; ig < ctl->ng; ig++) {
00980     ret->err_q[ig] = scan_ctl(argc, argv, "ERR_Q", ig, "0", NULL);
00981     ret->err_q_cz[ig] = scan_ctl(argc, argv, "ERR_Q_CZ", ig, "-999", NULL);
00982     ret->err_q_ch[ig] = scan_ctl(argc, argv, "ERR_Q_CH", ig, "-999", NULL);
00983   }
00984
00985   for (iw = 0; iw < ctl->nw; iw++) {
00986     ret->err_k[iw] = scan_ctl(argc, argv, "ERR_K", iw, "0", NULL);
00987     ret->err_k_cz[iw] = scan_ctl(argc, argv, "ERR_K_CZ", iw, "-999", NULL);
00988     ret->err_k_ch[iw] = scan_ctl(argc, argv, "ERR_K_CH", iw, "-999", NULL);
00989   }
00990 }
00991
00992 /*****************************************************************************/
00993
00994 void set_cov_apr(
00995   ret_t * ret,
00996   ctl_t * ctl,
00997   atm_t * atm,
00998   int *iqa,
00999   int *ipa,
01000   gsl_matrix * s_a) {
01001
01002   gsl_vector *x_a;
01003
01004   double ch, cz, rho, x0[3], x1[3];
01005
01006   int ig, iw;
01007
01008   size_t i, j, n;
01009
```

```
01010   /* Get sizes... */
01011   n = s_a->size1;
01012
01013   /* Allocate... */
01014   x_a = gsl_vector_alloc(n);
01015
01016   /* Get sigma vector... */
01017   atm2x(ctl, atm, x_a, NULL, NULL);
01018   for (i = 0; i < n; i++) {
01019     if (iqa[i] == IDXP)
01020       gsl_vector_set(x_a, i, ret->err_press / 100 * gsl_vector_get(x_a, i));
01021     if (iqa[i] == IDXT)
01022       gsl_vector_set(x_a, i, ret->err_temp);
01023     for (ig = 0; ig < ctl->ng; ig++)
01024       if (iqa[i] == IDXQ(ig))
01025         gsl_vector_set(x_a, i, ret->err_q[ig] / 100 * gsl_vector_get(x_a, i));
01026     for (iw = 0; iw < ctl->nw; iw++)
01027       if (iqa[i] == IDXK(iw))
01028         gsl_vector_set(x_a, i, ret->err_k[iw]);
01029   }
01030
01031   /* Check standard deviations... */
01032   for (i = 0; i < n; i++)
01033     if (gsl_pow_2(gsl_vector_get(x_a, i)) <= 0)
01034       ERRMSG("Check a priori data (zero standard deviation)!");
01035
01036   /* Initialize diagonal covariance... */
01037   gsl_matrix_set_zero(s_a);
01038   for (i = 0; i < n; i++)
01039     gsl_matrix_set(s_a, i, i, gsl_pow_2(gsl_vector_get(x_a, i)));
01040
01041   /* Loop over matrix elements... */
01042   for (i = 0; i < n; i++)
01043     for (j = 0; j < n; j++)
01044       if (i != j && iqa[i] == iqa[j]) {
01045
01046         /* Initialize... */
01047         cz = ch = 0;
01048
01049         /* Set correlation lengths for pressure... */
01050         if (iqa[i] == IDXP) {
01051           cz = ret->err_press_cz;
01052           ch = ret->err_press_ch;
01053         }
01054
01055         /* Set correlation lengths for temperature... */
01056         if (iqa[i] == IDXT) {
01057           cz = ret->err_temp_cz;
01058           ch = ret->err_temp_ch;
01059         }
01060
01061         /* Set correlation lengths for volume mixing ratios... */
01062         for (ig = 0; ig < ctl->ng; ig++)
01063           if (iqa[i] == IDXQ(ig)) {
01064             cz = ret->err_q_cz[ig];
01065             ch = ret->err_q_ch[ig];
01066           }
01067
01068         /* Set correlation lengths for extinction... */
01069         for (iw = 0; iw < ctl->nw; iw++)
01070           if (iqa[i] == IDXK(iw)) {
01071             cz = ret->err_k_cz[iw];
01072             ch = ret->err_k_ch[iw];
01073           }
01074
01075         /* Compute correlations... */
01076         if (cz > 0 && ch > 0) {
01077
01078           /* Get Cartesian coordinates... */
01079           geo2cart(0, atm->lon[ipa[i]], atm->lat[ipa[i]], x0);
01080           geo2cart(0, atm->lon[ipa[j]], atm->lat[ipa[j]], x1);
01081
01082           /* Compute correlations... */
01083           rho =
01084             exp(-DIST(x0, x1) / ch -
01085                 fabs(atm->z[ipa[i]] - atm->z[ipa[j]]) / cz);
01086
01087           /* Set covariance... */
01088           gsl_matrix_set(s_a, i, j, gsl_vector_get(x_a, i)
01089                          * gsl_vector_get(x_a, j) * rho);
01090         }
01091       }
01092
01093   /* Free... */
01094   gsl_vector_free(x_a);
01095 }
01096
```

```
01097 /*****************************************************************************/
01098
01099 void set_cov_meas(
01100   ret_t * ret,
01101   ctl_t * ctl,
01102   obs_t * obs,
01103   gsl_vector * sig_noise,
01104   gsl_vector * sig_formod,
01105   gsl_vector * sig_eps_inv) {
01106
01107   static obs_t obs_err;
01108
01109   int id, ir;
01110
01111   size_t i, m;
01112
01113   /* Get size... */
01114   m = sig_eps_inv->size;
01115
01116   /* Noise error (always considered in retrieval fit)... */
01117   copy_obs(ctl, &obs_err, obs, 1);
01118   for (ir = 0; ir < obs_err.nr; ir++)
01119     for (id = 0; id < ctl->nd; id++)
01120       obs_err.rad[id][ir]
01121         = (gsl_finite(obs->rad[id][ir]) ? ret->err_noise[id] : GSL_NAN);
01122   obs2y(ctl, &obs_err, sig_noise, NULL, NULL);
01123
01124   /* Forward model error (always considered in retrieval fit)... */
01125   copy_obs(ctl, &obs_err, obs, 1);
01126   for (ir = 0; ir < obs_err.nr; ir++)
01127     for (id = 0; id < ctl->nd; id++)
01128       obs_err.rad[id][ir]
01129         = fabs(ret->err_formod[id] / 100 * obs->rad[id][ir]);
01130   obs2y(ctl, &obs_err, sig_formod, NULL, NULL);
01131
01132   /* Total error... */
01133   for (i = 0; i < m; i++)
01134     gsl_vector_set(sig_eps_inv, i,
01135                    1 / sqrt(gsl_pow_2(gsl_vector_get(sig_noise, i))
01136                             + gsl_pow_2(gsl_vector_get(sig_formod, i))));
01137
01138   /* Check standard deviations... */
01139   for (i = 0; i < m; i++)
01140     if (gsl_vector_get(sig_eps_inv, i) <= 0)
01141       ERRMSG("Check measurement errors (zero standard deviation)!");
01142 }
01143
01144 /*****************************************************************************/
01145
01146 double sza(
01147   double sec,
01148   double lon,
01149   double lat) {
01150
01151   double D, dec, e, g, GMST, h, L, LST, q, ra;
01152
01153   /* Number of days and fraction with respect to 2000-01-01T12:00Z... */
01154   D = sec / 86400 - 0.5;
01155
01156   /* Geocentric apparent ecliptic longitude [rad]... */
01157   g = (357.529 + 0.98560028 * D) * M_PI / 180;
01158   q = 280.459 + 0.98564736 * D;
01159   L = (q + 1.915 * sin(g) + 0.020 * sin(2 * g)) * M_PI / 180;
01160
01161   /* Mean obliquity of the ecliptic [rad]... */
01162   e = (23.439 - 0.00000036 * D) * M_PI / 180;
01163
01164   /* Declination [rad]... */
01165   dec = asin(sin(e) * sin(L));
01166
01167   /* Right ascension [rad]... */
01168   ra = atan2(cos(e) * sin(L), cos(L));
01169
01170   /* Greenwich Mean Sidereal Time [h]... */
01171   GMST = 18.697374558 + 24.06570982441908 * D;
01172
01173   /* Local Sidereal Time [h]... */
01174   LST = GMST + lon / 15;
01175
01176   /* Hour angle [rad]... */
01177   h = LST / 12 * M_PI - ra;
01178
01179   /* Convert latitude... */
01180   lat *= M_PI / 180;
01181
01182   /* Return solar zenith angle [deg]... */
01183   return acos(sin(lat) * sin(dec) +
```

```
01184               cos(lat) * cos(dec) * cos(h)) * 180 / M_PI;
01185 }
01186
01187 /*****************************************************************************/
01188
01189 void write_nc(
01190   char *filename,
01191   ncd_t * ncd) {
01192
01193   int dimid[10], c_id, p_id, t_id, z_id;
01194
01195   /* Create netCDF file... */
01196   printf("Write netCDF file: %s\n", filename);
01197
01198   /* Read existing dimensions... */
01199   NC(nc_inq_dimid(ncd->ncid, "L1_NTRACK", &dimid[0]));
01200   NC(nc_inq_dimid(ncd->ncid, "L1_NXTRACK", &dimid[1]));
01201
01202   /* Set define mode... */
01203   NC(nc_redef(ncd->ncid));
01204
01205   /* Set new dimensions... */
01206   if (nc_inq_dimid(ncd->ncid, "RET_NP", &dimid[2]) != NC_NOERR)
01207     NC(nc_def_dim(ncd->ncid, "RET_NP", (size_t) ncd->np, &dimid[2]));
01208
01209   /* Set new variables... */
01210   add_var(ncd->ncid, "ret_z", "km", "altitude", NC_FLOAT, &dimid[2], &z_id,
01211           1);
01212   add_var(ncd->ncid, "ret_press", "hPa", "pressure", NC_FLOAT, dimid, &p_id,
01213           2);
01214   add_var(ncd->ncid, "ret_temp", "K", "temperature", NC_FLOAT, dimid, &t_id,
01215           3);
01216   add_var(ncd->ncid, "ret_chisq", "1", "chi^2 value of fit", NC_FLOAT, dimid,
01217           &c_id, 2);
01218
01219   /* Leave define mode... */
01220   NC(nc_enddef(ncd->ncid));
01221
01222   /* Write data... */
01223   NC(nc_put_var_float(ncd->ncid, z_id, ncd->ret_z));
01224   NC(nc_put_var_float(ncd->ncid, p_id, ncd->ret_p));
01225   NC(nc_put_var_float(ncd->ncid, t_id, ncd->ret_t));
01226   NC(nc_put_var_float(ncd->ncid, c_id, ncd->ret_chisq));
01227
01228   /* Close netCDF file... */
01229   NC(nc_close(ncd->ncid));
01230 }
```

5.19 spec2tab.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

5.19.1 Function Documentation

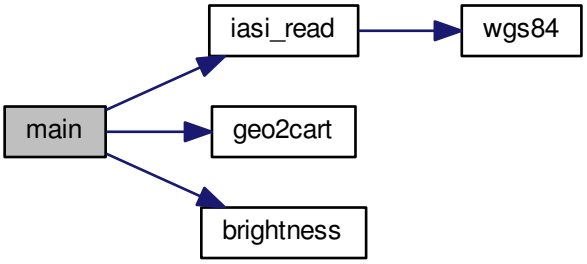
5.19.1.1 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 3 of file spec2tab.c.

```
00005                 {
00006
00007   static iasi_rad_t *iasi_rad;
00008
00009   FILE *out;
00010
00011   double dmin = 1e100, x0[3], x1[3];
00012
00013   int ichan, track = -1, track2, xtrack = -1, xtrack2;
00014
00015   /* Check arguments... */
00016   if (argc != 6)
00017     ERRMSG("Give parameters: <iasi_l1b_file> "
00018            "[index <track> <xtrack> | geo <lon> <lat>] <spec.tab>");
```

```
00019
00020   /* Allocate... */
00021   ALLOC(iasi_rad, iasi_rad_t, 1);
00022
00023   /* Read IASI data... */
00024   printf("Read IASI Level-1C data file: %s\n", argv[1]);
00025   iasi_read(argv[1], iasi_rad);
00026
00027   /* Get indices... */
00028   if (argv[2][0] == 'i') {
00029     track = atoi(argv[3]);
00030     xtrack = atoi(argv[4]);
00031   }
00032
00033   /* Find nearest footprint... */
00034   else {
00035     geo2cart(0, atof(argv[3]), atof(argv[4]), x0);
00036     for (track2 = 0; track2 < iasi_rad->ntrack; track2++)
00037       for (xtrack2 = 0; xtrack2 < L1_NXTRACK; xtrack2++) {
00038         geo2cart(0, iasi_rad->Longitude[track2][xtrack2],
00039                  iasi_rad->Latitude[track2][xtrack2], x1);
00040         if (DIST2(x0, x1) < dmin) {
00041           dmin = DIST2(x0, x1);
00042           track = track2;
00043           xtrack = xtrack2;
00044         }
00045       }
00046     if (dmin > 2500)
00047       ERRMSG("Geolocation not covered by granule!");
00048   }
00049
00050   /* Check indices... */
00051   if (track < 0 || track >= iasi_rad->ntrack)
00052     ERRMSG("Along-track index out of range!");
00053   if (xtrack < 0 || xtrack >= L1_NXTRACK)
00054     ERRMSG("Across-track index out of range!");
00055
00056   /* Create file... */
00057   printf("Write spectrum: %s\n", argv[5]);
00058   if (!(out = fopen(argv[5], "w")))
00059     ERRMSG("Cannot create file!");
00060
00061   /* Write header... */
00062   fprintf(out,
00063           "# $1 = time (seconds since 01-JAN-2000, 00:00 UTC)\n"
00064           "# $2 = satellite longitude [deg]\n"
00065           "# $3 = satellite latitude [deg]\n"
00066           "# $4 = footprint longitude [deg]\n"
00067           "# $5 = footprint latitude [deg]\n"
00068           "# $6 = wavenumber [cm^-1]\n"
00069           "# $7 = brightness temperature [K]\n"
00070           "# $8 = radiance [W/(m^2 sr cm^-1)]\n\n");
00071
00072   /* Write data... */
00073   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00074     fprintf(out, "%.2f %g %g %g %g %g %g %g\n",
00075             iasi_rad->Time[track][xtrack],
00076             iasi_rad->Sat_lon[track],
00077             iasi_rad->Sat_lat[track],
00078             iasi_rad->Longitude[track][xtrack],
00079             iasi_rad->Latitude[track][xtrack],
00080             iasi_rad->freq[ichan],
00081             brightness(iasi_rad->Rad[track][xtrack][ichan],
00082                        iasi_rad->freq[ichan]),
00083             iasi_rad->Rad[track][xtrack][ichan]);
00084
00085   /* Close file... */
00086   fclose(out);
00087
00088   /* Free... */
00089   free(iasi_rad);
00090
00091   return EXIT_SUCCESS;
00092 }
```

Here is the call graph for this function:



5.20 spec2tab.c

```
00001 #include "libiasi.h"
00002
00003 int main(
00004   int argc,
00005   char *argv[]) {
00006
00007   static iasi_rad_t *iasi_rad;
00008
00009   FILE *out;
00010
00011   double dmin = 1e100, x0[3], x1[3];
00012
00013   int ichan, track = -1, track2, xtrack = -1, xtrack2;
00014
00015   /* Check arguments... */
00016   if (argc != 6)
00017     ERRMSG("Give parameters: <iasi_l1b_file> "
00018            "[index <track> <xtrack> | geo <lon> <lat>] <spec.tab>");
00019
00020   /* Allocate... */
00021   ALLOC(iasi_rad, iasi_rad_t, 1);
00022
00023   /* Read IASI data... */
00024   printf("Read IASI Level-1C data file: %s\n", argv[1]);
00025   iasi_read(argv[1], iasi_rad);
00026
00027   /* Get indices... */
00028   if (argv[2][0] == 'i') {
00029     track = atoi(argv[3]);
00030     xtrack = atoi(argv[4]);
00031   }
00032
00033   /* Find nearest footprint... */
00034   else {
00035     geo2cart(0, atof(argv[3]), atof(argv[4]), x0);
00036     for (track2 = 0; track2 < iasi_rad->ntrack; track2++)
00037       for (xtrack2 = 0; xtrack2 < L1_NXTRACK; xtrack2++) {
00038         geo2cart(0, iasi_rad->Longitude[track2][xtrack2],
00039                  iasi_rad->Latitude[track2][xtrack2], x1);
00040         if (DIST2(x0, x1) < dmin) {
00041           dmin = DIST2(x0, x1);
00042           track = track2;
00043           xtrack = xtrack2;
00044         }
00045       }
00046     if (dmin > 2500)
00047       ERRMSG("Geolocation not covered by granule!");
00048   }
00049
00050   /* Check indices... */
00051   if (track < 0 || track >= iasi_rad->ntrack)
00052     ERRMSG("Along-track index out of range!");
00053   if (xtrack < 0 || xtrack >= L1_NXTRACK)
00054     ERRMSG("Across-track index out of range!");
```

```
00055
00056   /* Create file... */
00057   printf("Write spectrum: %s\n", argv[5]);
00058   if (!(out = fopen(argv[5], "w")))
00059     ERRMSG("Cannot create file!");
00060
00061   /* Write header... */
00062   fprintf(out,
00063           "# $1 = time (seconds since 01-JAN-2000, 00:00 UTC)\n"
00064           "# $2 = satellite longitude [deg]\n"
00065           "# $3 = satellite latitude [deg]\n"
00066           "# $4 = footprint longitude [deg]\n"
00067           "# $5 = footprint latitude [deg]\n"
00068           "# $6 = wavenumber [cm^-1]\n"
00069           "# $7 = brightness temperature [K]\n"
00070           "# $8 = radiance [W/(m^2 sr cm^-1)]\n\n");
00071
00072   /* Write data... */
00073   for (ichan = 0; ichan < IASI_L1_NCHAN; ichan++)
00074     fprintf(out, "%.2f %g %g %g %g %g %g %g\n",
00075             iasi_rad->Time[track][xtrack],
00076             iasi_rad->Sat_lon[track],
00077             iasi_rad->Sat_lat[track],
00078             iasi_rad->Longitude[track][xtrack],
00079             iasi_rad->Latitude[track][xtrack],
00080             iasi_rad->freq[ichan],
00081             brightness(iasi_rad->Rad[track][xtrack][ichan],
00082                        iasi_rad->freq[ichan]),
00083             iasi_rad->Rad[track][xtrack][ichan]);
00084
00085   /* Close file... */
00086   fclose(out);
00087
00088   /* Free... */
00089   free(iasi_rad);
00090
00091   return EXIT_SUCCESS;
00092 }
```

Index