MPTRAC

Generated by Doxygen 1.8.11

Contents

1 Main Page

Massive-Parallel Trajectory Calculations (MPTRAC) is a Lagrangian particle dispersion model for the troposphere and stratosphere.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/mptrac

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 <libtrac.h>
```

Data Fields

- int np

  Number of air pacels.
- double time [NP]

  Time [s].
- double p [NP]

  Pressure [hPa].
- double lon [NP]

  Longitude [deg].
- double lat [NP]

  Latitude [deg].
- double q [NQ][NP]

Quantitiy data (for various, user-defined attributes).

- double up [NP]

  Zonal wind perturbation [m/s].

- double vp [NP]

  Meridional wind perturbation [m/s].

- double wp [NP]

  Vertical velocity perturbation [hPa/s].

4.1.1 Detailed Description

Atmospheric data.

Definition at line 456 of file libtrac.h.

4.1.2 Field Documentation

4.1.2.1 int atm_t::np

Number of air pacels.

Definition at line 459 of file libtrac.h.

4.1.2.2 double atm_t::time[NP]

Time [s].

Definition at line 462 of file libtrac.h.

4.1.2.3 double atm_t::p[NP]

Pressure [hPa].

Definition at line 465 of file libtrac.h.

4.1.2.4 double atm_t::lon[NP]

Longitude [deg].

Definition at line 468 of file libtrac.h.

4.1.2.5 double atm_t::lat[NP]

Latitude [deg].

Definition at line 471 of file libtrac.h.

4.1.2.6 double atm_t::q[NQ][NP]

Quantitiy data (for various, user-defined attributes).

Definition at line 474 of file libtrac.h.

4.1.2.7 double atm_t::up[NP]

Zonal wind perturbation [m/s].

Definition at line 477 of file libtrac.h.

4.1.2.8 double atm_t::vp[NP]

Meridional wind perturbation [m/s].

Definition at line 480 of file libtrac.h.

4.1.2.9 double atm_t::wp[NP]

Vertical velocity perturbation [hPa/s].

Definition at line 483 of file libtrac.h.

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

- libtrac.h

4.2 ctl_t Struct Reference

Control parameters.

```
#include <libtrac.h>
```

Data Fields

- int nq
  Number of quantities.
- char qnt_name [NQ][LEN]
  Quantity names.
- char qnt_unit [NQ][LEN]
  Quantity units.
- char qnt_format [NQ][LEN]
  Quantity output format.
- int qnt_ens
  Quantity array index for ensemble IDs.
- int qnt_m
  Quantity array index for mass.
- int qnt_rho
  Quantity array index for particle density.
- int qnt_r
  Quantity array index for particle radius.
- int qnt_ps
  Quantity array index for surface pressure.
- int qnt_p
  Quantity array index for pressure.

- int qnt_t
  Quantity array index for temperature.
- int qnt_u
  Quantity array index for zonal wind.
- int qnt_v
  Quantity array index for meridional wind.
- int qnt_w
  Quantity array index for vertical velocity.
- int qnt_h2o
  Quantity array index for water vapor vmr.
- int qnt_o3
  Quantity array index for ozone vmr.
- int qnt_theta
  Quantity array index for potential temperature.
- int qnt_pv
  Quantity array index for potential vorticity.
- int qnt_tice
  Quantity array index for T_ice.
- int qnt_tsts
  Quantity array index for T_STS.
- int qnt_tnat
  Quantity array index for T_NAT.
- int qnt_stat
  Quantity array index for station flag.
- int qnt_gw_wind
  Quantity array index for low-level wind speed.
- int qnt_gw_sso
  Quantity array index for subgrid-scale orography.
- int qnt_gw_var
  Quantity array index for gravity wave variances.
- int direction
  Direction flag (1=forward calculation, -1=backward calculation).
- double t_start
  Start time of simulation [s].
- double t_stop
  Stop time of simulation [s].
- double dt_mod
  Time step of simulation [s].
- double dt_met
  Time step of meteorological data [s].
- int met_np
  Number of target pressure levels.
- double met_p [EP]
  Target pressure levels [hPa].
- int isosurf
  Isosurface parameter (0=none, 1=pressure, 2=density, 3=theta, 4=balloon).
- char balloon [LEN]
  Balloon position filename.
- double turb_dx_trop
  Horizontal turbulent diffusion coefficient (troposphere) [m^2/s].
- double turb_dx_strat

Horizontal turbulent diffusion coefficient (stratosphere) [m^2/s].

- double turb_dz_trop

  Vertical turbulent diffusion coefficient (troposphere) [m^2/s].
- double turb_dz_strat

  Vertical turbulent diffusion coefficient (stratosphere) [m^2/s].
- double turb_meso

  Scaling factor for mesoscale wind fluctuations.
- double tdec_trop

  Life time of particles (troposphere) [s].
- double tdec_strat

  Life time of particles (stratosphere) [s].
- double psc_h2o

  H2O volume mixing ratio for PSC analysis.
- double psc_hno3

  HNO3 volume mixing ratio for PSC analysis.
- char gw_basename [LEN]

  Basename for gravity wave variance data.
- char atm_basename [LEN]

  Basename of atmospheric data files.
- char atm_gpfile [LEN]

  Gnuplot file for atmospheric data.
- double atm_dt_out

  Time step for atmospheric data output [s].
- int atm_filter

  Time filter for atmospheric data output (0=no, 1=yes).
- char csi_basename [LEN]

  Basename of CSI data files.
- double csi_dt_out

  Time step for CSI data output [s].
- char csi_obsfile [LEN]

  Observation data file for CSI analysis.
- double csi_obsmin

  Minimum observation index to trigger detection.
- double csi_modmin

  Minimum column density to trigger detection [kg/m^2].
- int csi_nz

  Number of altitudes of gridded CSI data.
- double csi_z0

  Lower altitude of gridded CSI data [km].
- double csi_z1

  Upper altitude of gridded CSI data [km].
- int csi_nx

  Number of longitudes of gridded CSI data.
- double csi_lon0

  Lower longitude of gridded CSI data [deg].
- double csi_lon1

  Upper longitude of gridded CSI data [deg].
- int csi_ny

  Number of latitudes of gridded CSI data.
- double csi_lat0

  Lower latitude of gridded CSI data [deg].

- double csi_lat1

  Upper latitude of gridded CSI data [deg].
- char grid_basename [LEN]

  Basename of grid data files.
- char grid_gpfile [LEN]

  Gnuplot file for gridded data.
- double grid_dt_out

  Time step for gridded data output [s].
- int grid_sparse

  Sparse output in grid data files (0=no, 1=yes).
- int grid_nz

  Number of altitudes of gridded data.
- double grid_z0

  Lower altitude of gridded data [km].
- double grid_z1

  Upper altitude of gridded data [km].
- int grid_nx

  Number of longitudes of gridded data.
- double grid_lon0

  Lower longitude of gridded data [deg].
- double grid_lon1

  Upper longitude of gridded data [deg].
- int grid_ny

  Number of latitudes of gridded data.
- double grid_lat0

  Lower latitude of gridded data [deg].
- double grid_lat1

  Upper latitude of gridded data [deg].
- char prof_basename [LEN]

  Basename for profile output file.
- char prof_obsfile [LEN]

  Observation data file for profile output.
- int prof_nz

  Number of altitudes of gridded profile data.
- double prof_z0

  Lower altitude of gridded profile data [km].
- double prof_z1

  Upper altitude of gridded profile data [km].
- int prof_nx

  Number of longitudes of gridded profile data.
- double prof_lon0

  Lower longitude of gridded profile data [deg].
- double prof_lon1

  Upper longitude of gridded profile data [deg].
- int prof_ny

  Number of latitudes of gridded profile data.
- double prof_lat0

  Lower latitude of gridded profile data [deg].
- double prof_lat1

  Upper latitude of gridded profile data [deg].
- char ens_basename [LEN]

Basename of ensemble data file.

- char stat_basename [LEN]

  Basename of station data file.

- double stat_lon

  Longitude of station [deg].

- double stat_lat

  Latitude of station [deg].

- double stat_r

  Search radius around station [km].

4.2.1 Detailed Description

Control parameters.

Definition at line 177 of file libtrac.h.

4.2.2 Field Documentation

4.2.2.1 int ctl_t::nq

Number of quantities.

Definition at line 180 of file libtrac.h.

4.2.2.2 char ctl_t::qnt_name[NQ][LEN]

Quantity names.

Definition at line 183 of file libtrac.h.

4.2.2.3 char ctl_t::qnt_unit[NQ][LEN]

Quantity units.

Definition at line 186 of file libtrac.h.

4.2.2.4 char ctl_t::qnt_format[NQ][LEN]

Quantity output format.

Definition at line 189 of file libtrac.h.

4.2.2.5 int ctl_t::qnt_ens

Quantity array index for ensemble IDs.

Definition at line 192 of file libtrac.h.

4.2.2.6 int ctl_t::qnt_m

Quantity array index for mass.

Definition at line 195 of file libtrac.h.

4.2.2.7 int ctl_t::qnt_rho

Quantity array index for particle density.

Definition at line 198 of file libtrac.h.

4.2.2.8 int ctl_t::qnt_r

Quantity array index for particle radius.

Definition at line 201 of file libtrac.h.

4.2.2.9 int ctl_t::qnt_ps

Quantity array index for surface pressure.

Definition at line 204 of file libtrac.h.

4.2.2.10 int ctl_t::qnt_p

Quantity array index for pressure.

Definition at line 207 of file libtrac.h.

4.2.2.11 int ctl_t::qnt_t

Quantity array index for temperature.

Definition at line 210 of file libtrac.h.

4.2.2.12 int ctl_t::qnt_u

Quantity array index for zonal wind.

Definition at line 213 of file libtrac.h.

4.2.2.13 int ctl_t::qnt_v

Quantity array index for meridional wind.

Definition at line 216 of file libtrac.h.

4.2.2.14 int ctl_t::qnt_w

Quantity array index for vertical velocity.

Definition at line 219 of file libtrac.h.

4.2.2.15 int ctl_t::qnt_h2o

Quantity array index for water vapor vmr.

Definition at line 222 of file libtrac.h.

4.2.2.16 int ctl_t::qnt_o3

Quantity array index for ozone vmr.

Definition at line 225 of file libtrac.h.

4.2.2.17 int ctl_t::qnt_theta

Quantity array index for potential temperature.

Definition at line 228 of file libtrac.h.

4.2.2.18 int ctl_t::qnt_pv

Quantity array index for potential vorticity.

Definition at line 231 of file libtrac.h.

4.2.2.19 int ctl_t::qnt_tice

Quantity array index for T_ice.

Definition at line 234 of file libtrac.h.

4.2.2.20 int ctl_t::qnt_tsts

Quantity array index for T_STS.

Definition at line 237 of file libtrac.h.

4.2.2.21 int ctl_t::qnt_tnat

Quantity array index for T_NAT.

Definition at line 240 of file libtrac.h.

4.2.2.22 int ctl_t::qnt_stat

Quantity array index for station flag.

Definition at line 243 of file libtrac.h.

4.2.2.23 int ctl_t::qnt_gw_wind

Quantity array index for low-level wind speed.

Definition at line 246 of file libtrac.h.

4.2.2.24 int ctl_t::qnt_gw_sso

Quantity array index for subgrid-scale orography.

Definition at line 249 of file libtrac.h.

4.2.2.25 int ctl_t::qnt_gw_var

Quantity array index for gravity wave variances.

Definition at line 252 of file libtrac.h.

4.2.2.26 int ctl_t::direction

Direction flag (1=forward calculation, -1=backward calculation).

Definition at line 255 of file libtrac.h.

4.2.2.27 double ctl_t::t_start

Start time of simulation [s].

Definition at line 258 of file libtrac.h.

4.2.2.28 double ctl_t::t_stop

Stop time of simulation [s].

Definition at line 261 of file libtrac.h.

4.2.2.29 double ctl_t::dt_mod

Time step of simulation [s].

Definition at line 264 of file libtrac.h.

4.2.2.30 double ctl_t::dt_met

Time step of meteorological data [s].

Definition at line 267 of file libtrac.h.

4.2.2.31 int ctl_t::met_np

Number of target pressure levels.

Definition at line 270 of file libtrac.h.

4.2.2.32 double ctl_t::met_p[EP]

Target pressure levels [hPa].

Definition at line 273 of file libtrac.h.

4.2.2.33 int ctl_t::isosurf

Isosurface parameter (0=none, 1=pressure, 2=density, 3=theta, 4=balloon).

Definition at line 277 of file libtrac.h.

4.2.2.34 char ctl_t::balloon[LEN]

Balloon position filename.

Definition at line 280 of file libtrac.h.

4.2.2.35 double ctl_t::turb_dx_trop

Horizontal turbulent diffusion coefficient (troposphere) [m^2/s].

Definition at line 283 of file libtrac.h.

4.2.2.36 double ctl_t::turb_dx_strat

Horizontal turbulent diffusion coefficient (stratosphere) [m^2/s].

Definition at line 286 of file libtrac.h.

4.2.2.37 double ctl_t::turb_dz_trop

Vertical turbulent diffusion coefficient (troposphere) [m^2/s].

Definition at line 289 of file libtrac.h.

4.2.2.38 double ctl_t::turb_dz_strat

Vertical turbulent diffusion coefficient (stratosphere) [m^2/s].

Definition at line 292 of file libtrac.h.

4.2.2.39 double ctl_t::turb_meso

Scaling factor for mesoscale wind fluctuations.

Definition at line 295 of file libtrac.h.

4.2.2.40 double ctl_t::tdec_trop

Life time of particles (troposphere) [s].

Definition at line 298 of file libtrac.h.

4.2.2.41 double ctl_t::tdec_strat

Life time of particles (stratosphere) [s].

Definition at line 301 of file libtrac.h.

4.2.2.42 double ctl_t::psc_h2o

H2O volume mixing ratio for PSC analysis.

Definition at line 304 of file libtrac.h.

4.2.2.43 double ctl_t::psc_hno3

HNO3 volume mixing ratio for PSC analysis.

Definition at line 307 of file libtrac.h.

4.2.2.44 char ctl_t::gw_basename[LEN]

Basename for gravity wave variance data.

Definition at line 310 of file libtrac.h.

4.2.2.45 char ctl_t::atm_basename[LEN]

Basename of atmospheric data files.

Definition at line 313 of file libtrac.h.

4.2.2.46 char ctl_t::atm_gpfile[LEN]

Gnuplot file for atmospheric data.

Definition at line 316 of file libtrac.h.

4.2.2.47 double ctl_t::atm_dt_out

Time step for atmospheric data output [s].

Definition at line 319 of file libtrac.h.

4.2.2.48 int ctl_t::atm_filter

Time filter for atmospheric data output (0=no, 1=yes).

Definition at line 322 of file libtrac.h.

4.2.2.49 char ctl_t::csi_basename[LEN]

Basename of CSI data files.

Definition at line 325 of file libtrac.h.

4.2.2.50 double ctl_t::csi_dt_out

Time step for CSI data output [s].

Definition at line 328 of file libtrac.h.

4.2.2.51 char ctl_t::csi_obsfile[LEN]

Observation data file for CSI analysis.

Definition at line 331 of file libtrac.h.

4.2.2.52 double ctl_t::csi_obsmin

Minimum observation index to trigger detection.

Definition at line 334 of file libtrac.h.

4.2.2.53 double ctl_t::csi_modmin

Minimum column density to trigger detection [kg/m^2].

Definition at line 337 of file libtrac.h.

4.2.2.54 int ctl_t::csi_nz

Number of altitudes of gridded CSI data.

Definition at line 340 of file libtrac.h.

4.2.2.55 double ctl_t::csi_z0

Lower altitude of gridded CSI data [km].

Definition at line 343 of file libtrac.h.

4.2.2.56 double ctl_t::csi_z1

Upper altitude of gridded CSI data [km].

Definition at line 346 of file libtrac.h.

4.2.2.57 int ctl_t::csi_nx

Number of longitudes of gridded CSI data.

Definition at line 349 of file libtrac.h.

4.2.2.58 double ctl_t::csi_lon0

Lower longitude of gridded CSI data [deg].

Definition at line 352 of file libtrac.h.

4.2.2.59 double ctl_t::csi_lon1

Upper longitude of gridded CSI data [deg].

Definition at line 355 of file libtrac.h.

4.2.2.60 int ctl_t::csi_ny

Number of latitudes of gridded CSI data.

Definition at line 358 of file libtrac.h.

4.2.2.61 double ctl_t::csi_lat0

Lower latitude of gridded CSI data [deg].

Definition at line 361 of file libtrac.h.

4.2.2.62 double ctl_t::csi_lat1

Upper latitude of gridded CSI data [deg].

Definition at line 364 of file libtrac.h.

4.2.2.63 char ctl_t::grid_basename[LEN]

Basename of grid data files.

Definition at line 367 of file libtrac.h.

4.2.2.64 char ctl_t::grid_gpfile[LEN]

Gnuplot file for gridded data.

Definition at line 370 of file libtrac.h.

4.2.2.65 double ctl_t::grid_dt_out

Time step for gridded data output [s].

Definition at line 373 of file libtrac.h.

4.2.2.66 int ctl_t::grid_sparse

Sparse output in grid data files (0=no, 1=yes).

Definition at line 376 of file libtrac.h.

4.2.2.67 int ctl_t::grid_nz

Number of altitudes of gridded data.

Definition at line 379 of file libtrac.h.

4.2.2.68 double ctl_t::grid_z0

Lower altitude of gridded data [km].

Definition at line 382 of file libtrac.h.

4.2.2.69 double ctl_t::grid_z1

Upper altitude of gridded data [km].

Definition at line 385 of file libtrac.h.

4.2.2.70 int ctl_t::grid_nx

Number of longitudes of gridded data.

Definition at line 388 of file libtrac.h.

4.2.2.71 double ctl_t::grid_lon0

Lower longitude of gridded data [deg].

Definition at line 391 of file libtrac.h.

4.2.2.72 double ctl_t::grid_lon1

Upper longitude of gridded data [deg].

Definition at line 394 of file libtrac.h.

4.2.2.73 int ctl_t::grid_ny

Number of latitudes of gridded data.

Definition at line 397 of file libtrac.h.

4.2.2.74 double ctl_t::grid_lat0

Lower latitude of gridded data [deg].

Definition at line 400 of file libtrac.h.

4.2.2.75 double ctl_t::grid_lat1

Upper latitude of gridded data [deg].

Definition at line 403 of file libtrac.h.

4.2.2.76 char ctl_t::prof_basename[LEN]

Basename for profile output file.

Definition at line 406 of file libtrac.h.

4.2.2.77 char ctl_t::prof_obsfile[LEN]

Observation data file for profile output.

Definition at line 409 of file libtrac.h.

4.2.2.78 int ctl_t::prof_nz

Number of altitudes of gridded profile data.

Definition at line 412 of file libtrac.h.

4.2.2.79 double ctl_t::prof_z0

Lower altitude of gridded profile data [km].

Definition at line 415 of file libtrac.h.

4.2.2.80 double ctl_t::prof_z1

Upper altitude of gridded profile data [km].

Definition at line 418 of file libtrac.h.

4.2.2.81 int ctl_t::prof_nx

Number of longitudes of gridded profile data.

Definition at line 421 of file libtrac.h.

4.2.2.82 double ctl_t::prof_lon0

Lower longitude of gridded profile data [deg].

Definition at line 424 of file libtrac.h.

4.2.2.83 double ctl_t::prof_lon1

Upper longitude of gridded profile data [deg].

Definition at line 427 of file libtrac.h.

4.2.2.84 int ctl_t::prof_ny

Number of latitudes of gridded profile data.

Definition at line 430 of file libtrac.h.

4.2.2.85 double ctl_t::prof_lat0

Lower latitude of gridded profile data [deg].

Definition at line 433 of file libtrac.h.

4.2.2.86 double ctl_t::prof_lat1

Upper latitude of gridded profile data [deg].

Definition at line 436 of file libtrac.h.

4.2.2.87 char ctl_t::ens_basename[LEN]

Basename of ensemble data file.

Definition at line 439 of file libtrac.h.

4.2.2.88 char ctl_t::stat_basename[LEN]

Basename of station data file.

Definition at line 442 of file libtrac.h.

4.2.2.89 double ctl_t::stat_lon

Longitude of station [deg].

Definition at line 445 of file libtrac.h.

4.2.2.90 double ctl_t::stat_lat

Latitude of station [deg].

Definition at line 448 of file libtrac.h.

4.2.2.91 double ctl_t::stat_r

Search radius around station [km].

Definition at line 451 of file libtrac.h.

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

- libtrac.h

4.3 met_t Struct Reference

Meteorological data.

`#include <libtrac.h>`

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].
- float pl [EX][EY][EP]
  Pressure on model levels [hPa].
- 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 h2o [EX][EY][EP]
  Water vapor volume mixing ratio [1].
- float o3 [EX][EY][EP]
  Ozone volume mixing ratio [1].

4.3.1 Detailed Description

Meteorological data.

Definition at line 488 of file libtrac.h.

4.3.2 Field Documentation

4.3.2.1 double met_t::time

Time [s].

Definition at line 491 of file libtrac.h.

4.3.2.2 int met_t::nx

Number of longitudes.

Definition at line 494 of file libtrac.h.

4.3.2.3 int met_t::ny

Number of latitudes.

Definition at line 497 of file libtrac.h.

4.3.2.4 int met_t::np

Number of pressure levels.

Definition at line 500 of file libtrac.h.

4.3.2.5 double met_t::lon[EX]

Longitude [deg].

Definition at line 503 of file libtrac.h.

4.3.2.6 double met_t::lat[EY]

Latitude [deg].

Definition at line 506 of file libtrac.h.

4.3.2.7 double met_t::p[EP]

Pressure [hPa].

Definition at line 509 of file libtrac.h.

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

Surface pressure [hPa].

Definition at line 512 of file libtrac.h.

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

Pressure on model levels [hPa].

Definition at line 515 of file libtrac.h.

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

Temperature [K].

Definition at line 518 of file libtrac.h.

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

Zonal wind [m/s].

Definition at line 521 of file libtrac.h.

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

Meridional wind [m/s].

Definition at line 524 of file libtrac.h.

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

Vertical wind [hPa/s].

Definition at line 527 of file libtrac.h.

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

Water vapor volume mixing ratio [1].

Definition at line 530 of file libtrac.h.

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

Ozone volume mixing ratio [1].

Definition at line 533 of file libtrac.h.

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

- libtrac.h

5 File Documentation

5.1 center.c File Reference

Calculate center of mass of air parcels.

Functions

- int main (int argc, char ∗argv[ ])

5.1.1 Detailed Description

Calculate center of mass of air parcels.

Definition in file center.c.

5.1.2 Function Documentation

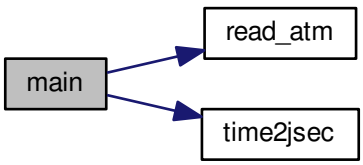
5.1.2.1 int main (int *argc*, char * *argv[]*)

Definition at line 28 of file center.c.

```
00030                {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm;
00035
00036   FILE *out;
00037
00038   char *name, *year, *mon, *day, *hour, *min;
00039
00040   double latm, lats, lonm, lons, t, zm, zs;
00041
00042   int i, f, ip;
00043
00044   /* Allocate... */
00045   ALLOC(atm, atm_t, 1);
00046
00047   /* Check arguments... */
00048   if (argc < 3)
00049     ERRMSG("Give parameters: <outfile> <atm1> [<atm2> ...]");
00050
00051   /* Write info... */
00052   printf("Write center of mass data: %s\n", argv[1]);
00053
00054   /* Create output file... */
00055   if (!(out = fopen(argv[1], "w")))
00056     ERRMSG("Cannot create file!");
00057
00058   /* Write header... */
00059   fprintf(out,
00060           "# $1  = time [s]\n"
00061           "# $2  = altitude (mean) [km]\n"
00062           "# $3  = altitude (sigma) [km]\n"
00063           "# $4  = altitude (minimum) [km]\n"
00064           "# $5  = altitude (10%% percentile) [km]\n"
00065           "# $6  = altitude (1st quarter) [km]\n"
00066           "# $7  = altitude (median) [km]\n"
00067           "# $8  = altitude (3rd quarter) [km]\n"
00068           "# $9  = altitude (90%% percentile) [km]\n"
00069           "# $10 = altitude (maximum) [km]\n");
00070   fprintf(out,
00071           "# $11 = longitude (mean) [deg]\n"
00072           "# $12 = longitude (sigma) [deg]\n"
00073           "# $13 = longitude (minimum) [deg]\n"
00074           "# $14 = longitude (10%% percentile) [deg]\n"
00075           "# $15 = longitude (1st quarter) [deg]\n"
00076           "# $16 = longitude (median) [deg]\n"
00077           "# $17 = longitude (3rd quarter) [deg]\n"
00078           "# $18 = longitude (90%% percentile) [deg]\n"
00079           "# $19 = longitude (maximum) [deg]\n");
00080   fprintf(out,
00081           "# $20 = latitude (mean) [deg]\n"
00082           "# $21 = latitude (sigma) [deg]\n"
00083           "# $22 = latitude (minimum) [deg]\n"
00084           "# $23 = latitude (10%% percentile) [deg]\n"
00085           "# $24 = latitude (1st quarter) [deg]\n"
00086           "# $25 = latitude (median) [deg]\n"
00087           "# $26 = latitude (3rd quarter) [deg]\n"
00088           "# $27 = latitude (90%% percentile) [deg]\n"
00089           "# $28 = latitude (maximum) [deg]\n\n");
00090
00091   /* Loop over files... */
00092   for (f = 2; f < argc; f++) {
00093
00094     /* Read atmopheric data... */
00095     read_atm(argv[f], &ctl, atm);
00096
00097     /* Initialize... */
00098     zm = zs = 0;
00099     lonm = lons = 0;
00100     latm = lats = 0;
00101
00102     /* Calculate mean and standard deviation... */
00103     for (ip = 0; ip < atm->np; ip++) {
00104       zm += Z(atm->p[ip]) / atm->np;
00105       lonm += atm->lon[ip] / atm->np;
```

```
00106       latm += atm->lat[ip] / atm->np;
00107       zs += gsl_pow_2(Z(atm->p[ip])) / atm->np;
00108       lons += gsl_pow_2(atm->lon[ip]) / atm->np;
00109       lats += gsl_pow_2(atm->lat[ip]) / atm->np;
00110     }
00111
00112     /* Normalize... */
00113     zs = sqrt(zs - gsl_pow_2(zm));
00114     lons = sqrt(lons - gsl_pow_2(lonm));
00115     lats = sqrt(lats - gsl_pow_2(latm));
00116
00117     /* Sort arrays... */
00118     gsl_sort(atm->p, 1, (size_t) atm->np);
00119     gsl_sort(atm->lon, 1, (size_t) atm->np);
00120     gsl_sort(atm->lat, 1, (size_t) atm->np);
00121
00122     /* Get date from filename... */
00123     for (i = (int) strlen(argv[f]) - 1; argv[f][i] != '/' || i == 0; i--);
00124     name = strtok(&(argv[f][i]), "_");
00125     year = strtok(NULL, "_");
00126     mon = strtok(NULL, "_");
00127     day = strtok(NULL, "_");
00128     hour = strtok(NULL, "_");
00129     name = strtok(NULL, "_");    /* TODO: Why another "name" here? */
00130     min = strtok(name, ".");
00131     time2jsec(atoi(year), atoi(mon), atoi(day), atoi(hour), atoi(min), 0, 0,
00132               &t);
00133
00134     /* Write data... */
00135     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g "
00136             "%g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g\n",
00137             t, zm, zs, Z(atm->p[atm->np - 1]),
00138             Z(atm->p[atm->np - atm->np / 10]),
00139             Z(atm->p[atm->np - atm->np / 4]),
00140             Z(atm->p[atm->np / 2]), Z(atm->p[atm->np / 4]),
00141             Z(atm->p[atm->np / 10]), Z(atm->p[0]),
00142             lonm, lons, atm->lon[0], atm->lon[atm->np / 10],
00143             atm->lon[atm->np / 4], atm->lon[atm->np / 2],
00144             atm->lon[atm->np - atm->np / 4],
00145             atm->lon[atm->np - atm->np / 10],
00146             atm->lon[atm->np - 1],
00147             latm, lats, atm->lat[0], atm->lat[atm->np / 10],
00148             atm->lat[atm->np / 4], atm->lat[atm->np / 2],
00149             atm->lat[atm->np - atm->np / 4],
00150             atm->lat[atm->np - atm->np / 10], atm->lat[atm->np - 1]);
00151   }
00152
00153   /* Close file... */
00154   fclose(out);
00155
00156   /* Free... */
00157   free(atm);
00158
00159   return EXIT_SUCCESS;
00160 }
```

Here is the call graph for this function:



5.2 center.c

```
00001 /*
```

```
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026 #include <gsl/gsl_sort.h>
00027
00028 int main(
00029   int argc,
00030   char *argv[]) {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm;
00035
00036   FILE *out;
00037
00038   char *name, *year, *mon, *day, *hour, *min;
00039
00040   double latm, lats, lonm, lons, t, zm, zs;
00041
00042   int i, f, ip;
00043
00044   /* Allocate... */
00045   ALLOC(atm, atm_t, 1);
00046
00047   /* Check arguments... */
00048   if (argc < 3)
00049     ERRMSG("Give parameters: <outfile> <atm1> [<atm2> ...]");
00050
00051   /* Write info... */
00052   printf("Write center of mass data: %s\n", argv[1]);
00053
00054   /* Create output file... */
00055   if (!(out = fopen(argv[1], "w")))
00056     ERRMSG("Cannot create file!");
00057
00058   /* Write header... */
00059   fprintf(out,
00060           "# $1  = time [s]\n"
00061           "# $2  = altitude (mean) [km]\n"
00062           "# $3  = altitude (sigma) [km]\n"
00063           "# $4  = altitude (minimum) [km]\n"
00064           "# $5  = altitude (10%% percentile) [km]\n"
00065           "# $6  = altitude (1st quarter) [km]\n"
00066           "# $7  = altitude (median) [km]\n"
00067           "# $8  = altitude (3rd quarter) [km]\n"
00068           "# $9  = altitude (90%% percentile) [km]\n"
00069           "# $10 = altitude (maximum) [km]\n");
00070   fprintf(out,
00071           "# $11 = longitude (mean) [deg]\n"
00072           "# $12 = longitude (sigma) [deg]\n"
00073           "# $13 = longitude (minimum) [deg]\n"
00074           "# $14 = longitude (10%% percentile) [deg]\n"
00075           "# $15 = longitude (1st quarter) [deg]\n"
00076           "# $16 = longitude (median) [deg]\n"
00077           "# $17 = longitude (3rd quarter) [deg]\n"
00078           "# $18 = longitude (90%% percentile) [deg]\n"
00079           "# $19 = longitude (maximum) [deg]\n");
00080   fprintf(out,
00081           "# $20 = latitude (mean) [deg]\n"
00082           "# $21 = latitude (sigma) [deg]\n"
00083           "# $22 = latitude (minimum) [deg]\n"
00084           "# $23 = latitude (10%% percentile) [deg]\n"
00085           "# $24 = latitude (1st quarter) [deg]\n"
00086           "# $25 = latitude (median) [deg]\n"
00087           "# $26 = latitude (3rd quarter) [deg]\n"
00088           "# $27 = latitude (90%% percentile) [deg]\n"
00089           "# $28 = latitude (maximum) [deg]\n\n");
00090
00091   /* Loop over files... */
00092   for (f = 2; f < argc; f++) {
00093
```

```
00094     /* Read atmopheric data... */
00095     read_atm(argv[f], &ctl, atm);
00096
00097     /* Initialize... */
00098     zm = zs = 0;
00099     lonm = lons = 0;
00100     latm = lats = 0;
00101
00102     /* Calculate mean and standard deviation... */
00103     for (ip = 0; ip < atm->np; ip++) {
00104       zm += Z(atm->p[ip]) / atm->np;
00105       lonm += atm->lon[ip] / atm->np;
00106       latm += atm->lat[ip] / atm->np;
00107       zs += gsl_pow_2(Z(atm->p[ip])) / atm->np;
00108       lons += gsl_pow_2(atm->lon[ip]) / atm->np;
00109       lats += gsl_pow_2(atm->lat[ip]) / atm->np;
00110     }
00111
00112     /* Normalize... */
00113     zs = sqrt(zs - gsl_pow_2(zm));
00114     lons = sqrt(lons - gsl_pow_2(lonm));
00115     lats = sqrt(lats - gsl_pow_2(latm));
00116
00117     /* Sort arrays... */
00118     gsl_sort(atm->p, 1, (size_t) atm->np);
00119     gsl_sort(atm->lon, 1, (size_t) atm->np);
00120     gsl_sort(atm->lat, 1, (size_t) atm->np);
00121
00122     /* Get date from filename... */
00123     for (i = (int) strlen(argv[f]) - 1; argv[f][i] != '/' || i == 0; i--);
00124     name = strtok(&(argv[f][i]), "_");
00125     year = strtok(NULL, "_");
00126     mon = strtok(NULL, "_");
00127     day = strtok(NULL, "_");
00128     hour = strtok(NULL, "_");
00129     name = strtok(NULL, "_");   /* TODO: Why another "name" here? */
00130     min = strtok(name, ".");
00131     time2jsec(atoi(year), atoi(mon), atoi(day), atoi(hour), atoi(min), 0, 0,
00132               &t);
00133
00134     /* Write data... */
00135     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g "
00136             "%g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g %g\n",
00137             t, zm, zs, Z(atm->p[atm->np - 1]),
00138             Z(atm->p[atm->np - atm->np / 10]),
00139             Z(atm->p[atm->np - atm->np / 4]),
00140             Z(atm->p[atm->np / 2]), Z(atm->p[atm->np / 4]),
00141             Z(atm->p[atm->np / 10]), Z(atm->p[0]),
00142             lonm, lons, atm->lon[0], atm->lon[atm->np / 10],
00143             atm->lon[atm->np / 4], atm->lon[atm->np / 2],
00144             atm->lon[atm->np - atm->np / 4],
00145             atm->lon[atm->np - atm->np / 10],
00146             atm->lon[atm->np - 1],
00147             latm, lats, atm->lat[0], atm->lat[atm->np / 10],
00148             atm->lat[atm->np / 4], atm->lat[atm->np / 2],
00149             atm->lat[atm->np - atm->np / 4],
00150             atm->lat[atm->np - atm->np / 10], atm->lat[atm->np - 1]);
00151   }
00152
00153   /* Close file... */
00154   fclose(out);
00155
00156   /* Free... */
00157   free(atm);
00158
00159   return EXIT_SUCCESS;
00160 }
```

5.3 dist.c File Reference

Calculate transport deviations of trajectories.

Functions

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

5.3.1 Detailed Description

Calculate transport deviations of trajectories.

Definition in file dist.c.

5.3.2 Function Documentation

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

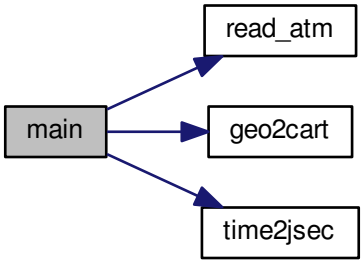
Definition at line 28 of file dist.c.

```
00030                {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm1, *atm2;
00035
00036   FILE *out;
00037
00038   char *name, *year, *mon, *day, *hour, *min;
00039
00040   double aux, x0[3], x1[3], x2[3], *lon1, *lat1, *p1, *lh1, *lv1,
00041     *lon2, *lat2, *p2, *lh2, *lv2, ahtd, avtd, ahtd2, avtd2,
00042     rhtd, rvtd, rhtd2, rvtd2, t, *dh, *dv;
00043
00044   int f, i, ip, iph, ipv;
00045
00046   /* Allocate... */
00047   ALLOC(atm1, atm_t, 1);
00048   ALLOC(atm2, atm_t, 1);
00049   ALLOC(lon1, double,
00050         NP);
00051   ALLOC(lat1, double,
00052         NP);
00053   ALLOC(p1, double,
00054         NP);
00055   ALLOC(lh1, double,
00056         NP);
00057   ALLOC(lv1, double,
00058         NP);
00059   ALLOC(lon2, double,
00060         NP);
00061   ALLOC(lat2, double,
00062         NP);
00063   ALLOC(p2, double,
00064         NP);
00065   ALLOC(lh2, double,
00066         NP);
00067   ALLOC(lv2, double,
00068         NP);
00069   ALLOC(dh, double,
00070         NP);
00071   ALLOC(dv, double,
00072         NP);
00073
00074   /* Check arguments... */
00075   if (argc < 4)
00076     ERRMSG
00077       ("Give parameters: <outfile> <atm1a> <atm1b> [<atm2a> <atm2b> ...]");
00078
00079   /* Write info... */
00080   printf("Write transport deviations: %s\n", argv[1]);
00081
00082   /* Create output file... */
00083   if (!(out = fopen(argv[1], "w")))
00084     ERRMSG("Cannot create file!");
00085
00086   /* Write header... */
00087   fprintf(out,
00088           "# $1  = time [s]\n"
00089           "# $2  = AHTD (mean) [km]\n"
00090           "# $3  = AHTD (sigma) [km]\n"
00091           "# $4  = AHTD (minimum) [km]\n"
00092           "# $5  = AHTD (10%% percentile) [km]\n"
00093           "# $6  = AHTD (1st quartile) [km]\n"
00094           "# $7  = AHTD (median) [km]\n"
```

```
00095           "# $8  = AHTD (3rd quartile) [km]\n"
00096           "# $9  = AHTD (90%% percentile) [km]\n"
00097           "# $10 = AHTD (maximum) [km]\n"
00098           "# $11 = AHTD (maximum trajectory index)\n"
00099           "# $12 = RHTD (mean) [%%]\n" "# $13 = RHTD (sigma) [%%]\n");
00100   fprintf(out,
00101           "# $14 = AVTD (mean) [km]\n"
00102           "# $15 = AVTD (sigma) [km]\n"
00103           "# $16 = AVTD (minimum) [km]\n"
00104           "# $17 = AVTD (10%% percentile) [km]\n"
00105           "# $18 = AVTD (1st quartile) [km]\n"
00106           "# $19 = AVTD (median) [km]\n"
00107           "# $20 = AVTD (3rd quartile) [km]\n"
00108           "# $21 = AVTD (90%% percentile) [km]\n"
00109           "# $22 = AVTD (maximum) [km]\n"
00110           "# $23 = AVTD (maximum trajectory index)\n"
00111           "# $24 = RVTD (mean) [%%]\n" "# $25 = RVTD (sigma) [%%]\n\n");
00112
00113   /* Loop over file pairs... */
00114   for (f = 2; f < argc; f += 2) {
00115
00116     /* Read atmopheric data... */
00117     read_atm(argv[f], &ctl, atm1);
00118     read_atm(argv[f + 1], &ctl, atm2);
00119
00120     /* Check if structs match... */
00121     if (atm1->np != atm2->np)
00122       ERRMSG("Different numbers of parcels!");
00123     for (ip = 0; ip < atm1->np; ip++)
00124       if (atm1->time[ip] != atm2->time[ip])
00125         ERRMSG("Times do not match!");
00126
00127     /* Init... */
00128     ahtd = ahtd2 = 0;
00129     avtd = avtd2 = 0;
00130     rhtd = rhtd2 = 0;
00131     rvtd = rvtd2 = 0;
00132
00133     /* Loop over air parcels... */
00134     for (ip = 0; ip < atm1->np; ip++) {
00135
00136       /* Get Cartesian coordinates... */
00137       geo2cart(0, atm1->lon[ip], atm1->lat[ip], x1);
00138       geo2cart(0, atm2->lon[ip], atm2->lat[ip], x2);
00139
00140       /* Calculate absolute transport deviations... */
00141       dh[ip] = DIST(x1, x2);
00142       ahtd += dh[ip];
00143       ahtd2 += gsl_pow_2(dh[ip]);
00144
00145       dv[ip] = fabs(Z(atm1->p[ip]) - Z(atm2->p[ip]));
00146       avtd += dv[ip];
00147       avtd2 += gsl_pow_2(dv[ip]);
00148
00149       /* Calculate relative transport deviations... */
00150       if (f > 2) {
00151
00152         /* Get trajectory lengths... */
00153         geo2cart(0, lon1[ip], lat1[ip], x0);
00154         lh1[ip] += DIST(x0, x1);
00155         lv1[ip] += fabs(Z(p1[ip]) - Z(atm1->p[ip]));
00156
00157         geo2cart(0, lon2[ip], lat2[ip], x0);
00158         lh2[ip] += DIST(x0, x2);
00159         lv2[ip] += fabs(Z(p2[ip]) - Z(atm2->p[ip]));
00160
00161         /* Get relative transport devations... */
00162         if (lh1[ip] + lh2[ip] > 0) {
00163           aux = 200. * DIST(x1, x2) / (lh1[ip] + lh2[ip]);
00164           rhtd += aux;
00165           rhtd2 += gsl_pow_2(aux);
00166         }
00167         if (lv1[ip] + lv2[ip] > 0) {
00168           aux =
00169             200. * fabs(Z(atm1->p[ip]) - Z(atm2->p[ip])) / (lv1[ip] +
00170                                                             lv2[ip]);
00171           rvtd += aux;
00172           rvtd2 += gsl_pow_2(aux);
00173         }
00174       }
00175
00176       /* Save positions of air parcels... */
00177       lon1[ip] = atm1->lon[ip];
00178       lat1[ip] = atm1->lat[ip];
00179       p1[ip] = atm1->p[ip];
00180
00181       lon2[ip] = atm2->lon[ip];
```

```
00182       lat2[ip] = atm2->lat[ip];
00183       p2[ip] = atm2->p[ip];
00184     }
00185
00186     /* Get indices of trajectories with maximum errors... */
00187     iph = (int) gsl_stats_max_index(dh, 1, (size_t) atm1->np);
00188     ipv = (int) gsl_stats_max_index(dv, 1, (size_t) atm1->np);
00189
00190     /* Sort distances to calculate percentiles... */
00191     gsl_sort(dh, 1, (size_t) atm1->np);
00192     gsl_sort(dv, 1, (size_t) atm1->np);
00193
00194     /* Get date from filename... */
00195     for (i = (int) strlen(argv[f]) - 1; argv[f][i] != '/' || i == 0; i--);
00196     name = strtok(&(argv[f][i]), "_");
00197     year = strtok(NULL, "_");
00198     mon = strtok(NULL, "_");
00199     day = strtok(NULL, "_");
00200     hour = strtok(NULL, "_");
00201     name = strtok(NULL, "_");   /* TODO: Why another "name" here? */
00202     min = strtok(name, ".");
00203     time2jsec(atoi(year), atoi(mon), atoi(day), atoi(hour), atoi(min), 0, 0,
00204               &t);
00205
00206     /* Write output... */
00207     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %d %g %g"
00208             " %g %g %g %g %g %g %g %g %d %g %g\n", t,
00209             ahtd / atm1->np,
00210             sqrt(ahtd2 / atm1->np - gsl_pow_2(ahtd / atm1->np)),
00211             dh[0], dh[atm1->np / 10], dh[atm1->np / 4], dh[atm1->np / 2],
00212             dh[atm1->np - atm1->np / 4], dh[atm1->np - atm1->np / 10],
00213             dh[atm1->np - 1], iph, rhtd / atm1->np,
00214             sqrt(rhtd2 / atm1->np - gsl_pow_2(rhtd / atm1->np)),
00215             avtd / atm1->np,
00216             sqrt(avtd2 / atm1->np - gsl_pow_2(avtd / atm1->np)),
00217             dv[0], dv[atm1->np / 10], dv[atm1->np / 4], dv[atm1->np / 2],
00218             dv[atm1->np - atm1->np / 4], dv[atm1->np - atm1->np / 10],
00219             dv[atm1->np - 1], ipv, rvtd / atm1->np,
00220             sqrt(rvtd2 / atm1->np - gsl_pow_2(rvtd / atm1->np)));
00221   }
00222
00223   /* Close file... */
00224   fclose(out);
00225
00226   /* Free... */
00227   free(atm1);
00228   free(atm2);
00229   free(lon1);
00230   free(lat1);
00231   free(p1);
00232   free(lh1);
00233   free(lv1);
00234   free(lon2);
00235   free(lat2);
00236   free(p2);
00237   free(lh2);
00238   free(lv2);
00239   free(dh);
00240   free(dv);
00241
00242   return EXIT_SUCCESS;
00243 }
```

Here is the call graph for this function:



5.4 dist.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026 #include <gsl/gsl_sort.h>
00027
00028 int main(
00029   int argc,
00030   char *argv[]) {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm1, *atm2;
00035
00036   FILE *out;
00037
00038   char *name, *year, *mon, *day, *hour, *min;
00039
00040   double aux, x0[3], x1[3], x2[3], *lon1, *lat1, *p1, *lh1, *lv1,
00041     *lon2, *lat2, *p2, *lh2, *lv2, ahtd, avtd, ahtd2, avtd2,
00042     rhtd, rvtd, rhtd2, rvtd2, t, *dh, *dv;
00043
00044   int f, i, ip, iph, ipv;
00045
00046   /* Allocate... */
00047   ALLOC(atm1, atm_t, 1);
00048   ALLOC(atm2, atm_t, 1);
00049   ALLOC(lon1, double,
00050         NP);
00051   ALLOC(lat1, double,
00052         NP);
00053   ALLOC(p1, double,
00054         NP);
00055   ALLOC(lh1, double,
00056         NP);
00057   ALLOC(lv1, double,
00058         NP);
00059   ALLOC(lon2, double,
```

```
00060         NP);
00061   ALLOC(lat2, double,
00062         NP);
00063   ALLOC(p2, double,
00064         NP);
00065   ALLOC(lh2, double,
00066         NP);
00067   ALLOC(lv2, double,
00068         NP);
00069   ALLOC(dh, double,
00070         NP);
00071   ALLOC(dv, double,
00072         NP);
00073
00074   /* Check arguments... */
00075   if (argc < 4)
00076     ERRMSG
00077       ("Give parameters: <outfile> <atm1a> <atm1b> [<atm2a> <atm2b> ...]");
00078
00079   /* Write info... */
00080   printf("Write transport deviations: %s\n", argv[1]);
00081
00082   /* Create output file... */
00083   if (!(out = fopen(argv[1], "w")))
00084     ERRMSG("Cannot create file!");
00085
00086   /* Write header... */
00087   fprintf(out,
00088           "# $1  = time [s]\n"
00089           "# $2  = AHTD (mean) [km]\n"
00090           "# $3  = AHTD (sigma) [km]\n"
00091           "# $4  = AHTD (minimum) [km]\n"
00092           "# $5  = AHTD (10%% percentile) [km]\n"
00093           "# $6  = AHTD (1st quartile) [km]\n"
00094           "# $7  = AHTD (median) [km]\n"
00095           "# $8  = AHTD (3rd quartile) [km]\n"
00096           "# $9  = AHTD (90%% percentile) [km]\n"
00097           "# $10 = AHTD (maximum) [km]\n"
00098           "# $11 = AHTD (maximum trajectory index)\n"
00099           "# $12 = RHTD (mean) [%%]\n" "# $13 = RHTD (sigma) [%%]\n");
00100   fprintf(out,
00101           "# $14 = AVTD (mean) [km]\n"
00102           "# $15 = AVTD (sigma) [km]\n"
00103           "# $16 = AVTD (minimum) [km]\n"
00104           "# $17 = AVTD (10%% percentile) [km]\n"
00105           "# $18 = AVTD (1st quartile) [km]\n"
00106           "# $19 = AVTD (median) [km]\n"
00107           "# $20 = AVTD (3rd quartile) [km]\n"
00108           "# $21 = AVTD (90%% percentile) [km]\n"
00109           "# $22 = AVTD (maximum) [km]\n"
00110           "# $23 = AVTD (maximum trajectory index)\n"
00111           "# $24 = RVTD (mean) [%%]\n" "# $25 = RVTD (sigma) [%%]\n\n");
00112
00113   /* Loop over file pairs... */
00114   for (f = 2; f < argc; f += 2) {
00115
00116     /* Read atmopheric data... */
00117     read_atm(argv[f], &ctl, atm1);
00118     read_atm(argv[f + 1], &ctl, atm2);
00119
00120     /* Check if structs match... */
00121     if (atm1->np != atm2->np)
00122       ERRMSG("Different numbers of parcels!");
00123     for (ip = 0; ip < atm1->np; ip++)
00124       if (atm1->time[ip] != atm2->time[ip])
00125         ERRMSG("Times do not match!");
00126
00127     /* Init... */
00128     ahtd = ahtd2 = 0;
00129     avtd = avtd2 = 0;
00130     rhtd = rhtd2 = 0;
00131     rvtd = rvtd2 = 0;
00132
00133     /* Loop over air parcels... */
00134     for (ip = 0; ip < atm1->np; ip++) {
00135
00136       /* Get Cartesian coordinates... */
00137       geo2cart(0, atm1->lon[ip], atm1->lat[ip], x1);
00138       geo2cart(0, atm2->lon[ip], atm2->lat[ip], x2);
00139
00140       /* Calculate absolute transport deviations... */
00141       dh[ip] = DIST(x1, x2);
00142       ahtd += dh[ip];
00143       ahtd2 += gsl_pow_2(dh[ip]);
00144
00145       dv[ip] = fabs(Z(atm1->p[ip]) - Z(atm2->p[ip]));
00146       avtd += dv[ip];
```

```
00147       avtd2 += gsl_pow_2(dv[ip]);
00148
00149       /* Calculate relative transport deviations... */
00150       if (f > 2) {
00151
00152         /* Get trajectory lengths... */
00153         geo2cart(0, lon1[ip], lat1[ip], x0);
00154         lh1[ip] += DIST(x0, x1);
00155         lv1[ip] += fabs(Z(p1[ip]) - Z(atm1->p[ip]));
00156
00157         geo2cart(0, lon2[ip], lat2[ip], x0);
00158         lh2[ip] += DIST(x0, x2);
00159         lv2[ip] += fabs(Z(p2[ip]) - Z(atm2->p[ip]));
00160
00161         /* Get relative transport devations... */
00162         if (lh1[ip] + lh2[ip] > 0) {
00163           aux = 200. * DIST(x1, x2) / (lh1[ip] + lh2[ip]);
00164           rhtd += aux;
00165           rhtd2 += gsl_pow_2(aux);
00166         }
00167         if (lv1[ip] + lv2[ip] > 0) {
00168           aux =
00169             200. * fabs(Z(atm1->p[ip]) - Z(atm2->p[ip])) / (lv1[ip] +
00170                                                             lv2[ip]);
00171           rvtd += aux;
00172           rvtd2 += gsl_pow_2(aux);
00173         }
00174       }
00175
00176       /* Save positions of air parcels... */
00177       lon1[ip] = atm1->lon[ip];
00178       lat1[ip] = atm1->lat[ip];
00179       p1[ip] = atm1->p[ip];
00180
00181       lon2[ip] = atm2->lon[ip];
00182       lat2[ip] = atm2->lat[ip];
00183       p2[ip] = atm2->p[ip];
00184     }
00185
00186     /* Get indices of trajectories with maximum errors... */
00187     iph = (int) gsl_stats_max_index(dh, 1, (size_t) atm1->np);
00188     ipv = (int) gsl_stats_max_index(dv, 1, (size_t) atm1->np);
00189
00190     /* Sort distances to calculate percentiles... */
00191     gsl_sort(dh, 1, (size_t) atm1->np);
00192     gsl_sort(dv, 1, (size_t) atm1->np);
00193
00194     /* Get date from filename... */
00195     for (i = (int) strlen(argv[f]) - 1; argv[f][i] != '/' || i == 0; i--);
00196     name = strtok(&(argv[f][i]), "_");
00197     year = strtok(NULL, "_");
00198     mon = strtok(NULL, "_");
00199     day = strtok(NULL, "_");
00200     hour = strtok(NULL, "_");
00201     name = strtok(NULL, "_");   /* TODO: Why another "name" here? */
00202     min = strtok(name, ".");
00203     time2jsec(atoi(year), atoi(mon), atoi(day), atoi(hour), atoi(min), 0, 0,
00204               &t);
00205
00206     /* Write output... */
00207     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %d %g %g"
00208             " %g %g %g %g %g %g %g %g %g %d %g %g\n", t,
00209             ahtd / atm1->np,
00210             sqrt(ahtd2 / atm1->np - gsl_pow_2(ahtd / atm1->np)),
00211             dh[0], dh[atm1->np / 10], dh[atm1->np / 4], dh[atm1->np / 2],
00212             dh[atm1->np - atm1->np / 4], dh[atm1->np - atm1->np / 10],
00213             dh[atm1->np - 1], iph, rhtd / atm1->np,
00214             sqrt(rhtd2 / atm1->np - gsl_pow_2(rhtd / atm1->np)),
00215             avtd / atm1->np,
00216             sqrt(avtd2 / atm1->np - gsl_pow_2(avtd / atm1->np)),
00217             dv[0], dv[atm1->np / 10], dv[atm1->np / 4], dv[atm1->np / 2],
00218             dv[atm1->np - atm1->np / 4], dv[atm1->np - atm1->np / 10],
00219             dv[atm1->np - 1], ipv, rvtd / atm1->np,
00220             sqrt(rvtd2 / atm1->np - gsl_pow_2(rvtd / atm1->np)));
00221   }
00222
00223   /* Close file... */
00224   fclose(out);
00225
00226   /* Free... */
00227   free(atm1);
00228   free(atm2);
00229   free(lon1);
00230   free(lat1);
00231   free(p1);
00232   free(lh1);
00233   free(lv1);
```

```
00234   free(lon2);
00235   free(lat2);
00236   free(p2);
00237   free(lh2);
00238   free(lv2);
00239   free(dh);
00240   free(dv);
00241
00242   return EXIT_SUCCESS;
00243 }
```

5.5 extract.c File Reference

Extract single trajectory from atmospheric data files.

Functions

- int main (int argc, char ∗argv[ ])

5.5.1 Detailed Description

Extract single trajectory from atmospheric data files.

Definition in file extract.c.

5.5.2 Function Documentation

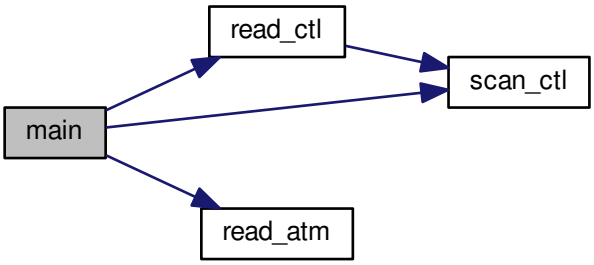
5.5.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Definition at line 28 of file extract.c.

```
00030                {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm;
00035
00036   FILE *in, *out;
00037
00038   int f, ip, iq;
00039
00040   /* Allocate... */
00041   ALLOC(atm, atm_t, 1);
00042
00043   /* Check arguments... */
00044   if (argc < 4)
00045     ERRMSG("Give parameters: <ctl> <outfile> <atm1> [<atm2> ...]");
00046
00047   /* Read control parameters... */
00048   read_ctl(argv[1], argc, argv, &ctl);
00049   ip = (int) scan_ctl(argv[1], argc, argv, "EXTRACT_IP", -1, "0", NULL);
00050
00051   /* Write info... */
00052   printf("Write trajectory data: %s\n", argv[2]);
00053
00054   /* Create output file... */
00055   if (!(out = fopen(argv[2], "w")))
00056     ERRMSG("Cannot create file!");
00057
00058   /* Write header... */
00059   fprintf(out,
00060           "# $1 = time [s]\n"
00061           "# $2 = altitude [km]\n"
00062           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
00063   for (iq = 0; iq < ctl.nq; iq++)
```

```
00064     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl.qnt_name[iq],
00065             ctl.qnt_unit[iq]);
00066   fprintf(out, "\n");
00067
00068   /* Loop over files... */
00069   for (f = 3; f < argc; f++) {
00070
00071     /* Read atmopheric data... */
00072     if (!(in = fopen(argv[f], "r")))
00073       continue;
00074     else
00075       fclose(in);
00076     read_atm(argv[f], &ctl, atm);
00077
00078     /* Write data... */
00079     fprintf(out, "%.2f %g %g %g", atm->time[ip],
00080             Z(atm->p[ip]), atm->lon[ip], atm->lat[ip]);
00081     for (iq = 0; iq < ctl.nq; iq++) {
00082       fprintf(out, " ");
00083       fprintf(out, ctl.qnt_format[iq], atm->q[iq][ip]);
00084     }
00085     fprintf(out, "\n");
00086   }
00087
00088   /* Close file... */
00089   fclose(out);
00090
00091   /* Free... */
00092   free(atm);
00093
00094   return EXIT_SUCCESS;
00095 }
```

Here is the call graph for this function:



5.6 extract.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
```

```
00025 #include "libtrac.h"
00026 #include <gsl/gsl_sort.h>
00027
00028 int main(
00029   int argc,
00030   char *argv[]) {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm;
00035
00036   FILE *in, *out;
00037
00038   int f, ip, iq;
00039
00040   /* Allocate... */
00041   ALLOC(atm, atm_t, 1);
00042
00043   /* Check arguments... */
00044   if (argc < 4)
00045     ERRMSG("Give parameters: <ctl> <outfile> <atm1> [<atm2> ...]");
00046
00047   /* Read control parameters... */
00048   read_ctl(argv[1], argc, argv, &ctl);
00049   ip = (int) scan_ctl(argv[1], argc, argv, "EXTRACT_IP", -1, "0", NULL);
00050
00051   /* Write info... */
00052   printf("Write trajectory data: %s\n", argv[2]);
00053
00054   /* Create output file... */
00055   if (!(out = fopen(argv[2], "w")))
00056     ERRMSG("Cannot create file!");
00057
00058   /* Write header... */
00059   fprintf(out,
00060           "# $1 = time [s]\n"
00061           "# $2 = altitude [km]\n"
00062           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
00063   for (iq = 0; iq < ctl.nq; iq++)
00064     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl.qnt_name[iq],
00065             ctl.qnt_unit[iq]);
00066   fprintf(out, "\n");
00067
00068   /* Loop over files... */
00069   for (f = 3; f < argc; f++) {
00070
00071     /* Read atmopheric data... */
00072     if (!(in = fopen(argv[f], "r")))
00073       continue;
00074     else
00075       fclose(in);
00076     read_atm(argv[f], &ctl, atm);
00077
00078     /* Write data... */
00079     fprintf(out, "%.2f %g %g %g", atm->time[ip],
00080             Z(atm->p[ip]), atm->lon[ip], atm->lat[ip]);
00081     for (iq = 0; iq < ctl.nq; iq++) {
00082       fprintf(out, " ");
00083       fprintf(out, ctl.qnt_format[iq], atm->q[iq][ip]);
00084     }
00085     fprintf(out, "\n");
00086   }
00087
00088   /* Close file... */
00089   fclose(out);
00090
00091   /* Free... */
00092   free(atm);
00093
00094   return EXIT_SUCCESS;
00095 }
```

5.7 init.c File Reference

Create atmospheric data file with initial air parcel positions.

Functions

- int main (int argc, char ∗argv[ ])

5.7.1 Detailed Description

Create atmospheric data file with initial air parcel positions.

Definition in file init.c.

5.7.2 Function Documentation

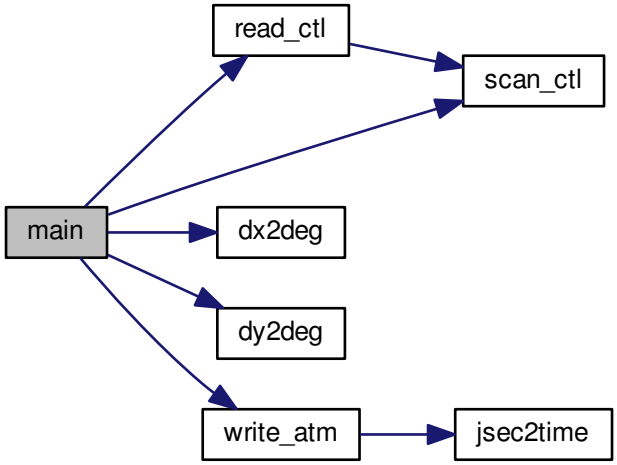
5.7.2.1 int main (int *argc*, char * *argv[ ]*)

Definition at line 27 of file init.c.

```
00029                {
00030
00031   atm_t *atm;
00032
00033   ctl_t ctl;
00034
00035   gsl_rng *rng;
00036
00037   double dt, dz, dlon, dlat, lat0, lat1, lon0, lon1, t0, t1, z0, z1,
00038     t, z, lon, lat, st, sz, slon, slat, sx, ut, uz, ulon, ulat, m;
00039
00040   int ip, irep, rep;
00041
00042   /* Allocate... */
00043   ALLOC(atm, atm_t, 1);
00044
00045   /* Check arguments... */
00046   if (argc < 3)
00047     ERRMSG("Give parameters: <ctl> <atm_out>");
00048
00049   /* Read control parameters... */
00050   read_ctl(argv[1], argc, argv, &ctl);
00051   t0 = scan_ctl(argv[1], argc, argv, "INIT_T0", -1, "0", NULL);
00052   t1 = scan_ctl(argv[1], argc, argv, "INIT_T1", -1, "0", NULL);
00053   dt = scan_ctl(argv[1], argc, argv, "INIT_DT", -1, "1", NULL);
00054   z0 = scan_ctl(argv[1], argc, argv, "INIT_Z0", -1, "0", NULL);
00055   z1 = scan_ctl(argv[1], argc, argv, "INIT_Z1", -1, "0", NULL);
00056   dz = scan_ctl(argv[1], argc, argv, "INIT_DZ", -1, "1", NULL);
00057   lon0 = scan_ctl(argv[1], argc, argv, "INIT_LON0", -1, "0", NULL);
00058   lon1 = scan_ctl(argv[1], argc, argv, "INIT_LON1", -1, "0", NULL);
00059   dlon = scan_ctl(argv[1], argc, argv, "INIT_DLON", -1, "1", NULL);
00060   lat0 = scan_ctl(argv[1], argc, argv, "INIT_LAT0", -1, "0", NULL);
00061   lat1 = scan_ctl(argv[1], argc, argv, "INIT_LAT1", -1, "0", NULL);
00062   dlat = scan_ctl(argv[1], argc, argv, "INIT_DLAT", -1, "1", NULL);
00063   st = scan_ctl(argv[1], argc, argv, "INIT_ST", -1, "0", NULL);
00064   sz = scan_ctl(argv[1], argc, argv, "INIT_SZ", -1, "0", NULL);
00065   slon = scan_ctl(argv[1], argc, argv, "INIT_SLON", -1, "0", NULL);
00066   slat = scan_ctl(argv[1], argc, argv, "INIT_SLAT", -1, "0", NULL);
00067   sx = scan_ctl(argv[1], argc, argv, "INIT_SX", -1, "0", NULL);
00068   ut = scan_ctl(argv[1], argc, argv, "INIT_UT", -1, "0", NULL);
00069   uz = scan_ctl(argv[1], argc, argv, "INIT_UZ", -1, "0", NULL);
00070   ulon = scan_ctl(argv[1], argc, argv, "INIT_ULON", -1, "0", NULL);
00071   ulat = scan_ctl(argv[1], argc, argv, "INIT_ULAT", -1, "0", NULL);
00072   rep = (int) scan_ctl(argv[1], argc, argv, "INIT_REP", -1, "1", NULL);
00073   m = scan_ctl(argv[1], argc, argv, "INIT_MASS", -1, "0", NULL);
00074
00075   /* Initialize random number generator... */
00076   gsl_rng_env_setup();
00077   rng = gsl_rng_alloc(gsl_rng_default);
00078
00079   /* Create grid... */
00080   for (t = t0; t <= t1; t += dt)
00081     for (z = z0; z <= z1; z += dz)
00082       for (lon = lon0; lon <= lon1; lon += dlon)
00083         for (lat = lat0; lat <= lat1; lat += dlat)
00084           for (irep = 0; irep < rep; irep++) {
00085
00086             /* Set position... */
00087             atm->time[atm->np]
00088               = (t + gsl_ran_gaussian_ziggurat(rng, st / 2.3548)
00089                  + ut * (gsl_rng_uniform(rng) - 0.5));
00090             atm->p[atm->np]
00091               = P(z + gsl_ran_gaussian_ziggurat(rng, sz / 2.3548)
00092                   + uz * (gsl_rng_uniform(rng) - 0.5));
00093             atm->lon[atm->np]
```

```
00094               = (lon + gsl_ran_gaussian_ziggurat(rng, slon / 2.3548)
00095                  + gsl_ran_gaussian_ziggurat(rng, dx2deg(sx, lat) / 2.3548)
00096                  + ulon * (gsl_rng_uniform(rng) - 0.5));
00097             atm->lat[atm->np]
00098               = (lat + gsl_ran_gaussian_ziggurat(rng, slat / 2.3548)
00099                  + gsl_ran_gaussian_ziggurat(rng, dy2deg(sx) / 2.3548)
00100                  + ulat * (gsl_rng_uniform(rng) - 0.5));
00101
00102             /* Set particle counter... */
00103             if ((++atm->np) >= NP)
00104               ERRMSG("Too many particles!");
00105           }
00106
00107   /* Check number of air parcels... */
00108   if (atm->np <= 0)
00109     ERRMSG("Did not create any air parcels!");
00110
00111   /* Initialize mass... */
00112   if (ctl.qnt_m >= 0)
00113     for (ip = 0; ip < atm->np; ip++)
00114       atm->q[ctl.qnt_m][ip] = m / atm->np;
00115
00116   /* Save data... */
00117   write_atm(argv[2], &ctl, atm, t0);
00118
00119   /* Free... */
00120   gsl_rng_free(rng);
00121   free(atm);
00122
00123   return EXIT_SUCCESS;
00124 }
```

Here is the call graph for this function:



5.8 init.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 int main(
00028   int argc,
00029   char *argv[]) {
00030
00031   atm_t *atm;
00032
00033   ctl_t ctl;
00034
00035   gsl_rng *rng;
00036
00037   double dt, dz, dlon, dlat, lat0, lat1, lon0, lon1, t0, t1, z0, z1,
00038     t, z, lon, lat, st, sz, slon, slat, sx, ut, uz, ulon, ulat, m;
00039
00040   int ip, irep, rep;
00041
00042   /* Allocate... */
00043   ALLOC(atm, atm_t, 1);
00044
00045   /* Check arguments... */
00046   if (argc < 3)
00047     ERRMSG("Give parameters: <ctl> <atm_out>");
00048
00049   /* Read control parameters... */
00050   read_ctl(argv[1], argc, argv, &ctl);
00051   t0 = scan_ctl(argv[1], argc, argv, "INIT_T0", -1, "0", NULL);
00052   t1 = scan_ctl(argv[1], argc, argv, "INIT_T1", -1, "0", NULL);
00053   dt = scan_ctl(argv[1], argc, argv, "INIT_DT", -1, "1", NULL);
00054   z0 = scan_ctl(argv[1], argc, argv, "INIT_Z0", -1, "0", NULL);
00055   z1 = scan_ctl(argv[1], argc, argv, "INIT_Z1", -1, "0", NULL);
00056   dz = scan_ctl(argv[1], argc, argv, "INIT_DZ", -1, "1", NULL);
00057   lon0 = scan_ctl(argv[1], argc, argv, "INIT_LON0", -1, "0", NULL);
00058   lon1 = scan_ctl(argv[1], argc, argv, "INIT_LON1", -1, "0", NULL);
00059   dlon = scan_ctl(argv[1], argc, argv, "INIT_DLON", -1, "1", NULL);
00060   lat0 = scan_ctl(argv[1], argc, argv, "INIT_LAT0", -1, "0", NULL);
00061   lat1 = scan_ctl(argv[1], argc, argv, "INIT_LAT1", -1, "0", NULL);
00062   dlat = scan_ctl(argv[1], argc, argv, "INIT_DLAT", -1, "1", NULL);
00063   st = scan_ctl(argv[1], argc, argv, "INIT_ST", -1, "0", NULL);
00064   sz = scan_ctl(argv[1], argc, argv, "INIT_SZ", -1, "0", NULL);
00065   slon = scan_ctl(argv[1], argc, argv, "INIT_SLON", -1, "0", NULL);
00066   slat = scan_ctl(argv[1], argc, argv, "INIT_SLAT", -1, "0", NULL);
00067   sx = scan_ctl(argv[1], argc, argv, "INIT_SX", -1, "0", NULL);
00068   ut = scan_ctl(argv[1], argc, argv, "INIT_UT", -1, "0", NULL);
00069   uz = scan_ctl(argv[1], argc, argv, "INIT_UZ", -1, "0", NULL);
00070   ulon = scan_ctl(argv[1], argc, argv, "INIT_ULON", -1, "0", NULL);
00071   ulat = scan_ctl(argv[1], argc, argv, "INIT_ULAT", -1, "0", NULL);
00072   rep = (int) scan_ctl(argv[1], argc, argv, "INIT_REP", -1, "1", NULL);
00073   m = scan_ctl(argv[1], argc, argv, "INIT_MASS", -1, "0", NULL);
00074
00075   /* Initialize random number generator... */
00076   gsl_rng_env_setup();
00077   rng = gsl_rng_alloc(gsl_rng_default);
00078
00079   /* Create grid... */
00080   for (t = t0; t <= t1; t += dt)
00081     for (z = z0; z <= z1; z += dz)
00082       for (lon = lon0; lon <= lon1; lon += dlon)
00083         for (lat = lat0; lat <= lat1; lat += dlat)
00084           for (irep = 0; irep < rep; irep++) {
00085
00086             /* Set position... */
00087             atm->time[atm->np]
00088               = (t + gsl_ran_gaussian_ziggurat(rng, st / 2.3548)
00089                  + ut * (gsl_rng_uniform(rng) - 0.5));
00090             atm->p[atm->np]
00091               = P(z + gsl_ran_gaussian_ziggurat(rng, sz / 2.3548)
00092                   + uz * (gsl_rng_uniform(rng) - 0.5));
00093             atm->lon[atm->np]
00094               = (lon + gsl_ran_gaussian_ziggurat(rng, slon / 2.3548)
00095                  + gsl_ran_gaussian_ziggurat(rng, dx2deg(sx, lat) / 2.3548)
00096                  + ulon * (gsl_rng_uniform(rng) - 0.5));
00097             atm->lat[atm->np]
00098               = (lat + gsl_ran_gaussian_ziggurat(rng, slat / 2.3548)
00099                  + gsl_ran_gaussian_ziggurat(rng, dy2deg(sx) / 2.3548)
00100                 + ulat * (gsl_rng_uniform(rng) - 0.5));
00101
00102             /* Set particle counter... */
```

```
00103             if ((++atm->np) >= NP)
00104               ERRMSG("Too many particles!");
00105           }
00106
00107   /* Check number of air parcels... */
00108   if (atm->np <= 0)
00109     ERRMSG("Did not create any air parcels!");
00110
00111   /* Initialize mass... */
00112   if (ctl.qnt_m >= 0)
00113     for (ip = 0; ip < atm->np; ip++)
00114       atm->q[ctl.qnt_m][ip] = m / atm->np;
00115
00116   /* Save data... */
00117   write_atm(argv[2], &ctl, atm, t0);
00118
00119   /* Free... */
00120   gsl_rng_free(rng);
00121   free(atm);
00122
00123   return EXIT_SUCCESS;
00124 }
```

5.9 jsec2time.c File Reference

Convert Julian seconds to date.

Functions

- int main (int argc, char ∗argv[ ])

5.9.1 Detailed Description

Convert Julian seconds to date.

Definition in file jsec2time.c.

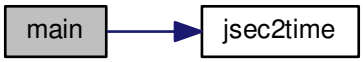
5.9.2 Function Documentation

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

Definition at line 27 of file jsec2time.c.

```
00029                {
00030
00031   double jsec, remain;
00032
00033   int day, hour, min, mon, sec, year;
00034
00035   /* Check arguments... */
00036   if (argc < 2)
00037     ERRMSG("Give parameters: <jsec>");
00038
00039   /* Read arguments... */
00040   jsec = atof(argv[1]);
00041
00042   /* Convert time... */
00043   jsec2time(jsec, &year, &mon, &day, &hour, &min, &sec, &remain);
00044   printf("%d %d %d %d %d %d %g\n", year, mon, day, hour, min, sec, remain);
00045
00046   return EXIT_SUCCESS;
00047 }
```

Here is the call graph for this function:



5.10 jsec2time.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 int main(
00028   int argc,
00029   char *argv[]) {
00030
00031   double jsec, remain;
00032
00033   int day, hour, min, mon, sec, year;
00034
00035   /* Check arguments... */
00036   if (argc < 2)
00037     ERRMSG("Give parameters: <jsec>");
00038
00039   /* Read arguments... */
00040   jsec = atof(argv[1]);
00041
00042   /* Convert time... */
00043   jsec2time(jsec, &year, &mon, &day, &hour, &min, &sec, &remain);
00044   printf("%d %d %d %d %d %d %g\n", year, mon, day, hour, min, sec, remain);
00045
00046   return EXIT_SUCCESS;
00047 }
```

5.11 libtrac.c File Reference

MPTRAC library definitions.

Functions

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

  Convert Cartesian coordinates to geolocation.
- double deg2dx (double dlon, double lat)

Convert degrees to horizontal distance.

- double deg2dy (double dlat)

  Convert degrees to horizontal distance.
- double dp2dz (double dp, double p)

  Convert pressure to vertical distance.
- double dx2deg (double dx, double lat)

  Convert horizontal distance to degrees.
- double dy2deg (double dy)

  Convert horizontal distance to degrees.
- double dz2dp (double dz, double p)

  Convert vertical distance to pressure.
- void geo2cart (double z, double lon, double lat, double *x)

  Convert geolocation to Cartesian coordinates.
- void get_met (ctl_t *ctl, 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 *t, double *u, double *v, double *w, 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 *t, double *u, double *v, double *w, double *h2o, double *o3)

  Temporal interpolation of meteorological data.
- void jsec2time (double jsec, int *year, int *mon, int *day, int *hour, int *min, int *sec, double *remain)

  Convert seconds to date.
- int locate (double *xx, int n, double x)

  Find array index.
- void read_atm (const char *filename, ctl_t *ctl, atm_t *atm)

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

  Read control parameters.
- void read_met (ctl_t *ctl, 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_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_ml2pl (ctl_t *ctl, met_t *met, float var[EX][EY][EP])

  Convert meteorological data from model levels to pressure levels.
- void read_met_periodic (met_t *met)

  Create meteorological data with periodic boundary conditions.
- double scan_ctl (const char *filename, int argc, char *argv[ ], const char *varname, int arridx, const char *defvalue, char *value)

  Read a control parameter from file or command line.
- 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, int id, int mode)

Measure wall-clock time.

- double tropopause (double t, double lat)
- void write_atm (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write atmospheric data.
- void write_csi (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write CSI data.
- void write_ens (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write ensemble data.
- void write_grid (const char *filename, ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, double t)

  Write gridded data.
- void write_prof (const char *filename, ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, double t)

  Write profile data.
- void write_station (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write station data.

5.11.1 Detailed Description

MPTRAC library definitions.

Definition in file libtrac.c.

5.11.2 Function Documentation

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

Convert Cartesian coordinates to geolocation.

Definition at line 29 of file libtrac.c.

```
00033                {
00034
00035   double radius;
00036
00037   radius = NORM(x);
00038   *lat = asin(x[2] / radius) * 180 / M_PI;
00039   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00040   *z = radius - RE;
00041 }
```

5.11.2.2 double deg2dx (double *dlon*, double *lat*)

Convert degrees to horizontal distance.

Definition at line 45 of file libtrac.c.

```
00047               {
00048
00049   return dlon * M_PI * RE / 180. * cos(lat / 180. * M_PI);
00050 }
```

5.11.2.3 double deg2dy (double *dlat*)

Convert degrees to horizontal distance.

Definition at line 54 of file libtrac.c.

```
00055                {
00056
00057   return dlat * M_PI * RE / 180.;
00058 }
```

5.11.2.4 double dp2dz (double *dp*, double *p*)

Convert pressure to vertical distance.

Definition at line 62 of file libtrac.c.

```
00064             {
00065
00066   return -dp * H0 / p;
00067 }
```

5.11.2.5 double dx2deg (double *dx*, double *lat*)

Convert horizontal distance to degrees.

Definition at line 71 of file libtrac.c.

```
00073               {
00074
00075   /* Avoid singularity at poles... */
00076   if (lat < -89.999 || lat > 89.999)
00077     return 0;
00078   else
00079     return dx * 180. / (M_PI * RE * cos(lat / 180. * M_PI));
00080 }
```

5.11.2.6 double dy2deg (double *dy*)

Convert horizontal distance to degrees.

Definition at line 84 of file libtrac.c.

```
00085              {
00086
00087   return dy * 180. / (M_PI * RE);
00088 }
```

5.11.2.7 double dz2dp (double *dz*, double *p*)

Convert vertical distance to pressure.

Definition at line 92 of file libtrac.c.

```
00094             {
00095
00096   return -dz * p / H0;
00097 }
```

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

Convert geolocation to Cartesian coordinates.

Definition at line 101 of file libtrac.c.

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

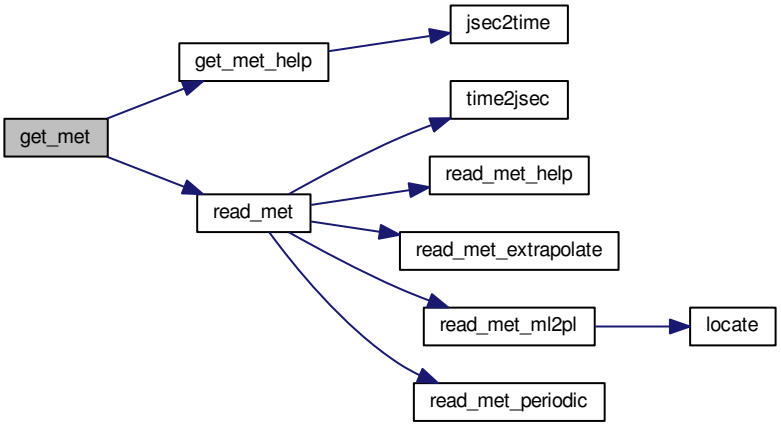
5.11.2.9 void get_met (ctl_t * *ctl*, char * *metbase*, double *t*, met_t * *met0*, met_t * *met1*)

Get meteorological data for given timestep.

Definition at line 117 of file libtrac.c.

```
00122                {
00123
00124   char filename[LEN];
00125
00126   static int init;
00127
00128   /* Init... */
00129   if (!init) {
00130     init = 1;
00131
00132     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00133     read_met(ctl, filename, met0);
00134
00135     get_met_help(t + 1.0 * ctl->direction, 1, metbase, ctl->
      dt_met, filename);
00136     read_met(ctl, filename, met1);
00137   }
00138
00139   /* Read new data for forward trajectories... */
00140   if (t > met1->time && ctl->direction == 1) {
00141     memcpy(met0, met1, sizeof(met_t));
00142     get_met_help(t, 1, metbase, ctl->dt_met, filename);
00143     read_met(ctl, filename, met1);
00144   }
00145
00146   /* Read new data for backward trajectories... */
00147   if (t < met0->time && ctl->direction == -1) {
00148     memcpy(met1, met0, sizeof(met_t));
00149     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00150     read_met(ctl, filename, met0);
00151   }
00152 }
```

Here is the call graph for this function:



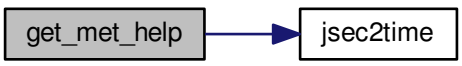
5.11.2.10 void get_met_help (double *t*, int *direct*, char * *metbase*, double *dt_met*, char * *filename*)

Get meteorological data for timestep.

Definition at line 156 of file libtrac.c.

```
00161                   {
00162
00163   double t6, r;
00164
00165   int year, mon, day, hour, min, sec;
00166
00167   /* Round time to fixed intervals... */
00168   if (direct == -1)
00169     t6 = floor(t / dt_met) * dt_met;
00170   else
00171     t6 = ceil(t / dt_met) * dt_met;
00172
00173   /* Decode time... */
00174   jsec2time(t6, &year, &mon, &day, &hour, &min, &sec, &r);
00175
00176   /* Set filename... */
00177   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", metbase, year, mon, day, hour);
00178 }
```

Here is the call graph for this function:



5.11.2.11 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 182 of file libtrac.c.

```
00188                {
00189
00190   double aux00, aux01, aux10, aux11;
00191
00192   /* Set variables... */
00193   aux00 = array[ix][iy];
00194   aux01 = array[ix][iy + 1];
00195   aux10 = array[ix + 1][iy];
00196   aux11 = array[ix + 1][iy + 1];
00197
00198   /* Interpolate horizontally... */
00199   aux00 = wy * (aux00 - aux01) + aux01;
00200   aux11 = wy * (aux10 - aux11) + aux11;
00201   *var = wx * (aux00 - aux11) + aux11;
00202 }
```

5.11.2.12 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 206 of file libtrac.c.

```
00214                {
00215
00216   double aux00, aux01, aux10, aux11;
00217
00218   /* Interpolate vertically... */
00219   aux00 = wp * (array[ix][iy][ip] - array[ix][iy][ip + 1])
00220     + array[ix][iy][ip + 1];
00221   aux01 = wp * (array[ix][iy + 1][ip] - array[ix][iy + 1][ip + 1])
00222     + array[ix][iy + 1][ip + 1];
00223   aux10 = wp * (array[ix + 1][iy][ip] - array[ix + 1][iy][ip + 1])
00224     + array[ix + 1][iy][ip + 1];
00225   aux11 = wp * (array[ix + 1][iy + 1][ip] - array[ix + 1][iy + 1][ip + 1])
00226     + array[ix + 1][iy + 1][ip + 1];
00227
00228   /* Interpolate horizontally... */
00229   aux00 = wy * (aux00 - aux01) + aux01;
00230   aux11 = wy * (aux10 - aux11) + aux11;
00231   *var = wx * (aux00 - aux11) + aux11;
00232 }
```

5.11.2.13 void intpol_met_space (met_t * *met*, double *p*, double *lon*, double *lat*, double * *ps*, double * *t*, double * *u*, double * *v*, double * *w*, double * *h2o*, double * *o3*)

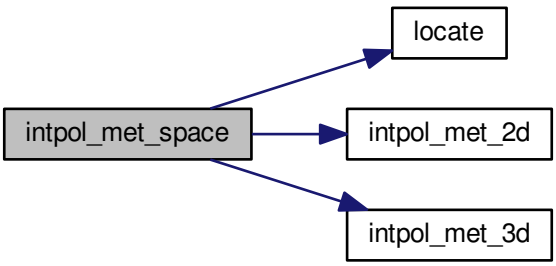
Spatial interpolation of meteorological data.

Definition at line 236 of file libtrac.c.

```
00247                {
00248
00249   double wp, wx, wy;
00250
00251   int ip, ix, iy;
00252
00253   /* Check longitude... */
00254   if (met->lon[met->nx - 1] > 180 && lon < 0)
00255     lon += 360;
00256
00257   /* Get indices... */
00258   ip = locate(met->p, met->np, p);
00259   ix = locate(met->lon, met->nx, lon);
00260   iy = locate(met->lat, met->ny, lat);
```

```
00261
00262   /* Get weights... */
00263   wp = (met->p[ip + 1] - p) / (met->p[ip + 1] - met->p[ip]);
00264   wx = (met->lon[ix + 1] - lon) / (met->lon[ix + 1] - met->lon[ix]);
00265   wy = (met->lat[iy + 1] - lat) / (met->lat[iy + 1] - met->lat[iy]);
00266
00267   /* Interpolate... */
00268   if (ps != NULL)
00269     intpol_met_2d(met->ps, ix, iy, wx, wy, ps);
00270   if (t != NULL)
00271     intpol_met_3d(met->t, ip, ix, iy, wp, wx, wy, t);
00272   if (u != NULL)
00273     intpol_met_3d(met->u, ip, ix, iy, wp, wx, wy, u);
00274   if (v != NULL)
00275     intpol_met_3d(met->v, ip, ix, iy, wp, wx, wy, v);
00276   if (w != NULL)
00277     intpol_met_3d(met->w, ip, ix, iy, wp, wx, wy, w);
00278   if (h2o != NULL)
00279     intpol_met_3d(met->h2o, ip, ix, iy, wp, wx, wy, h2o);
00280   if (o3 != NULL)
00281     intpol_met_3d(met->o3, ip, ix, iy, wp, wx, wy, o3);
00282 }
```

Here is the call graph for this function:



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

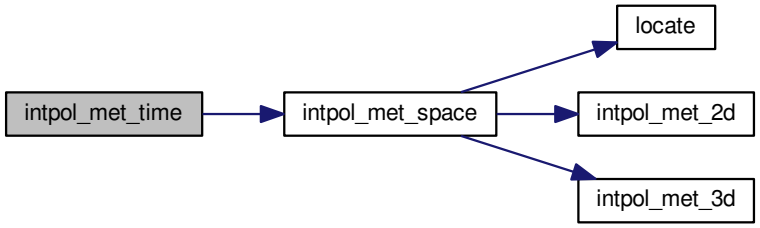
Temporal interpolation of meteorological data.

Definition at line 286 of file libtrac.c.

```
00299             {
00300
00301   double h2o0, h2o1, o30, o31, ps0, ps1, t0, t1, u0, u1, v0, v1, w0, w1, wt;
00302
00303   /* Spatial interpolation... */
00304   intpol_met_space(met0, p, lon, lat,
00305                    ps == NULL ? NULL : &ps0,
00306                    t == NULL ? NULL : &t0,
00307                    u == NULL ? NULL : &u0,
00308                    v == NULL ? NULL : &v0,
00309                    w == NULL ? NULL : &w0,
00310                    h2o == NULL ? NULL : &h2o0, o3 == NULL ? NULL : &o30);
00311   intpol_met_space(met1, p, lon, lat,
00312                    ps == NULL ? NULL : &ps1,
00313                    t == NULL ? NULL : &t1,
00314                    u == NULL ? NULL : &u1,
00315                    v == NULL ? NULL : &v1,
00316                    w == NULL ? NULL : &w1,
00317                    h2o == NULL ? NULL : &h2o1, o3 == NULL ? NULL : &o31);
00318
```

```
00319   /* Get weighting factor... */
00320   wt = (met1->time - ts) / (met1->time - met0->time);
00321
00322   /* Interpolate... */
00323   if (ps != NULL)
00324     *ps = wt * (ps0 - ps1) + ps1;
00325   if (t != NULL)
00326     *t = wt * (t0 - t1) + t1;
00327   if (u != NULL)
00328     *u = wt * (u0 - u1) + u1;
00329   if (v != NULL)
00330     *v = wt * (v0 - v1) + v1;
00331   if (w != NULL)
00332     *w = wt * (w0 - w1) + w1;
00333   if (h2o != NULL)
00334     *h2o = wt * (h2o0 - h2o1) + h2o1;
00335   if (o3 != NULL)
00336     *o3 = wt * (o30 - o31) + o31;
00337 }
```

Here is the call graph for this function:



5.11.2.15 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 341 of file libtrac.c.

```
00349                   {
00350
00351   struct tm t0, *t1;
00352
00353   time_t jsec0;
00354
00355   t0.tm_year = 100;
00356   t0.tm_mon = 0;
00357   t0.tm_mday = 1;
00358   t0.tm_hour = 0;
00359   t0.tm_min = 0;
00360   t0.tm_sec = 0;
00361
00362   jsec0 = (time_t) jsec + timegm(&t0);
00363   t1 = gmtime(&jsec0);
00364
00365   *year = t1->tm_year + 1900;
00366   *mon = t1->tm_mon + 1;
00367   *day = t1->tm_mday;
00368   *hour = t1->tm_hour;
00369   *min = t1->tm_min;
00370   *sec = t1->tm_sec;
00371   *remain = jsec - floor(jsec);
00372 }
```

5.11.2.16 int locate (double * xx, int n, double x)

Find array index.

Definition at line 376 of file libtrac.c.

```
00379            {
00380
00381   int i, ilo, ihi;
00382
00383   ilo = 0;
00384   ihi = n - 1;
00385   i = (ihi + ilo) >> 1;
00386
00387   if (xx[i] < xx[i + 1])
00388     while (ihi > ilo + 1) {
00389       i = (ihi + ilo) >> 1;
00390       if (xx[i] > x)
00391         ihi = i;
00392       else
00393         ilo = i;
00394   } else
00395     while (ihi > ilo + 1) {
00396       i = (ihi + ilo) >> 1;
00397       if (xx[i] <= x)
00398         ihi = i;
00399       else
00400         ilo = i;
00401     }
00402
00403   return ilo;
00404 }
```

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

Read atmospheric data.

Definition at line 408 of file libtrac.c.

```
00411               {
00412
00413   FILE *in;
00414
00415   char line[LEN], *tok;
00416
00417   int iq;
00418
00419   /* Init... */
00420   atm->np = 0;
00421
00422   /* Write info... */
00423   printf("Read atmospheric data: %s\n", filename);
00424
00425   /* Open file... */
00426   if (!(in = fopen(filename, "r")))
00427     ERRMSG("Cannot open file!");
00428
00429   /* Read line... */
00430   while (fgets(line, LEN, in)) {
00431
00432     /* Read data... */
00433     TOK(line, tok, "%lg", atm->time[atm->np]);
00434     TOK(NULL, tok, "%lg", atm->p[atm->np]);
00435     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
00436     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
00437     for (iq = 0; iq < ctl->nq; iq++)
00438       TOK(NULL, tok, "%lg", atm->q[iq][atm->np]);
00439
00440     /* Convert altitude to pressure... */
00441     atm->p[atm->np] = P(atm->p[atm->np]);
00442
00443     /* Increment data point counter... */
00444     if ((++atm->np) > NP)
00445       ERRMSG("Too many data points!");
00446   }
00447
00448   /* Close file... */
00449   fclose(in);
00450
00451   /* Check number of points... */
00452   if (atm->np < 1)
00453     ERRMSG("Can not read any data!");
00454 }
```

5.11.2.18 void read_ctl (const char * *filename*, int *argc*, char * *argv[]*, ctl_t * *ctl*)

Read control parameters.

Definition at line 458 of file libtrac.c.

```
00462               {
00463
00464   int ip, iq;
00465
00466   /* Write info... */
00467   printf("\nMassive-Parallel Trajectory Calculations (MPTRAC)\n"
00468          "(executable: %s | compiled: %s, %s)\n\n",
00469          argv[0], __DATE__, __TIME__);
00470
00471   /* Initialize quantity indices... */
00472   ctl->qnt_ens = -1;
00473   ctl->qnt_m = -1;
00474   ctl->qnt_r = -1;
00475   ctl->qnt_rho = -1;
00476   ctl->qnt_ps = -1;
00477   ctl->qnt_p = -1;
00478   ctl->qnt_t = -1;
00479   ctl->qnt_u = -1;
00480   ctl->qnt_v = -1;
00481   ctl->qnt_w = -1;
00482   ctl->qnt_h2o = -1;
00483   ctl->qnt_o3 = -1;
00484   ctl->qnt_theta = -1;
00485   ctl->qnt_pv = -1;
00486   ctl->qnt_tice = -1;
00487   ctl->qnt_tsts = -1;
00488   ctl->qnt_tnat = -1;
00489   ctl->qnt_gw_wind = -1;
00490   ctl->qnt_gw_sso = -1;
00491   ctl->qnt_gw_var = -1;
00492   ctl->qnt_stat = -1;
00493
00494   /* Read quantities... */
00495   ctl->nq = (int) scan_ctl(filename, argc, argv, "NQ", -1, "0", NULL);
00496   if (ctl->nq > NQ)
00497     ERRMSG("Too many quantities!");
00498   for (iq = 0; iq < ctl->nq; iq++) {
00499
00500     /* Read quantity name and format... */
00501     scan_ctl(filename, argc, argv, "QNT_NAME", iq, "", ctl->qnt_name[iq]);
00502     scan_ctl(filename, argc, argv, "QNT_FORMAT", iq, "%g",
00503              ctl->qnt_format[iq]);
00504
00505     /* Try to identify quantity... */
00506     if (strcmp(ctl->qnt_name[iq], "ens") == 0) {
00507       ctl->qnt_ens = iq;
00508       sprintf(ctl->qnt_unit[iq], "-");
00509     } else if (strcmp(ctl->qnt_name[iq], "m") == 0) {
00510       ctl->qnt_m = iq;
00511       sprintf(ctl->qnt_unit[iq], "kg");
00512     } else if (strcmp(ctl->qnt_name[iq], "r") == 0) {
00513       ctl->qnt_r = iq;
00514       sprintf(ctl->qnt_unit[iq], "m");
00515     } else if (strcmp(ctl->qnt_name[iq], "rho") == 0) {
00516       ctl->qnt_rho = iq;
00517       sprintf(ctl->qnt_unit[iq], "kg/m^3");
00518     } else if (strcmp(ctl->qnt_name[iq], "ps") == 0) {
00519       ctl->qnt_ps = iq;
00520       sprintf(ctl->qnt_unit[iq], "hPa");
00521     } else if (strcmp(ctl->qnt_name[iq], "p") == 0) {
00522       ctl->qnt_p = iq;
00523       sprintf(ctl->qnt_unit[iq], "hPa");
00524     } else if (strcmp(ctl->qnt_name[iq], "t") == 0) {
00525       ctl->qnt_t = iq;
00526       sprintf(ctl->qnt_unit[iq], "K");
00527     } else if (strcmp(ctl->qnt_name[iq], "u") == 0) {
00528       ctl->qnt_u = iq;
00529       sprintf(ctl->qnt_unit[iq], "m/s");
00530     } else if (strcmp(ctl->qnt_name[iq], "v") == 0) {
00531       ctl->qnt_v = iq;
00532       sprintf(ctl->qnt_unit[iq], "m/s");
00533     } else if (strcmp(ctl->qnt_name[iq], "w") == 0) {
00534       ctl->qnt_w = iq;
00535       sprintf(ctl->qnt_unit[iq], "hPa/s");
00536     } else if (strcmp(ctl->qnt_name[iq], "h2o") == 0) {
00537       ctl->qnt_h2o = iq;
00538       sprintf(ctl->qnt_unit[iq], "1");
```

```
00539     } else if (strcmp(ctl->qnt_name[iq], "o3") == 0) {
00540       ctl->qnt_o3 = iq;
00541       sprintf(ctl->qnt_unit[iq], "1");
00542     } else if (strcmp(ctl->qnt_name[iq], "theta") == 0) {
00543       ctl->qnt_theta = iq;
00544       sprintf(ctl->qnt_unit[iq], "K");
00545     } else if (strcmp(ctl->qnt_name[iq], "pv") == 0) {
00546       ctl->qnt_pv = iq;
00547       sprintf(ctl->qnt_unit[iq], "PVU");
00548     } else if (strcmp(ctl->qnt_name[iq], "tice") == 0) {
00549       ctl->qnt_tice = iq;
00550       sprintf(ctl->qnt_unit[iq], "K");
00551     } else if (strcmp(ctl->qnt_name[iq], "tsts") == 0) {
00552       ctl->qnt_tsts = iq;
00553       sprintf(ctl->qnt_unit[iq], "K");
00554     } else if (strcmp(ctl->qnt_name[iq], "tnat") == 0) {
00555       ctl->qnt_tnat = iq;
00556       sprintf(ctl->qnt_unit[iq], "K");
00557     } else if (strcmp(ctl->qnt_name[iq], "gw_wind") == 0) {
00558       ctl->qnt_gw_wind = iq;
00559       sprintf(ctl->qnt_unit[iq], "m/s");
00560     } else if (strcmp(ctl->qnt_name[iq], "gw_sso") == 0) {
00561       ctl->qnt_gw_sso = iq;
00562       sprintf(ctl->qnt_unit[iq], "m^2");
00563     } else if (strcmp(ctl->qnt_name[iq], "gw_var") == 0) {
00564       ctl->qnt_gw_var = iq;
00565       sprintf(ctl->qnt_unit[iq], "K^2");
00566     } else if (strcmp(ctl->qnt_name[iq], "stat") == 0) {
00567       ctl->qnt_stat = iq;
00568       sprintf(ctl->qnt_unit[iq], "-");
00569     } else
00570       scan_ctl(filename, argc, argv, "QNT_UNIT", iq, "", ctl->qnt_unit[iq]);
00571   }
00572
00573   /* Time steps of simulation... */
00574   ctl->direction =
00575     (int) scan_ctl(filename, argc, argv, "DIRECTION", -1, "1", NULL);
00576   if (ctl->direction != -1 && ctl->direction != 1)
00577     ERRMSG("Set DIRECTION to -1 or 1!");
00578   ctl->t_start =
00579     scan_ctl(filename, argc, argv, "T_START", -1, "-1e100", NULL);
00580   ctl->t_stop = scan_ctl(filename, argc, argv, "T_STOP", -1, "-1e100", NULL);
00581   ctl->dt_mod = scan_ctl(filename, argc, argv, "DT_MOD", -1, "600", NULL);
00582
00583   /* Meteorological data... */
00584   ctl->dt_met = scan_ctl(filename, argc, argv, "DT_MET", -1, "21600", NULL);
00585   ctl->met_np = (int) scan_ctl(filename, argc, argv, "MET_NP", -1, "0", NULL);
00586   if (ctl->met_np > EP)
00587     ERRMSG("Too many levels!");
00588   for (ip = 0; ip < ctl->met_np; ip++)
00589     ctl->met_p[ip] = scan_ctl(filename, argc, argv, "MET_P", ip, "", NULL);
00590
00591   /* Isosurface parameters... */
00592   ctl->isosurf
00593     = (int) scan_ctl(filename, argc, argv, "ISOSURF", -1, "0", NULL);
00594   scan_ctl(filename, argc, argv, "BALLOON", -1, "-", ctl->balloon);
00595
00596   /* Diffusion parameters... */
00597   ctl->turb_dx_trop
00598     = scan_ctl(filename, argc, argv, "TURB_DX_TROP", -1, "50.0", NULL);
00599   ctl->turb_dx_strat
00600     = scan_ctl(filename, argc, argv, "TURB_DX_STRAT", -1, "0.0", NULL);
00601   ctl->turb_dz_trop
00602     = scan_ctl(filename, argc, argv, "TURB_DZ_TROP", -1, "0.0", NULL);
00603   ctl->turb_dz_strat
00604     = scan_ctl(filename, argc, argv, "TURB_DZ_STRAT", -1, "0.1", NULL);
00605   ctl->turb_meso =
00606     scan_ctl(filename, argc, argv, "TURB_MESO", -1, "0.16", NULL);
00607
00608   /* Life time of particles... */
00609   ctl->tdec_trop = scan_ctl(filename, argc, argv, "TDEC_TROP", -1, "0", NULL);
00610   ctl->tdec_strat =
00611     scan_ctl(filename, argc, argv, "TDEC_STRAT", -1, "0", NULL);
00612
00613   /* PSC analysis... */
00614   ctl->psc_h2o = scan_ctl(filename, argc, argv, "PSC_H2O", -1, "4e-6", NULL);
00615   ctl->psc_hno3 =
00616     scan_ctl(filename, argc, argv, "PSC_HNO3", -1, "9e-9", NULL);
00617
00618   /* Gravity wave analysis... */
00619   scan_ctl(filename, argc, argv, "GW_BASENAME", -1, "-", ctl->
     gw_basename);
00620
00621   /* Output of atmospheric data... */
00622   scan_ctl(filename, argc, argv, "ATM_BASENAME", -1, "-", ctl->
     atm_basename);
00623   scan_ctl(filename, argc, argv, "ATM_GPFILE", -1, "-", ctl->atm_gpfile);
```

```
00624   ctl->atm_dt_out =
00625     scan_ctl(filename, argc, argv, "ATM_DT_OUT", -1, "86400", NULL);
00626   ctl->atm_filter =
00627     (int) scan_ctl(filename, argc, argv, "ATM_FILTER", -1, "0", NULL);
00628
00629   /* Output of CSI data... */
00630   scan_ctl(filename, argc, argv, "CSI_BASENAME", -1, "-", ctl->
      csi_basename);
00631   ctl->csi_dt_out =
00632     scan_ctl(filename, argc, argv, "CSI_DT_OUT", -1, "86400", NULL);
00633   scan_ctl(filename, argc, argv, "CSI_OBSFILE", -1, "obs.tab",
00634            ctl->csi_obsfile);
00635   ctl->csi_obsmin =
00636     scan_ctl(filename, argc, argv, "CSI_OBSMIN", -1, "0", NULL);
00637   ctl->csi_modmin =
00638     scan_ctl(filename, argc, argv, "CSI_MODMIN", -1, "0", NULL);
00639   ctl->csi_z0 = scan_ctl(filename, argc, argv, "CSI_Z0", -1, "0", NULL);
00640   ctl->csi_z1 = scan_ctl(filename, argc, argv, "CSI_Z1", -1, "100", NULL);
00641   ctl->csi_nz = (int) scan_ctl(filename, argc, argv, "CSI_NZ", -1, "1", NULL);
00642   ctl->csi_lon0 =
00643     scan_ctl(filename, argc, argv, "CSI_LON0", -1, "-180", NULL);
00644   ctl->csi_lon1 = scan_ctl(filename, argc, argv, "CSI_LON1", -1, "180", NULL);
00645   ctl->csi_nx =
00646     (int) scan_ctl(filename, argc, argv, "CSI_NX", -1, "360", NULL);
00647   ctl->csi_lat0 = scan_ctl(filename, argc, argv, "CSI_LAT0", -1, "-90", NULL);
00648   ctl->csi_lat1 = scan_ctl(filename, argc, argv, "CSI_LAT1", -1, "90", NULL);
00649   ctl->csi_ny =
00650     (int) scan_ctl(filename, argc, argv, "CSI_NY", -1, "180", NULL);
00651
00652   /* Output of ensemble data... */
00653   scan_ctl(filename, argc, argv, "ENS_BASENAME", -1, "-", ctl->
      ens_basename);
00654
00655   /* Output of grid data... */
00656   scan_ctl(filename, argc, argv, "GRID_BASENAME", -1, "-",
00657            ctl->grid_basename);
00658   scan_ctl(filename, argc, argv, "GRID_GPFILE", -1, "-", ctl->
      grid_gpfile);
00659   ctl->grid_dt_out =
00660     scan_ctl(filename, argc, argv, "GRID_DT_OUT", -1, "86400", NULL);
00661   ctl->grid_sparse =
00662     (int) scan_ctl(filename, argc, argv, "GRID_SPARSE", -1, "0", NULL);
00663   ctl->grid_z0 = scan_ctl(filename, argc, argv, "GRID_Z0", -1, "0", NULL);
00664   ctl->grid_z1 = scan_ctl(filename, argc, argv, "GRID_Z1", -1, "100", NULL);
00665   ctl->grid_nz =
00666     (int) scan_ctl(filename, argc, argv, "GRID_NZ", -1, "1", NULL);
00667   ctl->grid_lon0 =
00668     scan_ctl(filename, argc, argv, "GRID_LON0", -1, "-180", NULL);
00669   ctl->grid_lon1 =
00670     scan_ctl(filename, argc, argv, "GRID_LON1", -1, "180", NULL);
00671   ctl->grid_nx =
00672     (int) scan_ctl(filename, argc, argv, "GRID_NX", -1, "360", NULL);
00673   ctl->grid_lat0 =
00674     scan_ctl(filename, argc, argv, "GRID_LAT0", -1, "-90", NULL);
00675   ctl->grid_lat1 =
00676     scan_ctl(filename, argc, argv, "GRID_LAT1", -1, "90", NULL);
00677   ctl->grid_ny =
00678     (int) scan_ctl(filename, argc, argv, "GRID_NY", -1, "180", NULL);
00679
00680   /* Output of profile data... */
00681   scan_ctl(filename, argc, argv, "PROF_BASENAME", -1, "-",
00682            ctl->prof_basename);
00683   scan_ctl(filename, argc, argv, "PROF_OBSFILE", -1, "-", ctl->
      prof_obsfile);
00684   ctl->prof_z0 = scan_ctl(filename, argc, argv, "PROF_Z0", -1, "0", NULL);
00685   ctl->prof_z1 = scan_ctl(filename, argc, argv, "PROF_Z1", -1, "60", NULL);
00686   ctl->prof_nz =
00687     (int) scan_ctl(filename, argc, argv, "PROF_NZ", -1, "60", NULL);
00688   ctl->prof_lon0 =
00689     scan_ctl(filename, argc, argv, "PROF_LON0", -1, "-180", NULL);
00690   ctl->prof_lon1 =
00691     scan_ctl(filename, argc, argv, "PROF_LON1", -1, "180", NULL);
00692   ctl->prof_nx =
00693     (int) scan_ctl(filename, argc, argv, "PROF_NX", -1, "360", NULL);
00694   ctl->prof_lat0 =
00695     scan_ctl(filename, argc, argv, "PROF_LAT0", -1, "-90", NULL);
00696   ctl->prof_lat1 =
00697     scan_ctl(filename, argc, argv, "PROF_LAT1", -1, "90", NULL);
00698   ctl->prof_ny =
00699     (int) scan_ctl(filename, argc, argv, "PROF_NY", -1, "180", NULL);
00700
00701   /* Output of station data... */
00702   scan_ctl(filename, argc, argv, "STAT_BASENAME", -1, "-",
00703            ctl->stat_basename);
00704   ctl->stat_lon = scan_ctl(filename, argc, argv, "STAT_LON", -1, "0", NULL);
00705   ctl->stat_lat = scan_ctl(filename, argc, argv, "STAT_LAT", -1, "0", NULL);
00706   ctl->stat_r = scan_ctl(filename, argc, argv, "STAT_R", -1, "50", NULL);
```

```
00707 }
```

Here is the call graph for this function:

5.11.2.19 void read_met (ctl_t * *ctl*, char * *filename*, met_t * *met*)

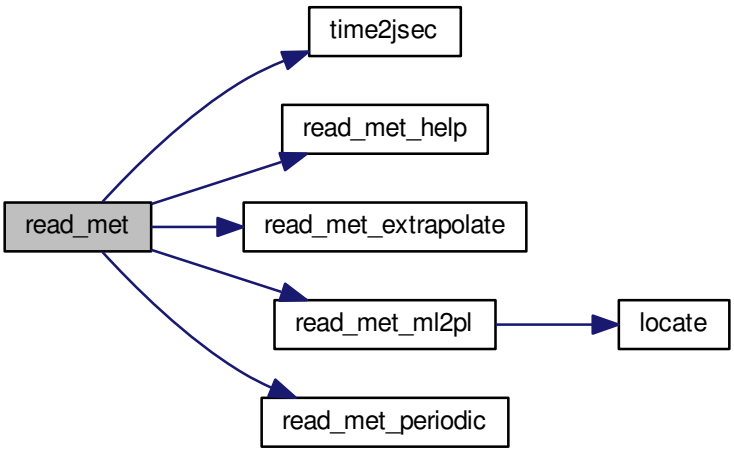
Read meteorological data file.

Definition at line 711 of file libtrac.c.

```
00714                {
00715
00716   char tstr[10];
00717
00718   static float help[EX * EY];
00719
00720   int ix, iy, ip, dimid, ncid, varid, year, mon, day, hour;
00721
00722   size_t np, nx, ny;
00723
00724   /* Write info... */
00725   printf("Read meteorological data: %s\n", filename);
00726
00727   /* Get time from filename... */
00728   sprintf(tstr, "%.4s", &filename[strlen(filename) - 16]);
00729   year = atoi(tstr);
00730   sprintf(tstr, "%.2s", &filename[strlen(filename) - 11]);
00731   mon = atoi(tstr);
00732   sprintf(tstr, "%.2s", &filename[strlen(filename) - 8]);
00733   day = atoi(tstr);
00734   sprintf(tstr, "%.2s", &filename[strlen(filename) - 5]);
00735   hour = atoi(tstr);
00736   time2jsec(year, mon, day, hour, 0, 0, 0, &met->time);
00737
00738   /* Open netCDF file... */
00739   NC(nc_open(filename, NC_NOWRITE, &ncid));
00740
00741   /* Get dimensions... */
00742   NC(nc_inq_dimid(ncid, "lon", &dimid));
00743   NC(nc_inq_dimlen(ncid, dimid, &nx));
00744   if (nx > EX)
00745     ERRMSG("Too many longitudes!");
00746
00747   NC(nc_inq_dimid(ncid, "lat", &dimid));
00748   NC(nc_inq_dimlen(ncid, dimid, &ny));
00749   if (ny > EY)
00750     ERRMSG("Too many latitudes!");
00751
00752   NC(nc_inq_dimid(ncid, "lev", &dimid));
00753   NC(nc_inq_dimlen(ncid, dimid, &np));
00754   if (np > EP)
00755     ERRMSG("Too many levels!");
00756
00757   /* Store dimensions... */
00758   met->np = (int) np;
00759   met->nx = (int) nx;
00760   met->ny = (int) ny;
00761
00762   /* Get horizontal grid... */
00763   NC(nc_inq_varid(ncid, "lon", &varid));
00764   NC(nc_get_var_double(ncid, varid, met->lon));
00765   NC(nc_inq_varid(ncid, "lat", &varid));
```

```
00766   NC(nc_get_var_double(ncid, varid, met->lat));
00767
00768   /* Read meteorological data... */
00769   read_met_help(ncid, "t", "T", met, met->t, 1.0);
00770   read_met_help(ncid, "u", "U", met, met->u, 1.0);
00771   read_met_help(ncid, "v", "V", met, met->v, 1.0);
00772   read_met_help(ncid, "w", "W", met, met->w, 0.01f);
00773   read_met_help(ncid, "q", "Q", met, met->h2o, 1.608f);
00774   read_met_help(ncid, "o3", "O3", met, met->o3, 0.602f);
00775
00776   /* Meteo data on pressure levels... */
00777   if (ctl->met_np <= 0) {
00778
00779     /* Read pressure levels from file... */
00780     NC(nc_inq_varid(ncid, "lev", &varid));
00781     NC(nc_get_var_double(ncid, varid, met->p));
00782     for (ip = 0; ip < met->np; ip++)
00783       met->p[ip] /= 100.;
00784
00785     /* Extrapolate data for lower boundary... */
00786     read_met_extrapolate(met);
00787   }
00788
00789   /* Meteo data on model levels... */
00790   else {
00791
00792     /* Read pressure data from file... */
00793     read_met_help(ncid, "pl", "PL", met, met->pl, 0.01f);
00794
00795     /* Interpolate from model levels to pressure levels... */
00796     read_met_ml2pl(ctl, met, met->t);
00797     read_met_ml2pl(ctl, met, met->u);
00798     read_met_ml2pl(ctl, met, met->v);
00799     read_met_ml2pl(ctl, met, met->w);
00800     read_met_ml2pl(ctl, met, met->h2o);
00801     read_met_ml2pl(ctl, met, met->o3);
00802
00803     /* Set pressure levels... */
00804     met->np = ctl->met_np;
00805     for (ip = 0; ip < met->np; ip++)
00806       met->p[ip] = ctl->met_p[ip];
00807   }
00808
00809   /* Check ordering of pressure levels... */
00810   for (ip = 1; ip < met->np; ip++)
00811     if (met->p[ip - 1] < met->p[ip])
00812       ERRMSG("Pressure levels must be descending!");
00813
00814   /* Read surface pressure... */
00815   if (nc_inq_varid(ncid, "PS", &varid) == NC_NOERR) {
00816     NC(nc_get_var_float(ncid, varid, help));
00817     for (iy = 0; iy < met->ny; iy++)
00818       for (ix = 0; ix < met->nx; ix++)
00819         met->ps[ix][iy] = help[iy * met->nx + ix] / 100.;
00820   } else if (nc_inq_varid(ncid, "LNSP", &varid) == NC_NOERR) {
00821     NC(nc_get_var_float(ncid, varid, help));
00822     for (iy = 0; iy < met->ny; iy++)
00823       for (ix = 0; ix < met->nx; ix++)
00824         met->ps[ix][iy] = exp(help[iy * met->nx + ix]) / 100.;
00825   } else
00826     for (ix = 0; ix < met->nx; ix++)
00827       for (iy = 0; iy < met->ny; iy++)
00828         met->ps[ix][iy] = met->p[0];
00829
00830   /* Create periodic boundary conditions... */
00831   read_met_periodic(met);
00832
00833   /* Close file... */
00834   NC(nc_close(ncid));
00835 }
```

Here is the call graph for this function:



5.11.2.20 void read_met_extrapolate (met_t * *met*)

Extrapolate meteorological data at lower boundary.

Definition at line 839 of file libtrac.c.

```
00840                {
00841
00842   int ip, ip0, ix, iy;
00843
00844   /* Loop over columns... */
00845   for (ix = 0; ix < met->nx; ix++)
00846     for (iy = 0; iy < met->ny; iy++) {
00847
00848       /* Find lowest valid data point... */
00849       for (ip0 = met->np - 1; ip0 >= 0; ip0--)
00850         if (!gsl_finite(met->t[ix][iy][ip0])
00851             || !gsl_finite(met->u[ix][iy][ip0])
00852             || !gsl_finite(met->v[ix][iy][ip0])
00853             || !gsl_finite(met->w[ix][iy][ip0]))
00854           break;
00855
00856       /* Extrapolate... */
00857       for (ip = ip0; ip >= 0; ip--) {
00858         met->t[ix][iy][ip] = met->t[ix][iy][ip + 1];
00859         met->u[ix][iy][ip] = met->u[ix][iy][ip + 1];
00860         met->v[ix][iy][ip] = met->v[ix][iy][ip + 1];
00861         met->w[ix][iy][ip] = met->w[ix][iy][ip + 1];
00862         met->h2o[ix][iy][ip] = met->h2o[ix][iy][ip + 1];
00863         met->o3[ix][iy][ip] = met->o3[ix][iy][ip + 1];
00864       }
00865     }
00866 }
```

5.11.2.21 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 870 of file libtrac.c.

```
00876             {
00877
00878   static float help[EX * EY * EP];
00879
00880   int ip, ix, iy, n = 0, varid;
00881
00882   /* Check if variable exists... */
00883   if (nc_inq_varid(ncid, varname, &varid) != NC_NOERR)
00884     if (nc_inq_varid(ncid, varname2, &varid) != NC_NOERR)
00885       return;
00886
00887   /* Read data... */
00888   NC(nc_get_var_float(ncid, varid, help));
00889
00890   /* Copy and check data... */
00891   for (ip = 0; ip < met->np; ip++)
00892     for (iy = 0; iy < met->ny; iy++)
00893       for (ix = 0; ix < met->nx; ix++) {
00894         dest[ix][iy][ip] = scl * help[n++];
00895         if (fabs(dest[ix][iy][ip] / scl) > 1e14)
00896           dest[ix][iy][ip] = GSL_NAN;
00897       }
00898 }
```

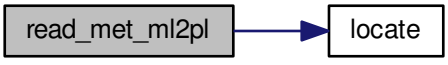
5.11.2.22 void read_met_ml2pl (ctl_t * *ctl*, met_t * *met*, float *var[EX][EY][EP]*)

Convert meteorological data from model levels to pressure levels.

Definition at line 902 of file libtrac.c.

```
00905                         {
00906
00907   double aux[EP], p[EP], pt;
00908
00909   int ip, ip2, ix, iy;
00910
00911   /* Loop over columns... */
00912   for (ix = 0; ix < met->nx; ix++)
00913     for (iy = 0; iy < met->ny; iy++) {
00914
00915       /* Copy pressure profile... */
00916       for (ip = 0; ip < met->np; ip++)
00917         p[ip] = met->pl[ix][iy][ip];
00918
00919       /* Interpolate... */
00920       for (ip = 0; ip < ctl->met_np; ip++) {
00921         pt = ctl->met_p[ip];
00922         if ((pt > p[0] && p[0] > p[1]) || (pt < p[0] && p[0] < p[1]))
00923           pt = p[0];
00924         else if ((pt > p[met->np - 1] && p[1] > p[0])
00925                  || (pt < p[met->np - 1] && p[1] < p[0]))
00926           pt = p[met->np - 1];
00927         ip2 = locate(p, met->np, pt);
00928         aux[ip] = LIN(p[ip2], var[ix][iy][ip2],
00929                       p[ip2 + 1], var[ix][iy][ip2 + 1], pt);
00930       }
00931
00932       /* Copy data... */
00933       for (ip = 0; ip < ctl->met_np; ip++)
00934         var[ix][iy][ip] = (float) aux[ip];
00935     }
00936 }
```

Here is the call graph for this function:



5.11.2.23 void read_met_periodic (met_t * *met*)

Create meteorological data with periodic boundary conditions.

Definition at line 940 of file libtrac.c.

```
00941                {
00942
00943   int ip, iy;
00944
00945   /* Check longitudes... */
00946   if (!(fabs(met->lon[met->nx - 1] - met->lon[0]
00947              + met->lon[1] - met->lon[0] - 360) < 0.01))
00948     return;
00949
00950   /* Increase longitude counter... */
00951   if ((++met->nx) > EX)
00952     ERRMSG("Cannot create periodic boundary conditions!");
00953
00954   /* Set longitude... */
00955   met->lon[met->nx - 1] = met->lon[met->nx - 2] + met->lon[1] - met->
      lon[0];
00956
00957   /* Loop over latitudes and pressure levels... */
00958   for (iy = 0; iy < met->ny; iy++)
00959     for (ip = 0; ip < met->np; ip++) {
00960       met->ps[met->nx - 1][iy] = met->ps[0][iy];
00961       met->u[met->nx - 1][iy][ip] = met->u[0][iy][ip];
00962       met->v[met->nx - 1][iy][ip] = met->v[0][iy][ip];
00963       met->w[met->nx - 1][iy][ip] = met->w[0][iy][ip];
00964       met->t[met->nx - 1][iy][ip] = met->t[0][iy][ip];
00965       met->h2o[met->nx - 1][iy][ip] = met->h2o[0][iy][ip];
00966       met->o3[met->nx - 1][iy][ip] = met->o3[0][iy][ip];
00967     }
00968 }
```

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

Read a control parameter from file or command line.

Definition at line 972 of file libtrac.c.

```
00979                {
00980
00981   FILE *in = NULL;
00982
00983   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
00984     msg[LEN], rvarname[LEN], rval[LEN];
00985
00986   int contain = 0, i;
00987
00988   /* Open file... */
00989   if (filename[strlen(filename) - 1] != '-')
00990     if (!(in = fopen(filename, "r")))
00991       ERRMSG("Cannot open file!");
00992
00993   /* Set full variable name... */
00994   if (arridx >= 0) {
00995     sprintf(fullname1, "%s[%d]", varname, arridx);
00996     sprintf(fullname2, "%s[*]", varname);
00997   } else {
00998     sprintf(fullname1, "%s", varname);
00999     sprintf(fullname2, "%s", varname);
01000   }
01001
01002   /* Read data... */
01003   if (in != NULL)
01004     while (fgets(line, LEN, in))
01005       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
01006         if (strcasecmp(rvarname, fullname1) == 0 ||
01007             strcasecmp(rvarname, fullname2) == 0) {
01008           contain = 1;
01009           break;
01010         }
01011   for (i = 1; i < argc - 1; i++)
```

```
01012     if (strcasecmp(argv[i], fullname1) == 0 ||
01013         strcasecmp(argv[i], fullname2) == 0) {
01014       sprintf(rval, "%s", argv[i + 1]);
01015       contain = 1;
01016       break;
01017     }
01018
01019   /* Close file... */
01020   if (in != NULL)
01021     fclose(in);
01022
01023   /* Check for missing variables... */
01024   if (!contain) {
01025     if (strlen(defvalue) > 0)
01026       sprintf(rval, "%s", defvalue);
01027     else {
01028       sprintf(msg, "Missing variable %s!\n", fullname1);
01029       ERRMSG(msg);
01030     }
01031   }
01032
01033   /* Write info... */
01034   printf("%s = %s\n", fullname1, rval);
01035
01036   /* Return values... */
01037   if (value != NULL)
01038     sprintf(value, "%s", rval);
01039   return atof(rval);
01040 }
```

5.11.2.25 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 1044 of file libtrac.c.

```
01052                 {
01053
01054   struct tm t0, t1;
01055
01056   t0.tm_year = 100;
01057   t0.tm_mon = 0;
01058   t0.tm_mday = 1;
01059   t0.tm_hour = 0;
01060   t0.tm_min = 0;
01061   t0.tm_sec = 0;
01062
01063   t1.tm_year = year - 1900;
01064   t1.tm_mon = mon - 1;
01065   t1.tm_mday = day;
01066   t1.tm_hour = hour;
01067   t1.tm_min = min;
01068   t1.tm_sec = sec;
01069
01070   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
01071 }
```

5.11.2.26 void timer (const char * *name*, int *id*, int *mode*)

Measure wall-clock time.

Definition at line 1075 of file libtrac.c.

```
01078             {
01079
01080   static double starttime[NTIMER], runtime[NTIMER];
01081
01082   /* Check id... */
01083   if (id < 0 || id >= NTIMER)
01084     ERRMSG("Too many timers!");
01085
01086   /* Start timer... */
01087   if (mode == 1) {
01088     if (starttime[id] <= 0)
```

```
01089       starttime[id] = omp_get_wtime();
01090     else
01091       ERRMSG("Timer already started!");
01092   }
01093
01094   /* Stop timer... */
01095   else if (mode == 2) {
01096     if (starttime[id] > 0) {
01097       runtime[id] = runtime[id] + omp_get_wtime() - starttime[id];
01098       starttime[id] = -1;
01099     } else
01100       ERRMSG("Timer not started!");
01101   }
01102
01103   /* Print timer... */
01104   else if (mode == 3)
01105     printf("%s = %g s\n", name, runtime[id]);
01106 }
```

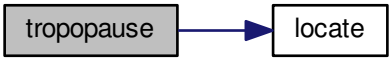
5.11.2.27 double tropopause (double *t*, double *lat*)

Definition at line 1110 of file libtrac.c.

```
01112               {
01113
01114   static double doys[12]
01115   = { 1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 };
01116
01117   static double lats[73]
01118     = { -90, -87.5, -85, -82.5, -80, -77.5, -75, -72.5, -70, -67.5,
01119     -65, -62.5, -60, -57.5, -55, -52.5, -50, -47.5, -45, -42.5,
01120     -40, -37.5, -35, -32.5, -30, -27.5, -25, -22.5, -20, -17.5,
01121     -15, -12.5, -10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5, 10, 12.5,
01122     15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40, 42.5,
01123     45, 47.5, 50, 52.5, 55, 57.5, 60, 62.5, 65, 67.5, 70, 72.5,
01124     75, 77.5, 80, 82.5, 85, 87.5, 90
01125   };
01126
01127   static double tps[12][73]
01128     = { {324.1, 325.6, 325, 324.3, 322.5, 319.7, 314, 307.2, 301.8, 299.6,
01129          297.1, 292.2, 285.6, 276.1, 264, 248.9, 231.9, 213.5, 194.4,
01130          175.3, 157, 140.4, 126.7, 116.3, 109.5, 105.4, 103, 101.4, 100.4,
01131          99.69, 99.19, 98.84, 98.56, 98.39, 98.39, 98.42, 98.44, 98.54,
01132          98.68, 98.81, 98.89, 98.96, 99.12, 99.65, 101.4, 105.4, 113.5, 128,
01133          152.1, 184.7, 214, 234.1, 247.3, 255.8, 262.6, 267.7, 271.7, 275,
01134          277.2, 279, 280.1, 280.4, 280.6, 280.1, 279.3, 278.3, 276.8, 275.8,
01135          275.3, 275.6, 275.4, 274.1, 273.5},
01136   {337.3, 338.7, 337.8, 336.4, 333, 328.8, 321.1, 312.6, 306.6, 303.7,
01137    300.2, 293.8, 285.4, 273.8, 259.6, 242.7, 224.4, 205.2, 186, 167.5,
01138    150.3, 135, 122.8, 113.9, 108.2, 104.7, 102.5, 101.1, 100.2, 99.42,
01139    98.88, 98.52, 98.25, 98.09, 98.07, 98.1, 98.12, 98.2, 98.25, 98.27,
01140    98.26, 98.27, 98.36, 98.79, 100.2, 104.2, 113.7, 131.2, 159.5, 193,
01141    220.4, 238.1, 250.2, 258.1, 264.7, 269.7, 273.7, 277.3, 280.2, 282.8,
01142    284.9, 286.5, 288.1, 288.8, 289, 288.5, 287.2, 286.3, 286.1, 287.2,
01143    287.5, 286.2, 285.8},
01144   {335, 336, 335.7, 335.1, 332.3, 328.1, 320.6, 311.8, 305.1, 301.9,
01145    297.6, 290, 280.4, 268.3, 254.6, 239.6, 223.9, 207.9, 192.2, 176.9,
01146    161.7, 146.4, 132.2, 120.6, 112.3, 107.2, 104.3, 102.4, 101.3,
01147    100.4, 99.86, 99.47, 99.16, 98.97, 98.94, 98.97, 99, 99.09, 99.2,
01148    99.31, 99.35, 99.41, 99.51, 99.86, 101.1, 104.9, 114.3, 131, 156.8,
01149    186.3, 209.3, 224.6, 236.8, 246.3, 254.9, 262.3, 268.8, 274.8,
01150    279.9, 284.6, 288.6, 291.6, 294.9, 297.5, 299.8, 301.8, 303.1,
01151    304.3, 304.9, 306, 306.6, 306.2, 306},
01152   {306.2, 306.7, 305.7, 307.1, 307.3, 306.4, 301.8, 296.2, 292.4,
01153    290.3, 287.1, 280.9, 273.4, 264.3, 254.1, 242.8, 231, 219, 207.2,
01154    195.5, 183.3, 169.7, 154.7, 138.7, 124.1, 113.6, 107.8, 104.7,
01155    102.8, 101.7, 100.9, 100.4, 100, 99.79, 99.7, 99.66, 99.68, 99.79,
01156    99.94, 100.2, 100.5, 100.9, 101.4, 102.1, 103.4, 107, 115.2, 129.1,
01157    148.7, 171, 190.8, 205.6, 218.4, 229.4, 239.6, 248.6, 256.5,
01158    263.7, 270.3, 276.6, 282.6, 288.1, 294.5, 300.4, 306.3, 311.4,
01159    315.1, 318.3, 320.3, 322.2, 322.8, 321.5, 321.1},
01160   {266.5, 264.9, 260.8, 261, 262, 263, 261.3, 259.7, 259.2, 259.8,
01161    260.1, 258.6, 256.7, 253.6, 249.5, 243.9, 237.4, 230, 222.1, 213.9,
01162    205, 194.4, 180.4, 161.8, 140.7, 122.9, 112.1, 106.7, 104.1, 102.7,
01163    101.8, 101.4, 101.1, 101, 101, 101, 101.1, 101.2, 101.5, 101.9,
01164    102.4, 103, 103.8, 104.9, 106.8, 110.1, 115.6, 124, 135.2, 148.9,
01165    165.2, 181.3, 198, 211.8, 223.5, 233.8, 242.9, 251.5, 259, 266.2,
01166    273.1, 279.2, 286.2, 292.8, 299.6, 306, 311.1, 315.5, 318.8, 322.6,
01167    325.3, 325.8, 325.8},
01168   {220.1, 218.1, 210.8, 207.2, 207.6, 210.5, 211.4, 213.5, 217.3,
01169    222.4, 227.9, 232.8, 237.4, 240.8, 242.8, 243, 241.5, 238.6, 234.2,
```

```
01170     228.5, 221, 210.7, 195.1, 172.9, 147.8, 127.6, 115.6, 109.9, 107.1,
01171     105.7, 105, 104.8, 104.8, 104.9, 105, 105.1, 105.3, 105.5, 105.8,
01172     106.4, 107, 107.6, 108.1, 108.8, 110, 111.8, 114.2, 117.4, 121.6,
01173     127.9, 137.3, 151.2, 169.5, 189, 205.8, 218.9, 229.1, 237.8, 245,
01174     251.5, 257.1, 262.3, 268.2, 274, 280.4, 286.7, 292.4, 297.9, 302.9,
01175     308.5, 312.2, 313.1, 313.3},
01176    {187.4, 184.5, 173.3, 166.1, 165.4, 167.8, 169.6, 173.6, 179.6,
01177     187.9, 198.9, 210, 220.5, 229.2, 235.7, 239.9, 241.8, 241.6, 239.6,
01178     235.8, 229.4, 218.6, 200.9, 175.9, 149.4, 129.4, 118.3, 113.1,
01179     110.8, 109.7, 109.3, 109.4, 109.7, 110, 110.2, 110.4, 110.5, 110.7,
01180     111, 111.4, 111.8, 112.1, 112.3, 112.7, 113.2, 113.9, 115, 116.4,
01181     117.9, 120.4, 124.1, 130.9, 142.2, 159.6, 179.6, 198.5, 212.9,
01182     224.2, 232.7, 239.1, 243.8, 247.7, 252.4, 257.3, 263.2, 269.5,
01183     275.4, 281.1, 286.3, 292, 296.3, 298.2, 298.8},
01184    {166, 166.4, 155.7, 148.3, 147.1, 149, 152.1, 157, 163.6, 172.4,
01185     185.3, 199.2, 212.6, 224, 233.2, 239.6, 243.3, 244.6, 243.6, 240.3,
01186     233.9, 222.6, 203.7, 177, 149.5, 129.7, 119, 114, 111.7, 110.7,
01187     110.3, 110.3, 110.6, 110.9, 111.1, 111.3, 111.5, 111.6, 111.9,
01188     112.2, 112.5, 112.6, 112.8, 113, 113.4, 114, 115.1, 116.5, 118.3,
01189     120.9, 124.4, 130.2, 139.4, 154.6, 173.8, 193.1, 208.1, 220.4,
01190     230.1, 238.2, 244.7, 249.5, 254.5, 259.3, 264.5, 269.4, 273.7,
01191     278.2, 282.6, 287.4, 290.9, 292.5, 293},
01192    {171.9, 172.8, 166.2, 162.3, 161.4, 162.5, 165.2, 169.6, 175.3,
01193     183.1, 193.8, 205.9, 218.3, 229.6, 238.5, 244.3, 246.9, 246.7,
01194     243.8, 238.4, 230.2, 217.9, 199.6, 174.9, 148.9, 129.8, 119.5,
01195     114.8, 112.3, 110.9, 110.3, 110.1, 110.2, 110.3, 110.4, 110.5,
01196     110.6, 110.8, 111, 111.4, 111.8, 112, 112.2, 112.4, 112.9, 113.6,
01197     114.7, 116.3, 118.4, 121.9, 127.1, 136.1, 149.8, 168.4, 186.9,
01198     203.3, 217, 229.1, 238.7, 247, 254, 259.3, 264.3, 268.3, 272.5,
01199     276.6, 280.4, 284.4, 288.4, 293.3, 297.2, 298.7, 299.1},
01200    {191.6, 192.2, 189, 188.1, 190.2, 193.7, 197.8, 202.9, 208.5,
01201     215.6, 224.2, 233.1, 241.2, 247.3, 250.8, 251.3, 248.9, 244.2,
01202     237.3, 228.4, 217.2, 202.9, 184.5, 162.5, 140.7, 124.8, 116.2,
01203     111.8, 109.4, 107.9, 107, 106.7, 106.6, 106.6, 106.7, 106.7,
01204     106.8, 107, 107.4, 108, 108.7, 109.3, 109.8, 110.4, 111.2,
01205     112.4, 114.2, 116.9, 121.1, 127.9, 139.3, 155.2, 173.6, 190.7,
01206     206.1, 220.1, 232.3, 243, 251.8, 259.2, 265.7, 270.6, 275.3,
01207     279.3, 283.3, 286.9, 289.7, 292.8, 296.1, 300.5, 303.9, 304.8,
01208     305.1},
01209    {241.5, 239.6, 236.8, 237.4, 239.4, 242.3, 244.2, 246.4, 249.2,
01210     253.6, 258.6, 262.7, 264.8, 264.2, 260.6, 254.1, 245.5, 235.3,
01211     223.9, 211.7, 198.3, 183.1, 165.6, 147.1, 130.5, 118.7, 111.9,
01212     108.1, 105.8, 104.3, 103.4, 102.8, 102.5, 102.4, 102.5, 102.5,
01213     102.5, 102.7, 103.1, 103.8, 104.6, 105.4, 106.1, 107, 108.2,
01214     109.9, 112.8, 117.5, 126, 140.4, 161, 181.9, 201.2, 216.8, 230.4,
01215     241.8, 251.4, 259.9, 266.9, 272.8, 277.4, 280.4, 282.9, 284.6,
01216     286.1, 287.4, 288.3, 289.5, 290.9, 294.2, 296.9, 297.5, 297.6},
01217    {301.2, 300.3, 296.6, 295.4, 295, 294.3, 291.2, 287.4, 284.9, 284.7,
01218     284.1, 281.5, 277.1, 270.4, 261.7, 250.6, 237.6, 223.1, 207.9, 192,
01219     175.8, 158.8, 142.1, 127.6, 116.8, 109.9, 106, 103.6, 102.1, 101.1,
01220     100.4, 99.96, 99.6, 99.37, 99.32, 99.32, 99.31, 99.46, 99.77, 100.2,
01221     100.7, 101.3, 101.8, 102.7, 104.1, 106.8, 111.9, 121, 136.7, 160,
01222     186.9, 209.9, 228.1, 241.2, 251.5, 259.5, 265.7, 270.9, 274.8, 278,
01223     280.3, 281.8, 283, 283.3, 283.7, 283.8, 283, 282.2, 281.2, 281.4,
01224     281.7, 281.1, 281.2}
01225    };
01226
01227    double doy, p0, p1, pt;
01228
01229    int imon, ilat;
01230
01231    /* Get day of year... */
01232    doy = fmod(t / 86400., 365.25);
01233    while (doy < 0)
01234      doy += 365.25;
01235
01236    /* Get indices... */
01237    imon = locate(doys, 12, doy);
01238    ilat = locate(lats, 73, lat);
01239
01240    /* Get tropopause pressure... */
01241    p0 = LIN(lats[ilat], tps[imon][ilat],
01242             lats[ilat + 1], tps[imon][ilat + 1], lat);
01243    p1 = LIN(lats[ilat], tps[imon + 1][ilat],
01244             lats[ilat + 1], tps[imon + 1][ilat + 1], lat);
01245    pt = LIN(doys[imon], p0, doys[imon + 1], p1, doy);
01246
01247    /* Return tropopause pressure... */
01248    return pt;
01249  }
```

Here is the call graph for this function:



5.11.2.28 void write_atm (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write atmospheric data.

Definition at line 1253 of file libtrac.c.

```
01257            {
01258
01259   FILE *in, *out;
01260
01261   char line[LEN];
01262
01263   double r, t0, t1;
01264
01265   int ip, iq, year, mon, day, hour, min, sec;
01266
01267   /* Set time interval for output... */
01268   t0 = t - 0.5 * ctl->dt_mod;
01269   t1 = t + 0.5 * ctl->dt_mod;
01270
01271   /* Check if gnuplot output is requested... */
01272   if (ctl->atm_gpfile[0] != '-') {
01273
01274     /* Write info... */
01275     printf("Plot atmospheric data: %s.png\n", filename);
01276
01277     /* Create gnuplot pipe... */
01278     if (!(out = popen("gnuplot", "w")))
01279       ERRMSG("Cannot create pipe to gnuplot!");
01280
01281     /* Set plot filename... */
01282     fprintf(out, "set out \"%s.png\"\n", filename);
01283
01284     /* Set time string... */
01285     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01286     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01287             year, mon, day, hour, min);
01288
01289     /* Dump gnuplot file to pipe... */
01290     if (!(in = fopen(ctl->atm_gpfile, "r")))
01291       ERRMSG("Cannot open file!");
01292     while (fgets(line, LEN, in))
01293       fprintf(out, "%s", line);
01294     fclose(in);
01295   }
01296
01297   else {
01298
01299     /* Write info... */
01300     printf("Write atmospheric data: %s\n", filename);
01301
01302     /* Create file... */
01303     if (!(out = fopen(filename, "w")))
01304       ERRMSG("Cannot create file!");
01305   }
01306
01307   /* Write header... */
01308   fprintf(out,
01309           "# $1 = time [s]\n"
01310           "# $2 = altitude [km]\n"
01311           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01312   for (iq = 0; iq < ctl->nq; iq++)
01313     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl->qnt_name[iq],
```

```
01314             ctl->qnt_unit[iq]);
01315   fprintf(out, "\n");
01316
01317   /* Write data... */
01318   for (ip = 0; ip < atm->np; ip++) {
01319
01320     /* Check time... */
01321     if (ctl->atm_filter && (atm->time[ip] < t0 || atm->time[ip] > t1))
01322       continue;
01323
01324     /* Write output... */
01325     fprintf(out, "%.2f %g %g %g", atm->time[ip], Z(atm->p[ip]),
01326             atm->lon[ip], atm->lat[ip]);
01327     for (iq = 0; iq < ctl->nq; iq++) {
01328       fprintf(out, " ");
01329       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
01330     }
01331     fprintf(out, "\n");
01332   }
01333
01334   /* Close file... */
01335   fclose(out);
01336 }
```

Here is the call graph for this function:

5.11.2.29 void write_csi (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write CSI data.

Definition at line 1340 of file libtrac.c.

```
01344             {
01345
01346   static FILE *in, *out;
01347
01348   static char line[LEN];
01349
01350   static double modmean[GX][GY][GZ], obsmean[GX][GY][GZ],
01351     rt, rz, rlon, rlat, robs, t0, t1, area, dlon, dlat, lat;
01352
01353   static int init, obscount[GX][GY][GZ], cx, cy, cz, ip, ix, iy, iz;
01354
01355   /* Init... */
01356   if (!init) {
01357     init = 1;
01358
01359     /* Check quantity index for mass... */
01360     if (ctl->qnt_m < 0)
01361       ERRMSG("Need quantity mass to analyze CSI!");
01362
01363     /* Open observation data file... */
01364     printf("Read CSI observation data: %s\n", ctl->csi_obsfile);
01365     if (!(in = fopen(ctl->csi_obsfile, "r")))
01366       ERRMSG("Cannot open file!");
01367
01368     /* Create new file... */
01369     printf("Write CSI data: %s\n", filename);
01370     if (!(out = fopen(filename, "w")))
01371       ERRMSG("Cannot create file!");
01372
01373     /* Write header... */
```

```
01374     fprintf(out,
01375             "# $1 = time [s]\n"
01376             "# $2 = number of hits (cx)\n"
01377             "# $3 = number of misses (cy)\n"
01378             "# $4 = number of false alarms (cz)\n"
01379             "# $5 = number of observations (cx + cy)\n"
01380             "# $6 = number of forecasts (cx + cz)\n"
01381             "# $7 = bias (forecasts/observations) [%%]\n"
01382             "# $8 = probability of detection (POD) [%%]\n"
01383             "# $9 = false alarm rate (FAR) [%%]\n"
01384             "# $10 = critical success index (CSI) [%%]\n\n");
01385   }
01386
01387   /* Set time interval... */
01388   t0 = t - 0.5 * ctl->dt_mod;
01389   t1 = t + 0.5 * ctl->dt_mod;
01390
01391   /* Initialize grid cells... */
01392   for (ix = 0; ix < ctl->csi_nx; ix++)
01393     for (iy = 0; iy < ctl->csi_ny; iy++)
01394       for (iz = 0; iz < ctl->csi_nz; iz++)
01395         modmean[ix][iy][iz] = obsmean[ix][iy][iz] = obscount[ix][iy][iz] = 0;
01396
01397   /* Read data... */
01398   while (fgets(line, LEN, in)) {
01399
01400     /* Read data... */
01401     if (sscanf(line, "%lg %lg %lg %lg %lg", &rt, &rz, &rlon, &rlat, &robs) !=
01402         5)
01403       continue;
01404
01405     /* Check time... */
01406     if (rt < t0)
01407       continue;
01408     if (rt > t1)
01409       break;
01410
01411     /* Calculate indices... */
01412     ix = (int) ((rlon - ctl->csi_lon0)
01413                 / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01414     iy = (int) ((rlat - ctl->csi_lat0)
01415                 / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01416     iz = (int) ((rz - ctl->csi_z0)
01417                 / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01418
01419     /* Check indices... */
01420     if (ix < 0 || ix >= ctl->csi_nx ||
01421         iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01422       continue;
01423
01424     /* Get mean observation index... */
01425     obsmean[ix][iy][iz] += robs;
01426     obscount[ix][iy][iz]++;
01427   }
01428
01429   /* Analyze model data... */
01430   for (ip = 0; ip < atm->np; ip++) {
01431
01432     /* Check time... */
01433     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01434       continue;
01435
01436     /* Get indices... */
01437     ix = (int) ((atm->lon[ip] - ctl->csi_lon0)
01438                 / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01439     iy = (int) ((atm->lat[ip] - ctl->csi_lat0)
01440                 / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01441     iz = (int) ((Z(atm->p[ip]) - ctl->csi_z0)
01442                 / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01443
01444     /* Check indices... */
01445     if (ix < 0 || ix >= ctl->csi_nx ||
01446         iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01447       continue;
01448
01449     /* Get total mass in grid cell... */
01450     modmean[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01451   }
01452
01453   /* Analyze all grid cells... */
01454   for (ix = 0; ix < ctl->csi_nx; ix++)
01455     for (iy = 0; iy < ctl->csi_ny; iy++)
01456       for (iz = 0; iz < ctl->csi_nz; iz++) {
01457
01458         /* Calculate mean observation index... */
01459         if (obscount[ix][iy][iz] > 0)
01460           obsmean[ix][iy][iz] /= obscount[ix][iy][iz];
```

```
01461
01462         /* Calculate column density... */
01463         if (modmean[ix][iy][iz] > 0) {
01464           dlon = (ctl->csi_lon1 - ctl->csi_lon0) / ctl->csi_nx;
01465           dlat = (ctl->csi_lat1 - ctl->csi_lat0) / ctl->csi_ny;
01466           lat = ctl->csi_lat0 + dlat * (iy + 0.5);
01467           area = dlat * M_PI * RE / 180. * dlon * M_PI * RE / 180.
01468             * cos(lat * M_PI / 180.);
01469           modmean[ix][iy][iz] /= (1e6 * area);
01470         }
01471
01472         /* Calculate CSI... */
01473         if (obscount[ix][iy][iz] > 0) {
01474           if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01475               modmean[ix][iy][iz] >= ctl->csi_modmin)
01476             cx++;
01477           else if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01478                    modmean[ix][iy][iz] < ctl->csi_modmin)
01479             cy++;
01480           else if (obsmean[ix][iy][iz] < ctl->csi_obsmin &&
01481                    modmean[ix][iy][iz] >= ctl->csi_modmin)
01482             cz++;
01483         }
01484       }
01485
01486   /* Write output... */
01487   if (fmod(t, ctl->csi_dt_out) == 0) {
01488
01489     /* Write... */
01490     fprintf(out, "%.2f %d %d %d %d %d %g %g %g %g\n",
01491             t, cx, cy, cz, cx + cy, cx + cz,
01492             (cx + cy > 0) ? 100. * (cx + cz) / (cx + cy) : GSL_NAN,
01493             (cx + cy > 0) ? (100. * cx) / (cx + cy) : GSL_NAN,
01494             (cx + cz > 0) ? (100. * cz) / (cx + cz) : GSL_NAN,
01495             (cx + cy + cz > 0) ? (100. * cx) / (cx + cy + cz) : GSL_NAN);
01496
01497     /* Set counters to zero... */
01498     cx = cy = cz = 0;
01499   }
01500
01501   /* Close file... */
01502   if (t == ctl->t_stop)
01503     fclose(out);
01504 }
```

5.11.2.30 void write_ens (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write ensemble data.

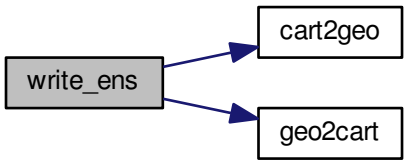
Definition at line 1508 of file libtrac.c.

```
01512             {
01513
01514   static FILE *out;
01515
01516   static double dummy, ens, lat, lon, p[NENS], q[NQ][NENS],
01517     t0, t1, x[NENS][3], xm[3];
01518
01519   static int init, ip, iq;
01520
01521   static size_t i, n;
01522
01523   /* Init... */
01524   if (!init) {
01525     init = 1;
01526
01527     /* Check quantities... */
01528     if (ctl->qnt_ens < 0)
01529       ERRMSG("Missing ensemble IDs!");
01530
01531     /* Create new file... */
01532     printf("Write ensemble data: %s\n", filename);
01533     if (!(out = fopen(filename, "w")))
01534       ERRMSG("Cannot create file!");
01535
01536     /* Write header... */
01537     fprintf(out,
01538             "# $1 = time [s]\n"
01539             "# $2 = altitude [km]\n"
01540             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
```

```
01541     for (iq = 0; iq < ctl->nq; iq++)
01542       fprintf(out, "# $%d = %s (mean) [%s]\n", 5 + iq,
01543               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01544     for (iq = 0; iq < ctl->nq; iq++)
01545       fprintf(out, "# $%d = %s (sigma) [%s]\n", 5 + ctl->nq + iq,
01546               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01547     fprintf(out, "# $%d = number of members\n\n", 5 + 2 * ctl->nq);
01548   }
01549
01550   /* Set time interval... */
01551   t0 = t - 0.5 * ctl->dt_mod;
01552   t1 = t + 0.5 * ctl->dt_mod;
01553
01554   /* Init... */
01555   ens = GSL_NAN;
01556   n = 0;
01557
01558   /* Loop over air parcels... */
01559   for (ip = 0; ip < atm->np; ip++) {
01560
01561     /* Check time... */
01562     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01563       continue;
01564
01565     /* Check ensemble id... */
01566     if (atm->q[ctl->qnt_ens][ip] != ens) {
01567
01568       /* Write results... */
01569       if (n > 0) {
01570
01571         /* Get mean position... */
01572         xm[0] = xm[1] = xm[2] = 0;
01573         for (i = 0; i < n; i++) {
01574           xm[0] += x[i][0] / (double) n;
01575           xm[1] += x[i][1] / (double) n;
01576           xm[2] += x[i][2] / (double) n;
01577         }
01578         cart2geo(xm, &dummy, &lon, &lat);
01579         fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon,
01580                 lat);
01581
01582         /* Get quantity statistics... */
01583         for (iq = 0; iq < ctl->nq; iq++) {
01584           fprintf(out, " ");
01585           fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01586         }
01587         for (iq = 0; iq < ctl->nq; iq++) {
01588           fprintf(out, " ");
01589           fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01590         }
01591         fprintf(out, " %lu\n", n);
01592       }
01593
01594       /* Init new ensemble... */
01595       ens = atm->q[ctl->qnt_ens][ip];
01596       n = 0;
01597     }
01598
01599     /* Save data... */
01600     p[n] = atm->p[ip];
01601     geo2cart(0, atm->lon[ip], atm->lat[ip], x[n]);
01602     for (iq = 0; iq < ctl->nq; iq++)
01603       q[iq][n] = atm->q[iq][ip];
01604     if ((++n) >= NENS)
01605       ERRMSG("Too many data points!");
01606   }
01607
01608   /* Write results... */
01609   if (n > 0) {
01610
01611     /* Get mean position... */
01612     xm[0] = xm[1] = xm[2] = 0;
01613     for (i = 0; i < n; i++) {
01614       xm[0] += x[i][0] / (double) n;
01615       xm[1] += x[i][1] / (double) n;
01616       xm[2] += x[i][2] / (double) n;
01617     }
01618     cart2geo(xm, &dummy, &lon, &lat);
01619     fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon, lat);
01620
01621     /* Get quantity statistics... */
01622     for (iq = 0; iq < ctl->nq; iq++) {
01623       fprintf(out, " ");
01624       fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01625     }
01626     for (iq = 0; iq < ctl->nq; iq++) {
01627       fprintf(out, " ");
```

```
01628       fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01629     }
01630     fprintf(out, " %lu\n", n);
01631   }
01632
01633   /* Close file... */
01634   if (t == ctl->t_stop)
01635     fclose(out);
01636 }
```

Here is the call graph for this function:



5.11.2.31 void write_grid (const char * *filename*, ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, double *t*)

Write gridded data.

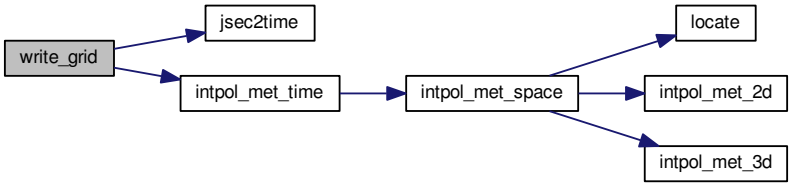
Definition at line 1640 of file libtrac.c.

```
01646             {
01647
01648   FILE *in, *out;
01649
01650   char line[LEN];
01651
01652   static double grid_m[GX][GY][GZ], z, dz, lon, dlon, lat, dlat,
01653     area, rho_air, press, temp, cd, mmr, t0, t1, r;
01654
01655   static int ip, ix, iy, iz, year, mon, day, hour, min, sec;
01656
01657   /* Check dimensions... */
01658   if (ctl->grid_nx > GX || ctl->grid_ny > GY || ctl->grid_nz > GZ)
01659     ERRMSG("Grid dimensions too large!");
01660
01661   /* Check quantity index for mass... */
01662   if (ctl->qnt_m < 0)
01663     ERRMSG("Need quantity mass to write grid data!");
01664
01665   /* Set time interval for output... */
01666   t0 = t - 0.5 * ctl->dt_mod;
01667   t1 = t + 0.5 * ctl->dt_mod;
01668
01669   /* Set grid box size... */
01670   dz = (ctl->grid_z1 - ctl->grid_z0) / ctl->grid_nz;
01671   dlon = (ctl->grid_lon1 - ctl->grid_lon0) / ctl->grid_nx;
01672   dlat = (ctl->grid_lat1 - ctl->grid_lat0) / ctl->grid_ny;
01673
01674   /* Initialize grid... */
01675   for (ix = 0; ix < ctl->grid_nx; ix++)
01676     for (iy = 0; iy < ctl->grid_ny; iy++)
01677       for (iz = 0; iz < ctl->grid_nz; iz++)
01678         grid_m[ix][iy][iz] = 0;
01679
01680   /* Average data... */
01681   for (ip = 0; ip < atm->np; ip++)
01682     if (atm->time[ip] >= t0 && atm->time[ip] <= t1) {
01683
01684       /* Get index... */
01685       ix = (int) ((atm->lon[ip] - ctl->grid_lon0) / dlon);
```

```
01686       iy = (int) ((atm->lat[ip] - ctl->grid_lat0) / dlat);
01687       iz = (int) ((Z(atm->p[ip]) - ctl->grid_z0) / dz);
01688
01689       /* Check indices... */
01690       if (ix < 0 || ix >= ctl->grid_nx ||
01691           iy < 0 || iy >= ctl->grid_ny || iz < 0 || iz >= ctl->grid_nz)
01692         continue;
01693
01694       /* Add mass... */
01695       grid_m[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01696     }
01697
01698   /* Check if gnuplot output is requested... */
01699   if (ctl->grid_gpfile[0] != '-') {
01700
01701     /* Write info... */
01702     printf("Plot grid data: %s.png\n", filename);
01703
01704     /* Create gnuplot pipe... */
01705     if (!(out = popen("gnuplot", "w")))
01706       ERRMSG("Cannot create pipe to gnuplot!");
01707
01708     /* Set plot filename... */
01709     fprintf(out, "set out \"%s.png\"\n", filename);
01710
01711     /* Set time string... */
01712     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01713     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01714             year, mon, day, hour, min);
01715
01716     /* Dump gnuplot file to pipe... */
01717     if (!(in = fopen(ctl->grid_gpfile, "r")))
01718       ERRMSG("Cannot open file!");
01719     while (fgets(line, LEN, in))
01720       fprintf(out, "%s", line);
01721     fclose(in);
01722   }
01723
01724   else {
01725
01726     /* Write info... */
01727     printf("Write grid data: %s\n", filename);
01728
01729     /* Create file... */
01730     if (!(out = fopen(filename, "w")))
01731       ERRMSG("Cannot create file!");
01732   }
01733
01734   /* Write header... */
01735   fprintf(out,
01736           "# $1 = time [s]\n"
01737           "# $2 = altitude [km]\n"
01738           "# $3 = longitude [deg]\n"
01739           "# $4 = latitude [deg]\n"
01740           "# $5 = surface area [km^2]\n"
01741           "# $6 = layer width [km]\n"
01742           "# $7 = temperature [K]\n"
01743           "# $8 = column density [kg/m^2]\n"
01744           "# $9 = mass mixing ratio [1]\n\n");
01745
01746   /* Write data... */
01747   for (ix = 0; ix < ctl->grid_nx; ix++) {
01748     if (ix > 0 && ctl->grid_ny > 1 && !ctl->grid_sparse)
01749       fprintf(out, "\n");
01750     for (iy = 0; iy < ctl->grid_ny; iy++) {
01751       if (iy > 0 && ctl->grid_nz > 1 && !ctl->grid_sparse)
01752         fprintf(out, "\n");
01753       for (iz = 0; iz < ctl->grid_nz; iz++)
01754         if (!ctl->grid_sparse
01755             || ix == 0 || iy == 0 || iz == 0 || grid_m[ix][iy][iz] > 0) {
01756
01757           /* Set coordinates... */
01758           z = ctl->grid_z0 + dz * (iz + 0.5);
01759           lon = ctl->grid_lon0 + dlon * (ix + 0.5);
01760           lat = ctl->grid_lat0 + dlat * (iy + 0.5);
01761
01762           /* Get pressure and temperature... */
01763           press = P(z);
01764           intpol_met_time(met0, met1, t, press, lon, lat,
01765                           NULL, &temp, NULL, NULL, NULL, NULL, NULL);
01766
01767           /* Calculate surface area... */
01768           area = dlat * dlon * gsl_pow_2(RE * M_PI / 180.)
01769             * cos(lat * M_PI / 180.);
01770
01771           /* Calculate column density... */
01772           cd = grid_m[ix][iy][iz] / (1e6 * area);
```

```
01773
01774           /* Calculate mass mixing ratio... */
01775           rho_air = 100. * press / (287.058 * temp);
01776           mmr = grid_m[ix][iy][iz] / (rho_air * 1e6 * area * 1e3 * dz);
01777
01778           /* Write output... */
01779           fprintf(out, "%.2f %g %g %g %g %g %g %g %g\n",
01780                   t, z, lon, lat, area, dz, temp, cd, mmr);
01781         }
01782     }
01783   }
01784
01785   /* Close file... */
01786   fclose(out);
01787 }
```

Here is the call graph for this function:



5.11.2.32 void write_prof (const char * *filename*, ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, double *t*)

Write profile data.

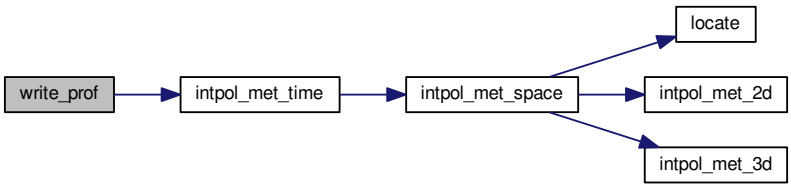
Definition at line 1791 of file libtrac.c.

```
01797           {
01798
01799   static FILE *in, *out;
01800
01801   static char line[LEN];
01802
01803   static double mass[GX][GY][GZ], obsmean[GX][GY], tmean[GX][GY],
01804     rt, rlon, rlat, robs, t0, t1, area, dz, dlon, dlat, lon, lat, z,
01805     press, temp, rho_air, mmr, h2o, o3;
01806
01807   static int init, obscount[GX][GY], ip, ix, iy, iz;
01808
01809   /* Init... */
01810   if (!init) {
01811     init = 1;
01812
01813     /* Check quantity index for mass... */
01814     if (ctl->qnt_m < 0)
01815       ERRMSG("Need quantity mass!");
01816
01817     /* Check dimensions... */
01818     if (ctl->prof_nx > GX || ctl->prof_ny > GY || ctl->prof_nz > GZ)
01819       ERRMSG("Grid dimensions too large!");
01820
01821     /* Open observation data file... */
01822     printf("Read profile observation data: %s\n", ctl->prof_obsfile);
01823     if (!(in = fopen(ctl->prof_obsfile, "r")))
01824       ERRMSG("Cannot open file!");
01825
01826     /* Create new file... */
01827     printf("Write profile data: %s\n", filename);
01828     if (!(out = fopen(filename, "w")))
01829       ERRMSG("Cannot create file!");
01830
01831     /* Write header... */
01832     fprintf(out,
```

```
01833             "# $1  = time [s]\n"
01834             "# $2  = altitude [km]\n"
01835             "# $3  = longitude [deg]\n"
01836             "# $4  = latitude [deg]\n"
01837             "# $5  = pressure [hPa]\n"
01838             "# $6  = temperature [K]\n"
01839             "# $7  = mass mixing ratio [1]\n"
01840             "# $8  = H2O volume mixing ratio [1]\n"
01841             "# $9  = O3 volume mixing ratio [1]\n"
01842             "# $10 = mean BT index [K]\n");
01843
01844     /* Set grid box size... */
01845     dz = (ctl->prof_z1 - ctl->prof_z0) / ctl->prof_nz;
01846     dlon = (ctl->prof_lon1 - ctl->prof_lon0) / ctl->prof_nx;
01847     dlat = (ctl->prof_lat1 - ctl->prof_lat0) / ctl->prof_ny;
01848   }
01849
01850   /* Set time interval... */
01851   t0 = t - 0.5 * ctl->dt_mod;
01852   t1 = t + 0.5 * ctl->dt_mod;
01853
01854   /* Initialize... */
01855   for (ix = 0; ix < ctl->prof_nx; ix++)
01856     for (iy = 0; iy < ctl->prof_ny; iy++) {
01857       obsmean[ix][iy] = 0;
01858       obscount[ix][iy] = 0;
01859       tmean[ix][iy] = 0;
01860       for (iz = 0; iz < ctl->prof_nz; iz++)
01861         mass[ix][iy][iz] = 0;
01862     }
01863
01864   /* Read data... */
01865   while (fgets(line, LEN, in)) {
01866
01867     /* Read data... */
01868     if (sscanf(line, "%lg %lg %lg %lg", &rt, &rlon, &rlat, &robs) != 4)
01869       continue;
01870
01871     /* Check time... */
01872     if (rt < t0)
01873       continue;
01874     if (rt > t1)
01875       break;
01876
01877     /* Calculate indices... */
01878     ix = (int) ((rlon - ctl->prof_lon0) / dlon);
01879     iy = (int) ((rlat - ctl->prof_lat0) / dlat);
01880
01881     /* Check indices... */
01882     if (ix < 0 || ix >= ctl->prof_nx || iy < 0 || iy >= ctl->prof_ny)
01883       continue;
01884
01885     /* Get mean observation index... */
01886     obsmean[ix][iy] += robs;
01887     tmean[ix][iy] += rt;
01888     obscount[ix][iy]++;
01889   }
01890
01891   /* Analyze model data... */
01892   for (ip = 0; ip < atm->np; ip++) {
01893
01894     /* Check time... */
01895     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01896       continue;
01897
01898     /* Get indices... */
01899     ix = (int) ((atm->lon[ip] - ctl->prof_lon0) / dlon);
01900     iy = (int) ((atm->lat[ip] - ctl->prof_lat0) / dlat);
01901     iz = (int) ((Z(atm->p[ip]) - ctl->prof_z0) / dz);
01902
01903     /* Check indices... */
01904     if (ix < 0 || ix >= ctl->prof_nx ||
01905         iy < 0 || iy >= ctl->prof_ny || iz < 0 || iz >= ctl->prof_nz)
01906       continue;
01907
01908     /* Get total mass in grid cell... */
01909     mass[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01910   }
01911
01912   /* Extract profiles... */
01913   for (ix = 0; ix < ctl->prof_nx; ix++)
01914     for (iy = 0; iy < ctl->prof_ny; iy++)
01915       if (obscount[ix][iy] > 0) {
01916
01917         /* Write output... */
01918         fprintf(out, "\n");
01919
```

```
01920         /* Loop over altitudes... */
01921         for (iz = 0; iz < ctl->prof_nz; iz++) {
01922
01923           /* Set coordinates... */
01924           z = ctl->prof_z0 + dz * (iz + 0.5);
01925           lon = ctl->prof_lon0 + dlon * (ix + 0.5);
01926           lat = ctl->prof_lat0 + dlat * (iy + 0.5);
01927
01928           /* Get meteorological data... */
01929           press = P(z);
01930           intpol_met_time(met0, met1, t, press, lon, lat,
01931                           NULL, &temp, NULL, NULL, NULL, &h2o, &o3);
01932
01933           /* Calculate mass mixing ratio... */
01934           rho_air = 100. * press / (287.058 * temp);
01935           area = dlat * dlon * gsl_pow_2(M_PI * RE / 180.)
01936             * cos(lat * M_PI / 180.);
01937           mmr = mass[ix][iy][iz] / (rho_air * area * dz * 1e9);
01938
01939           /* Write output... */
01940           fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g\n",
01941                   tmean[ix][iy] / obscount[ix][iy],
01942                   z, lon, lat, press, temp, mmr, h2o, o3,
01943                   obsmean[ix][iy] / obscount[ix][iy]);
01944         }
01945       }
01946
01947   /* Close file... */
01948   if (t == ctl->t_stop)
01949     fclose(out);
01950 }
```

Here is the call graph for this function:



5.11.2.33 void write_station (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

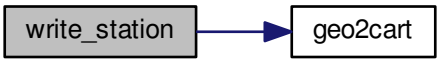
Write station data.

Definition at line 1954 of file libtrac.c.

```
01958             {
01959
01960   static FILE *out;
01961
01962   static double rmax2, t0, t1, x0[3], x1[3];
01963
01964   static int init, ip, iq;
01965
01966   /* Init... */
01967   if (!init) {
01968     init = 1;
01969
01970     /* Write info... */
01971     printf("Write station data: %s\n", filename);
01972
01973     /* Create new file... */
01974     if (!(out = fopen(filename, "w")))
01975       ERRMSG("Cannot create file!");
01976
01977     /* Write header... */
```

```
01978     fprintf(out,
01979             "# $1 = time [s]\n"
01980             "# $2 = altitude [km]\n"
01981             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01982     for (iq = 0; iq < ctl->nq; iq++)
01983       fprintf(out, "# $%i = %s [%s]\n", (iq + 5),
01984               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01985     fprintf(out, "\n");
01986
01987     /* Set geolocation and search radius... */
01988     geo2cart(0, ctl->stat_lon, ctl->stat_lat, x0);
01989     rmax2 = gsl_pow_2(ctl->stat_r);
01990   }
01991
01992   /* Set time interval for output... */
01993   t0 = t - 0.5 * ctl->dt_mod;
01994   t1 = t + 0.5 * ctl->dt_mod;
01995
01996   /* Loop over air parcels... */
01997   for (ip = 0; ip < atm->np; ip++) {
01998
01999     /* Check time... */
02000     if (atm->time[ip] < t0 || atm->time[ip] > t1)
02001       continue;
02002
02003     /* Check station flag... */
02004     if (ctl->qnt_stat >= 0)
02005       if (atm->q[ctl->qnt_stat][ip])
02006         continue;
02007
02008     /* Get Cartesian coordinates... */
02009     geo2cart(0, atm->lon[ip], atm->lat[ip], x1);
02010
02011     /* Check horizontal distance... */
02012     if (DIST2(x0, x1) > rmax2)
02013       continue;
02014
02015     /* Set station flag... */
02016     if (ctl->qnt_stat >= 0)
02017       atm->q[ctl->qnt_stat][ip] = 1;
02018
02019     /* Write data... */
02020     fprintf(out, "%.2f %g %g %g",
02021             atm->time[ip], Z(atm->p[ip]), atm->lon[ip], atm->lat[ip]);
02022     for (iq = 0; iq < ctl->nq; iq++) {
02023       fprintf(out, " ");
02024       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
02025     }
02026     fprintf(out, "\n");
02027   }
02028
02029   /* Close file... */
02030   if (t == ctl->t_stop)
02031     fclose(out);
02032 }
```

Here is the call graph for this function:



5.12 libtrac.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 /*****************************************************************************/
00028
00029 void cart2geo(
00030   double *x,
00031   double *z,
00032   double *lon,
00033   double *lat) {
00034
00035   double radius;
00036
00037   radius = NORM(x);
00038   *lat = asin(x[2] / radius) * 180 / M_PI;
00039   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00040   *z = radius - RE;
00041 }
00042
00043 /*****************************************************************************/
00044
00045 double deg2dx(
00046   double dlon,
00047   double lat) {
00048
00049   return dlon * M_PI * RE / 180. * cos(lat / 180. * M_PI);
00050 }
00051
00052 /*****************************************************************************/
00053
00054 double deg2dy(
00055   double dlat) {
00056
00057   return dlat * M_PI * RE / 180.;
00058 }
00059
00060 /*****************************************************************************/
00061
00062 double dp2dz(
00063   double dp,
00064   double p) {
00065
00066   return -dp * H0 / p;
00067 }
00068
00069 /*****************************************************************************/
00070
00071 double dx2deg(
00072   double dx,
00073   double lat) {
00074
00075   /* Avoid singularity at poles... */
00076   if (lat < -89.999 || lat > 89.999)
00077     return 0;
00078   else
00079     return dx * 180. / (M_PI * RE * cos(lat / 180. * M_PI));
00080 }
00081
00082 /*****************************************************************************/
00083
00084 double dy2deg(
00085   double dy) {
00086
00087   return dy * 180. / (M_PI * RE);
00088 }
00089
00090 /*****************************************************************************/
00091
00092 double dz2dp(
00093   double dz,
00094   double p) {
00095
00096   return -dz * p / H0;
00097 }
```

```
00098
00099 /*****************************************************************************/
00100
00101 void geo2cart(
00102   double z,
00103   double lon,
00104   double lat,
00105   double *x) {
00106
00107   double radius;
00108
00109   radius = z + RE;
00110   x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);
00111   x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);
00112   x[2] = radius * sin(lat / 180 * M_PI);
00113 }
00114
00115 /*****************************************************************************/
00116
00117 void get_met(
00118   ctl_t * ctl,
00119   char *metbase,
00120   double t,
00121   met_t * met0,
00122   met_t * met1) {
00123
00124   char filename[LEN];
00125
00126   static int init;
00127
00128   /* Init... */
00129   if (!init) {
00130     init = 1;
00131
00132     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00133     read_met(ctl, filename, met0);
00134
00135     get_met_help(t + 1.0 * ctl->direction, 1, metbase, ctl->
      dt_met, filename);
00136     read_met(ctl, filename, met1);
00137   }
00138
00139   /* Read new data for forward trajectories... */
00140   if (t > met1->time && ctl->direction == 1) {
00141     memcpy(met0, met1, sizeof(met_t));
00142     get_met_help(t, 1, metbase, ctl->dt_met, filename);
00143     read_met(ctl, filename, met1);
00144   }
00145
00146   /* Read new data for backward trajectories... */
00147   if (t < met0->time && ctl->direction == -1) {
00148     memcpy(met1, met0, sizeof(met_t));
00149     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00150     read_met(ctl, filename, met0);
00151   }
00152 }
00153
00154 /*****************************************************************************/
00155
00156 void get_met_help(
00157   double t,
00158   int direct,
00159   char *metbase,
00160   double dt_met,
00161   char *filename) {
00162
00163   double t6, r;
00164
00165   int year, mon, day, hour, min, sec;
00166
00167   /* Round time to fixed intervals... */
00168   if (direct == -1)
00169     t6 = floor(t / dt_met) * dt_met;
00170   else
00171     t6 = ceil(t / dt_met) * dt_met;
00172
00173   /* Decode time... */
00174   jsec2time(t6, &year, &mon, &day, &hour, &min, &sec, &r);
00175
00176   /* Set filename... */
00177   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", metbase, year, mon, day, hour);
00178 }
00179
00180 /*****************************************************************************/
00181
00182 void intpol_met_2d(
00183   double array[EX][EY],
```

```
00184   int ix,
00185   int iy,
00186   double wx,
00187   double wy,
00188   double *var) {
00189
00190   double aux00, aux01, aux10, aux11;
00191
00192   /* Set variables... */
00193   aux00 = array[ix][iy];
00194   aux01 = array[ix][iy + 1];
00195   aux10 = array[ix + 1][iy];
00196   aux11 = array[ix + 1][iy + 1];
00197
00198   /* Interpolate horizontally... */
00199   aux00 = wy * (aux00 - aux01) + aux01;
00200   aux11 = wy * (aux10 - aux11) + aux11;
00201   *var = wx * (aux00 - aux11) + aux11;
00202 }
00203
00204 /*****************************************************************************/
00205
00206 void intpol_met_3d(
00207   float array[EX][EY][EP],
00208   int ip,
00209   int ix,
00210   int iy,
00211   double wp,
00212   double wx,
00213   double wy,
00214   double *var) {
00215
00216   double aux00, aux01, aux10, aux11;
00217
00218   /* Interpolate vertically... */
00219   aux00 = wp * (array[ix][iy][ip] - array[ix][iy][ip + 1])
00220     + array[ix][iy][ip + 1];
00221   aux01 = wp * (array[ix][iy + 1][ip] - array[ix][iy + 1][ip + 1])
00222     + array[ix][iy + 1][ip + 1];
00223   aux10 = wp * (array[ix + 1][iy][ip] - array[ix + 1][iy][ip + 1])
00224     + array[ix + 1][iy][ip + 1];
00225   aux11 = wp * (array[ix + 1][iy + 1][ip] - array[ix + 1][iy + 1][ip + 1])
00226     + array[ix + 1][iy + 1][ip + 1];
00227
00228   /* Interpolate horizontally... */
00229   aux00 = wy * (aux00 - aux01) + aux01;
00230   aux11 = wy * (aux10 - aux11) + aux11;
00231   *var = wx * (aux00 - aux11) + aux11;
00232 }
00233
00234 /*****************************************************************************/
00235
00236 void intpol_met_space(
00237   met_t * met,
00238   double p,
00239   double lon,
00240   double lat,
00241   double *ps,
00242   double *t,
00243   double *u,
00244   double *v,
00245   double *w,
00246   double *h2o,
00247   double *o3) {
00248
00249   double wp, wx, wy;
00250
00251   int ip, ix, iy;
00252
00253   /* Check longitude... */
00254   if (met->lon[met->nx - 1] > 180 && lon < 0)
00255     lon += 360;
00256
00257   /* Get indices... */
00258   ip = locate(met->p, met->np, p);
00259   ix = locate(met->lon, met->nx, lon);
00260   iy = locate(met->lat, met->ny, lat);
00261
00262   /* Get weights... */
00263   wp = (met->p[ip + 1] - p) / (met->p[ip + 1] - met->p[ip]);
00264   wx = (met->lon[ix + 1] - lon) / (met->lon[ix + 1] - met->lon[ix]);
00265   wy = (met->lat[iy + 1] - lat) / (met->lat[iy + 1] - met->lat[iy]);
00266
00267   /* Interpolate... */
00268   if (ps != NULL)
00269     intpol_met_2d(met->ps, ix, iy, wx, wy, ps);
00270   if (t != NULL)
```

```
00271     intpol_met_3d(met->t, ip, ix, iy, wp, wx, wy, t);
00272   if (u != NULL)
00273     intpol_met_3d(met->u, ip, ix, iy, wp, wx, wy, u);
00274   if (v != NULL)
00275     intpol_met_3d(met->v, ip, ix, iy, wp, wx, wy, v);
00276   if (w != NULL)
00277     intpol_met_3d(met->w, ip, ix, iy, wp, wx, wy, w);
00278   if (h2o != NULL)
00279     intpol_met_3d(met->h2o, ip, ix, iy, wp, wx, wy, h2o);
00280   if (o3 != NULL)
00281     intpol_met_3d(met->o3, ip, ix, iy, wp, wx, wy, o3);
00282 }
00283
00284 /*****************************************************************************/
00285
00286 void intpol_met_time(
00287   met_t * met0,
00288   met_t * met1,
00289   double ts,
00290   double p,
00291   double lon,
00292   double lat,
00293   double *ps,
00294   double *t,
00295   double *u,
00296   double *v,
00297   double *w,
00298   double *h2o,
00299   double *o3) {
00300
00301   double h2o0, h2o1, o30, o31, ps0, ps1, t0, t1, u0, u1, v0, v1, w0, w1, wt;
00302
00303   /* Spatial interpolation... */
00304   intpol_met_space(met0, p, lon, lat,
00305                    ps == NULL ? NULL : &ps0,
00306                    t == NULL ? NULL : &t0,
00307                    u == NULL ? NULL : &u0,
00308                    v == NULL ? NULL : &v0,
00309                    w == NULL ? NULL : &w0,
00310                    h2o == NULL ? NULL : &h2o0, o3 == NULL ? NULL : &o30);
00311   intpol_met_space(met1, p, lon, lat,
00312                    ps == NULL ? NULL : &ps1,
00313                    t == NULL ? NULL : &t1,
00314                    u == NULL ? NULL : &u1,
00315                    v == NULL ? NULL : &v1,
00316                    w == NULL ? NULL : &w1,
00317                    h2o == NULL ? NULL : &h2o1, o3 == NULL ? NULL : &o31);
00318
00319   /* Get weighting factor... */
00320   wt = (met1->time - ts) / (met1->time - met0->time);
00321
00322   /* Interpolate... */
00323   if (ps != NULL)
00324     *ps = wt * (ps0 - ps1) + ps1;
00325   if (t != NULL)
00326     *t = wt * (t0 - t1) + t1;
00327   if (u != NULL)
00328     *u = wt * (u0 - u1) + u1;
00329   if (v != NULL)
00330     *v = wt * (v0 - v1) + v1;
00331   if (w != NULL)
00332     *w = wt * (w0 - w1) + w1;
00333   if (h2o != NULL)
00334     *h2o = wt * (h2o0 - h2o1) + h2o1;
00335   if (o3 != NULL)
00336     *o3 = wt * (o30 - o31) + o31;
00337 }
00338
00339 /*****************************************************************************/
00340
00341 void jsec2time(
00342   double jsec,
00343   int *year,
00344   int *mon,
00345   int *day,
00346   int *hour,
00347   int *min,
00348   int *sec,
00349   double *remain) {
00350
00351   struct tm t0, *t1;
00352
00353   time_t jsec0;
00354
00355   t0.tm_year = 100;
00356   t0.tm_mon = 0;
00357   t0.tm_mday = 1;
```

```
00358   t0.tm_hour = 0;
00359   t0.tm_min = 0;
00360   t0.tm_sec = 0;
00361
00362   jsec0 = (time_t) jsec + timegm(&t0);
00363   t1 = gmtime(&jsec0);
00364
00365   *year = t1->tm_year + 1900;
00366   *mon = t1->tm_mon + 1;
00367   *day = t1->tm_mday;
00368   *hour = t1->tm_hour;
00369   *min = t1->tm_min;
00370   *sec = t1->tm_sec;
00371   *remain = jsec - floor(jsec);
00372 }
00373
00374 /*****************************************************************************/
00375
00376 int locate(
00377   double *xx,
00378   int n,
00379   double x) {
00380
00381   int i, ilo, ihi;
00382
00383   ilo = 0;
00384   ihi = n - 1;
00385   i = (ihi + ilo) >> 1;
00386
00387   if (xx[i] < xx[i + 1])
00388     while (ihi > ilo + 1) {
00389       i = (ihi + ilo) >> 1;
00390       if (xx[i] > x)
00391         ihi = i;
00392       else
00393         ilo = i;
00394   } else
00395     while (ihi > ilo + 1) {
00396       i = (ihi + ilo) >> 1;
00397       if (xx[i] <= x)
00398         ihi = i;
00399       else
00400         ilo = i;
00401     }
00402
00403   return ilo;
00404 }
00405
00406 /*****************************************************************************/
00407
00408 void read_atm(
00409   const char *filename,
00410   ctl_t * ctl,
00411   atm_t * atm) {
00412
00413   FILE *in;
00414
00415   char line[LEN], *tok;
00416
00417   int iq;
00418
00419   /* Init... */
00420   atm->np = 0;
00421
00422   /* Write info... */
00423   printf("Read atmospheric data: %s\n", filename);
00424
00425   /* Open file... */
00426   if (!(in = fopen(filename, "r")))
00427     ERRMSG("Cannot open file!");
00428
00429   /* Read line... */
00430   while (fgets(line, LEN, in)) {
00431
00432     /* Read data... */
00433     TOK(line, tok, "%lg", atm->time[atm->np]);
00434     TOK(NULL, tok, "%lg", atm->p[atm->np]);
00435     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
00436     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
00437     for (iq = 0; iq < ctl->nq; iq++)
00438       TOK(NULL, tok, "%lg", atm->q[iq][atm->np]);
00439
00440     /* Convert altitude to pressure... */
00441     atm->p[atm->np] = P(atm->p[atm->np]);
00442
00443     /* Increment data point counter... */
00444     if ((++atm->np) > NP)
```

```
00445       ERRMSG("Too many data points!");
00446   }
00447
00448   /* Close file... */
00449   fclose(in);
00450
00451   /* Check number of points... */
00452   if (atm->np < 1)
00453     ERRMSG("Can not read any data!");
00454 }
00455
00456 /*****************************************************************************/
00457
00458 void read_ctl(
00459   const char *filename,
00460   int argc,
00461   char *argv[],
00462   ctl_t * ctl) {
00463
00464   int ip, iq;
00465
00466   /* Write info... */
00467   printf("\nMassive-Parallel Trajectory Calculations (MPTRAC)\n"
00468          "(executable: %s | compiled: %s, %s)\n\n",
00469          argv[0], __DATE__, __TIME__);
00470
00471   /* Initialize quantity indices... */
00472   ctl->qnt_ens = -1;
00473   ctl->qnt_m = -1;
00474   ctl->qnt_r = -1;
00475   ctl->qnt_rho = -1;
00476   ctl->qnt_ps = -1;
00477   ctl->qnt_p = -1;
00478   ctl->qnt_t = -1;
00479   ctl->qnt_u = -1;
00480   ctl->qnt_v = -1;
00481   ctl->qnt_w = -1;
00482   ctl->qnt_h2o = -1;
00483   ctl->qnt_o3 = -1;
00484   ctl->qnt_theta = -1;
00485   ctl->qnt_pv = -1;
00486   ctl->qnt_tice = -1;
00487   ctl->qnt_tsts = -1;
00488   ctl->qnt_tnat = -1;
00489   ctl->qnt_gw_wind = -1;
00490   ctl->qnt_gw_sso = -1;
00491   ctl->qnt_gw_var = -1;
00492   ctl->qnt_stat = -1;
00493
00494   /* Read quantities... */
00495   ctl->nq = (int) scan_ctl(filename, argc, argv, "NQ", -1, "0", NULL);
00496   if (ctl->nq > NQ)
00497     ERRMSG("Too many quantities!");
00498   for (iq = 0; iq < ctl->nq; iq++) {
00499
00500     /* Read quantity name and format... */
00501     scan_ctl(filename, argc, argv, "QNT_NAME", iq, "", ctl->qnt_name[iq]);
00502     scan_ctl(filename, argc, argv, "QNT_FORMAT", iq, "%g",
00503              ctl->qnt_format[iq]);
00504
00505     /* Try to identify quantity... */
00506     if (strcmp(ctl->qnt_name[iq], "ens") == 0) {
00507       ctl->qnt_ens = iq;
00508       sprintf(ctl->qnt_unit[iq], "-");
00509     } else if (strcmp(ctl->qnt_name[iq], "m") == 0) {
00510       ctl->qnt_m = iq;
00511       sprintf(ctl->qnt_unit[iq], "kg");
00512     } else if (strcmp(ctl->qnt_name[iq], "r") == 0) {
00513       ctl->qnt_r = iq;
00514       sprintf(ctl->qnt_unit[iq], "m");
00515     } else if (strcmp(ctl->qnt_name[iq], "rho") == 0) {
00516       ctl->qnt_rho = iq;
00517       sprintf(ctl->qnt_unit[iq], "kg/m^3");
00518     } else if (strcmp(ctl->qnt_name[iq], "ps") == 0) {
00519       ctl->qnt_ps = iq;
00520       sprintf(ctl->qnt_unit[iq], "hPa");
00521     } else if (strcmp(ctl->qnt_name[iq], "p") == 0) {
00522       ctl->qnt_p = iq;
00523       sprintf(ctl->qnt_unit[iq], "hPa");
00524     } else if (strcmp(ctl->qnt_name[iq], "t") == 0) {
00525       ctl->qnt_t = iq;
00526       sprintf(ctl->qnt_unit[iq], "K");
00527     } else if (strcmp(ctl->qnt_name[iq], "u") == 0) {
00528       ctl->qnt_u = iq;
00529       sprintf(ctl->qnt_unit[iq], "m/s");
00530     } else if (strcmp(ctl->qnt_name[iq], "v") == 0) {
00531       ctl->qnt_v = iq;
```

```
00532       sprintf(ctl->qnt_unit[iq], "m/s");
00533     } else if (strcmp(ctl->qnt_name[iq], "w") == 0) {
00534       ctl->qnt_w = iq;
00535       sprintf(ctl->qnt_unit[iq], "hPa/s");
00536     } else if (strcmp(ctl->qnt_name[iq], "h2o") == 0) {
00537       ctl->qnt_h2o = iq;
00538       sprintf(ctl->qnt_unit[iq], "1");
00539     } else if (strcmp(ctl->qnt_name[iq], "o3") == 0) {
00540       ctl->qnt_o3 = iq;
00541       sprintf(ctl->qnt_unit[iq], "1");
00542     } else if (strcmp(ctl->qnt_name[iq], "theta") == 0) {
00543       ctl->qnt_theta = iq;
00544       sprintf(ctl->qnt_unit[iq], "K");
00545     } else if (strcmp(ctl->qnt_name[iq], "pv") == 0) {
00546       ctl->qnt_pv = iq;
00547       sprintf(ctl->qnt_unit[iq], "PVU");
00548     } else if (strcmp(ctl->qnt_name[iq], "tice") == 0) {
00549       ctl->qnt_tice = iq;
00550       sprintf(ctl->qnt_unit[iq], "K");
00551     } else if (strcmp(ctl->qnt_name[iq], "tsts") == 0) {
00552       ctl->qnt_tsts = iq;
00553       sprintf(ctl->qnt_unit[iq], "K");
00554     } else if (strcmp(ctl->qnt_name[iq], "tnat") == 0) {
00555       ctl->qnt_tnat = iq;
00556       sprintf(ctl->qnt_unit[iq], "K");
00557     } else if (strcmp(ctl->qnt_name[iq], "gw_wind") == 0) {
00558       ctl->qnt_gw_wind = iq;
00559       sprintf(ctl->qnt_unit[iq], "m/s");
00560     } else if (strcmp(ctl->qnt_name[iq], "gw_sso") == 0) {
00561       ctl->qnt_gw_sso = iq;
00562       sprintf(ctl->qnt_unit[iq], "m^2");
00563     } else if (strcmp(ctl->qnt_name[iq], "gw_var") == 0) {
00564       ctl->qnt_gw_var = iq;
00565       sprintf(ctl->qnt_unit[iq], "K^2");
00566     } else if (strcmp(ctl->qnt_name[iq], "stat") == 0) {
00567       ctl->qnt_stat = iq;
00568       sprintf(ctl->qnt_unit[iq], "-");
00569     } else
00570       scan_ctl(filename, argc, argv, "QNT_UNIT", iq, "", ctl->qnt_unit[iq]);
00571   }
00572
00573   /* Time steps of simulation... */
00574   ctl->direction =
00575     (int) scan_ctl(filename, argc, argv, "DIRECTION", -1, "1", NULL);
00576   if (ctl->direction != -1 && ctl->direction != 1)
00577     ERRMSG("Set DIRECTION to -1 or 1!");
00578   ctl->t_start =
00579     scan_ctl(filename, argc, argv, "T_START", -1, "-1e100", NULL);
00580   ctl->t_stop = scan_ctl(filename, argc, argv, "T_STOP", -1, "-1e100", NULL);
00581   ctl->dt_mod = scan_ctl(filename, argc, argv, "DT_MOD", -1, "600", NULL);
00582
00583   /* Meteorological data... */
00584   ctl->dt_met = scan_ctl(filename, argc, argv, "DT_MET", -1, "21600", NULL);
00585   ctl->met_np = (int) scan_ctl(filename, argc, argv, "MET_NP", -1, "0", NULL);
00586   if (ctl->met_np > EP)
00587     ERRMSG("Too many levels!");
00588   for (ip = 0; ip < ctl->met_np; ip++)
00589     ctl->met_p[ip] = scan_ctl(filename, argc, argv, "MET_P", ip, "", NULL);
00590
00591   /* Isosurface parameters... */
00592   ctl->isosurf
00593     = (int) scan_ctl(filename, argc, argv, "ISOSURF", -1, "0", NULL);
00594   scan_ctl(filename, argc, argv, "BALLOON", -1, "-", ctl->balloon);
00595
00596   /* Diffusion parameters... */
00597   ctl->turb_dx_trop
00598     = scan_ctl(filename, argc, argv, "TURB_DX_TROP", -1, "50.0", NULL);
00599   ctl->turb_dx_strat
00600     = scan_ctl(filename, argc, argv, "TURB_DX_STRAT", -1, "0.0", NULL);
00601   ctl->turb_dz_trop
00602     = scan_ctl(filename, argc, argv, "TURB_DZ_TROP", -1, "0.0", NULL);
00603   ctl->turb_dz_strat
00604     = scan_ctl(filename, argc, argv, "TURB_DZ_STRAT", -1, "0.1", NULL);
00605   ctl->turb_meso =
00606     scan_ctl(filename, argc, argv, "TURB_MESO", -1, "0.16", NULL);
00607
00608   /* Life time of particles... */
00609   ctl->tdec_trop = scan_ctl(filename, argc, argv, "TDEC_TROP", -1, "0", NULL);
00610   ctl->tdec_strat =
00611     scan_ctl(filename, argc, argv, "TDEC_STRAT", -1, "0", NULL);
00612
00613   /* PSC analysis... */
00614   ctl->psc_h2o = scan_ctl(filename, argc, argv, "PSC_H2O", -1, "4e-6", NULL);
00615   ctl->psc_hno3 =
00616     scan_ctl(filename, argc, argv, "PSC_HNO3", -1, "9e-9", NULL);
00617
00618   /* Gravity wave analysis... */
```

```
00619   scan_ctl(filename, argc, argv, "GW_BASENAME", -1, "-", ctl->
      gw_basename);
00620
00621   /* Output of atmospheric data... */
00622   scan_ctl(filename, argc, argv, "ATM_BASENAME", -1, "-", ctl->
      atm_basename);
00623   scan_ctl(filename, argc, argv, "ATM_GPFILE", -1, "-", ctl->atm_gpfile);
00624   ctl->atm_dt_out =
00625     scan_ctl(filename, argc, argv, "ATM_DT_OUT", -1, "86400", NULL);
00626   ctl->atm_filter =
00627     (int) scan_ctl(filename, argc, argv, "ATM_FILTER", -1, "0", NULL);
00628
00629   /* Output of CSI data... */
00630   scan_ctl(filename, argc, argv, "CSI_BASENAME", -1, "-", ctl->
      csi_basename);
00631   ctl->csi_dt_out =
00632     scan_ctl(filename, argc, argv, "CSI_DT_OUT", -1, "86400", NULL);
00633   scan_ctl(filename, argc, argv, "CSI_OBSFILE", -1, "obs.tab",
00634            ctl->csi_obsfile);
00635   ctl->csi_obsmin =
00636     scan_ctl(filename, argc, argv, "CSI_OBSMIN", -1, "0", NULL);
00637   ctl->csi_modmin =
00638     scan_ctl(filename, argc, argv, "CSI_MODMIN", -1, "0", NULL);
00639   ctl->csi_z0 = scan_ctl(filename, argc, argv, "CSI_Z0", -1, "0", NULL);
00640   ctl->csi_z1 = scan_ctl(filename, argc, argv, "CSI_Z1", -1, "100", NULL);
00641   ctl->csi_nz = (int) scan_ctl(filename, argc, argv, "CSI_NZ", -1, "1", NULL);
00642   ctl->csi_lon0 =
00643     scan_ctl(filename, argc, argv, "CSI_LON0", -1, "-180", NULL);
00644   ctl->csi_lon1 = scan_ctl(filename, argc, argv, "CSI_LON1", -1, "180", NULL);
00645   ctl->csi_nx =
00646     (int) scan_ctl(filename, argc, argv, "CSI_NX", -1, "360", NULL);
00647   ctl->csi_lat0 = scan_ctl(filename, argc, argv, "CSI_LAT0", -1, "-90", NULL);
00648   ctl->csi_lat1 = scan_ctl(filename, argc, argv, "CSI_LAT1", -1, "90", NULL);
00649   ctl->csi_ny =
00650     (int) scan_ctl(filename, argc, argv, "CSI_NY", -1, "180", NULL);
00651
00652   /* Output of ensemble data... */
00653   scan_ctl(filename, argc, argv, "ENS_BASENAME", -1, "-", ctl->
      ens_basename);
00654
00655   /* Output of grid data... */
00656   scan_ctl(filename, argc, argv, "GRID_BASENAME", -1, "-",
00657            ctl->grid_basename);
00658   scan_ctl(filename, argc, argv, "GRID_GPFILE", -1, "-", ctl->
      grid_gpfile);
00659   ctl->grid_dt_out =
00660     scan_ctl(filename, argc, argv, "GRID_DT_OUT", -1, "86400", NULL);
00661   ctl->grid_sparse =
00662     (int) scan_ctl(filename, argc, argv, "GRID_SPARSE", -1, "0", NULL);
00663   ctl->grid_z0 = scan_ctl(filename, argc, argv, "GRID_Z0", -1, "0", NULL);
00664   ctl->grid_z1 = scan_ctl(filename, argc, argv, "GRID_Z1", -1, "100", NULL);
00665   ctl->grid_nz =
00666     (int) scan_ctl(filename, argc, argv, "GRID_NZ", -1, "1", NULL);
00667   ctl->grid_lon0 =
00668     scan_ctl(filename, argc, argv, "GRID_LON0", -1, "-180", NULL);
00669   ctl->grid_lon1 =
00670     scan_ctl(filename, argc, argv, "GRID_LON1", -1, "180", NULL);
00671   ctl->grid_nx =
00672     (int) scan_ctl(filename, argc, argv, "GRID_NX", -1, "360", NULL);
00673   ctl->grid_lat0 =
00674     scan_ctl(filename, argc, argv, "GRID_LAT0", -1, "-90", NULL);
00675   ctl->grid_lat1 =
00676     scan_ctl(filename, argc, argv, "GRID_LAT1", -1, "90", NULL);
00677   ctl->grid_ny =
00678     (int) scan_ctl(filename, argc, argv, "GRID_NY", -1, "180", NULL);
00679
00680   /* Output of profile data... */
00681   scan_ctl(filename, argc, argv, "PROF_BASENAME", -1, "-",
00682            ctl->prof_basename);
00683   scan_ctl(filename, argc, argv, "PROF_OBSFILE", -1, "-", ctl->
      prof_obsfile);
00684   ctl->prof_z0 = scan_ctl(filename, argc, argv, "PROF_Z0", -1, "0", NULL);
00685   ctl->prof_z1 = scan_ctl(filename, argc, argv, "PROF_Z1", -1, "60", NULL);
00686   ctl->prof_nz =
00687     (int) scan_ctl(filename, argc, argv, "PROF_NZ", -1, "60", NULL);
00688   ctl->prof_lon0 =
00689     scan_ctl(filename, argc, argv, "PROF_LON0", -1, "-180", NULL);
00690   ctl->prof_lon1 =
00691     scan_ctl(filename, argc, argv, "PROF_LON1", -1, "180", NULL);
00692   ctl->prof_nx =
00693     (int) scan_ctl(filename, argc, argv, "PROF_NX", -1, "360", NULL);
00694   ctl->prof_lat0 =
00695     scan_ctl(filename, argc, argv, "PROF_LAT0", -1, "-90", NULL);
00696   ctl->prof_lat1 =
00697     scan_ctl(filename, argc, argv, "PROF_LAT1", -1, "90", NULL);
00698   ctl->prof_ny =
00699     (int) scan_ctl(filename, argc, argv, "PROF_NY", -1, "180", NULL);
```

```
00700
00701   /* Output of station data... */
00702   scan_ctl(filename, argc, argv, "STAT_BASENAME", -1, "-",
00703            ctl->stat_basename);
00704   ctl->stat_lon = scan_ctl(filename, argc, argv, "STAT_LON", -1, "0", NULL);
00705   ctl->stat_lat = scan_ctl(filename, argc, argv, "STAT_LAT", -1, "0", NULL);
00706   ctl->stat_r = scan_ctl(filename, argc, argv, "STAT_R", -1, "50", NULL);
00707 }
00708
00709 /*****************************************************************************/
00710
00711 void read_met(
00712   ctl_t * ctl,
00713   char *filename,
00714   met_t * met) {
00715
00716   char tstr[10];
00717
00718   static float help[EX * EY];
00719
00720   int ix, iy, ip, dimid, ncid, varid, year, mon, day, hour;
00721
00722   size_t np, nx, ny;
00723
00724   /* Write info... */
00725   printf("Read meteorological data: %s\n", filename);
00726
00727   /* Get time from filename... */
00728   sprintf(tstr, "%.4s", &filename[strlen(filename) - 16]);
00729   year = atoi(tstr);
00730   sprintf(tstr, "%.2s", &filename[strlen(filename) - 11]);
00731   mon = atoi(tstr);
00732   sprintf(tstr, "%.2s", &filename[strlen(filename) - 8]);
00733   day = atoi(tstr);
00734   sprintf(tstr, "%.2s", &filename[strlen(filename) - 5]);
00735   hour = atoi(tstr);
00736   time2jsec(year, mon, day, hour, 0, 0, 0, &met->time);
00737
00738   /* Open netCDF file... */
00739   NC(nc_open(filename, NC_NOWRITE, &ncid));
00740
00741   /* Get dimensions... */
00742   NC(nc_inq_dimid(ncid, "lon", &dimid));
00743   NC(nc_inq_dimlen(ncid, dimid, &nx));
00744   if (nx > EX)
00745     ERRMSG("Too many longitudes!");
00746
00747   NC(nc_inq_dimid(ncid, "lat", &dimid));
00748   NC(nc_inq_dimlen(ncid, dimid, &ny));
00749   if (ny > EY)
00750     ERRMSG("Too many latitudes!");
00751
00752   NC(nc_inq_dimid(ncid, "lev", &dimid));
00753   NC(nc_inq_dimlen(ncid, dimid, &np));
00754   if (np > EP)
00755     ERRMSG("Too many levels!");
00756
00757   /* Store dimensions... */
00758   met->np = (int) np;
00759   met->nx = (int) nx;
00760   met->ny = (int) ny;
00761
00762   /* Get horizontal grid... */
00763   NC(nc_inq_varid(ncid, "lon", &varid));
00764   NC(nc_get_var_double(ncid, varid, met->lon));
00765   NC(nc_inq_varid(ncid, "lat", &varid));
00766   NC(nc_get_var_double(ncid, varid, met->lat));
00767
00768   /* Read meteorological data... */
00769   read_met_help(ncid, "t", "T", met, met->t, 1.0);
00770   read_met_help(ncid, "u", "U", met, met->u, 1.0);
00771   read_met_help(ncid, "v", "V", met, met->v, 1.0);
00772   read_met_help(ncid, "w", "W", met, met->w, 0.01f);
00773   read_met_help(ncid, "q", "Q", met, met->h2o, 1.608f);
00774   read_met_help(ncid, "o3", "O3", met, met->o3, 0.602f);
00775
00776   /* Meteo data on pressure levels... */
00777   if (ctl->met_np <= 0) {
00778
00779     /* Read pressure levels from file... */
00780     NC(nc_inq_varid(ncid, "lev", &varid));
00781     NC(nc_get_var_double(ncid, varid, met->p));
00782     for (ip = 0; ip < met->np; ip++)
00783       met->p[ip] /= 100.;
00784
00785     /* Extrapolate data for lower boundary... */
00786     read_met_extrapolate(met);
```

```
00787   }
00788
00789   /* Meteo data on model levels... */
00790   else {
00791
00792     /* Read pressure data from file... */
00793     read_met_help(ncid, "pl", "PL", met, met->pl, 0.01f);
00794
00795     /* Interpolate from model levels to pressure levels... */
00796     read_met_ml2pl(ctl, met, met->t);
00797     read_met_ml2pl(ctl, met, met->u);
00798     read_met_ml2pl(ctl, met, met->v);
00799     read_met_ml2pl(ctl, met, met->w);
00800     read_met_ml2pl(ctl, met, met->h2o);
00801     read_met_ml2pl(ctl, met, met->o3);
00802
00803     /* Set pressure levels... */
00804     met->np = ctl->met_np;
00805     for (ip = 0; ip < met->np; ip++)
00806       met->p[ip] = ctl->met_p[ip];
00807   }
00808
00809   /* Check ordering of pressure levels... */
00810   for (ip = 1; ip < met->np; ip++)
00811     if (met->p[ip - 1] < met->p[ip])
00812       ERRMSG("Pressure levels must be descending!");
00813
00814   /* Read surface pressure... */
00815   if (nc_inq_varid(ncid, "PS", &varid) == NC_NOERR) {
00816     NC(nc_get_var_float(ncid, varid, help));
00817     for (iy = 0; iy < met->ny; iy++)
00818       for (ix = 0; ix < met->nx; ix++)
00819         met->ps[ix][iy] = help[iy * met->nx + ix] / 100.;
00820   } else if (nc_inq_varid(ncid, "LNSP", &varid) == NC_NOERR) {
00821     NC(nc_get_var_float(ncid, varid, help));
00822     for (iy = 0; iy < met->ny; iy++)
00823       for (ix = 0; ix < met->nx; ix++)
00824         met->ps[ix][iy] = exp(help[iy * met->nx + ix]) / 100.;
00825   } else
00826     for (ix = 0; ix < met->nx; ix++)
00827       for (iy = 0; iy < met->ny; iy++)
00828         met->ps[ix][iy] = met->p[0];
00829
00830   /* Create periodic boundary conditions... */
00831   read_met_periodic(met);
00832
00833   /* Close file... */
00834   NC(nc_close(ncid));
00835 }
00836
00837 /*****************************************************************************/
00838
00839 void read_met_extrapolate(
00840   met_t * met) {
00841
00842   int ip, ip0, ix, iy;
00843
00844   /* Loop over columns... */
00845   for (ix = 0; ix < met->nx; ix++)
00846     for (iy = 0; iy < met->ny; iy++) {
00847
00848       /* Find lowest valid data point... */
00849       for (ip0 = met->np - 1; ip0 >= 0; ip0--)
00850         if (!gsl_finite(met->t[ix][iy][ip0])
00851             || !gsl_finite(met->u[ix][iy][ip0])
00852             || !gsl_finite(met->v[ix][iy][ip0])
00853             || !gsl_finite(met->w[ix][iy][ip0]))
00854           break;
00855
00856       /* Extrapolate... */
00857       for (ip = ip0; ip >= 0; ip--) {
00858         met->t[ix][iy][ip] = met->t[ix][iy][ip + 1];
00859         met->u[ix][iy][ip] = met->u[ix][iy][ip + 1];
00860         met->v[ix][iy][ip] = met->v[ix][iy][ip + 1];
00861         met->w[ix][iy][ip] = met->w[ix][iy][ip + 1];
00862         met->h2o[ix][iy][ip] = met->h2o[ix][iy][ip + 1];
00863         met->o3[ix][iy][ip] = met->o3[ix][iy][ip + 1];
00864       }
00865     }
00866 }
00867
00868 /*****************************************************************************/
00869
00870 void read_met_help(
00871   int ncid,
00872   char *varname,
00873   char *varname2,
```

```
00874   met_t * met,
00875   float dest[EX][EY][EP],
00876   float scl) {
00877
00878   static float help[EX * EY * EP];
00879
00880   int ip, ix, iy, n = 0, varid;
00881
00882   /* Check if variable exists... */
00883   if (nc_inq_varid(ncid, varname, &varid) != NC_NOERR)
00884     if (nc_inq_varid(ncid, varname2, &varid) != NC_NOERR)
00885       return;
00886
00887   /* Read data... */
00888   NC(nc_get_var_float(ncid, varid, help));
00889
00890   /* Copy and check data... */
00891   for (ip = 0; ip < met->np; ip++)
00892     for (iy = 0; iy < met->ny; iy++)
00893       for (ix = 0; ix < met->nx; ix++) {
00894         dest[ix][iy][ip] = scl * help[n++];
00895         if (fabs(dest[ix][iy][ip] / scl) > 1e14)
00896           dest[ix][iy][ip] = GSL_NAN;
00897       }
00898 }
00899
00900 /*****************************************************************************/
00901
00902 void read_met_ml2pl(
00903   ctl_t * ctl,
00904   met_t * met,
00905   float var[EX][EY][EP]) {
00906
00907   double aux[EP], p[EP], pt;
00908
00909   int ip, ip2, ix, iy;
00910
00911   /* Loop over columns... */
00912   for (ix = 0; ix < met->nx; ix++)
00913     for (iy = 0; iy < met->ny; iy++) {
00914
00915       /* Copy pressure profile... */
00916       for (ip = 0; ip < met->np; ip++)
00917         p[ip] = met->pl[ix][iy][ip];
00918
00919       /* Interpolate... */
00920       for (ip = 0; ip < ctl->met_np; ip++) {
00921         pt = ctl->met_p[ip];
00922         if ((pt > p[0] && p[0] > p[1]) || (pt < p[0] && p[0] < p[1]))
00923           pt = p[0];
00924         else if ((pt > p[met->np - 1] && p[1] > p[0])
00925                  || (pt < p[met->np - 1] && p[1] < p[0]))
00926           pt = p[met->np - 1];
00927         ip2 = locate(p, met->np, pt);
00928         aux[ip] = LIN(p[ip2], var[ix][iy][ip2],
00929                       p[ip2 + 1], var[ix][iy][ip2 + 1], pt);
00930       }
00931
00932       /* Copy data... */
00933       for (ip = 0; ip < ctl->met_np; ip++)
00934         var[ix][iy][ip] = (float) aux[ip];
00935     }
00936 }
00937
00938 /*****************************************************************************/
00939
00940 void read_met_periodic(
00941   met_t * met) {
00942
00943   int ip, iy;
00944
00945   /* Check longitudes... */
00946   if (!(fabs(met->lon[met->nx - 1] - met->lon[0]
00947              + met->lon[1] - met->lon[0] - 360) < 0.01))
00948     return;
00949
00950   /* Increase longitude counter... */
00951   if ((++met->nx) > EX)
00952     ERRMSG("Cannot create periodic boundary conditions!");
00953
00954   /* Set longitude... */
00955   met->lon[met->nx - 1] = met->lon[met->nx - 2] + met->lon[1] - met->
      lon[0];
00956
00957   /* Loop over latitudes and pressure levels... */
00958   for (iy = 0; iy < met->ny; iy++)
00959     for (ip = 0; ip < met->np; ip++) {
```

```
00960       met->ps[met->nx - 1][iy] = met->ps[0][iy];
00961       met->u[met->nx - 1][iy][ip] = met->u[0][iy][ip];
00962       met->v[met->nx - 1][iy][ip] = met->v[0][iy][ip];
00963       met->w[met->nx - 1][iy][ip] = met->w[0][iy][ip];
00964       met->t[met->nx - 1][iy][ip] = met->t[0][iy][ip];
00965       met->h2o[met->nx - 1][iy][ip] = met->h2o[0][iy][ip];
00966       met->o3[met->nx - 1][iy][ip] = met->o3[0][iy][ip];
00967     }
00968 }
00969
00970 /*****************************************************************************/
00971
00972 double scan_ctl(
00973   const char *filename,
00974   int argc,
00975   char *argv[],
00976   const char *varname,
00977   int arridx,
00978   const char *defvalue,
00979   char *value) {
00980
00981   FILE *in = NULL;
00982
00983   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
00984     msg[LEN], rvarname[LEN], rval[LEN];
00985
00986   int contain = 0, i;
00987
00988   /* Open file... */
00989   if (filename[strlen(filename) - 1] != '-')
00990     if (!(in = fopen(filename, "r")))
00991       ERRMSG("Cannot open file!");
00992
00993   /* Set full variable name... */
00994   if (arridx >= 0) {
00995     sprintf(fullname1, "%s[%d]", varname, arridx);
00996     sprintf(fullname2, "%s[*]", varname);
00997   } else {
00998     sprintf(fullname1, "%s", varname);
00999     sprintf(fullname2, "%s", varname);
01000   }
01001
01002   /* Read data... */
01003   if (in != NULL)
01004     while (fgets(line, LEN, in))
01005       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
01006         if (strcasecmp(rvarname, fullname1) == 0 ||
01007             strcasecmp(rvarname, fullname2) == 0) {
01008           contain = 1;
01009           break;
01010         }
01011   for (i = 1; i < argc - 1; i++)
01012     if (strcasecmp(argv[i], fullname1) == 0 ||
01013         strcasecmp(argv[i], fullname2) == 0) {
01014       sprintf(rval, "%s", argv[i + 1]);
01015       contain = 1;
01016       break;
01017     }
01018
01019   /* Close file... */
01020   if (in != NULL)
01021     fclose(in);
01022
01023   /* Check for missing variables... */
01024   if (!contain) {
01025     if (strlen(defvalue) > 0)
01026       sprintf(rval, "%s", defvalue);
01027     else {
01028       sprintf(msg, "Missing variable %s!\n", fullname1);
01029       ERRMSG(msg);
01030     }
01031   }
01032
01033   /* Write info... */
01034   printf("%s = %s\n", fullname1, rval);
01035
01036   /* Return values... */
01037   if (value != NULL)
01038     sprintf(value, "%s", rval);
01039   return atof(rval);
01040 }
01041
01042 /*****************************************************************************/
01043
01044 void time2jsec(
01045   int year,
01046   int mon,
```

```
01047   int day,
01048   int hour,
01049   int min,
01050   int sec,
01051   double remain,
01052   double *jsec) {
01053
01054   struct tm t0, t1;
01055
01056   t0.tm_year = 100;
01057   t0.tm_mon = 0;
01058   t0.tm_mday = 1;
01059   t0.tm_hour = 0;
01060   t0.tm_min = 0;
01061   t0.tm_sec = 0;
01062
01063   t1.tm_year = year - 1900;
01064   t1.tm_mon = mon - 1;
01065   t1.tm_mday = day;
01066   t1.tm_hour = hour;
01067   t1.tm_min = min;
01068   t1.tm_sec = sec;
01069
01070   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
01071 }
01072
01073 /*****************************************************************************/
01074
01075 void timer(
01076   const char *name,
01077   int id,
01078   int mode) {
01079
01080   static double starttime[NTIMER], runtime[NTIMER];
01081
01082   /* Check id... */
01083   if (id < 0 || id >= NTIMER)
01084     ERRMSG("Too many timers!");
01085
01086   /* Start timer... */
01087   if (mode == 1) {
01088     if (starttime[id] <= 0)
01089       starttime[id] = omp_get_wtime();
01090     else
01091       ERRMSG("Timer already started!");
01092   }
01093
01094   /* Stop timer... */
01095   else if (mode == 2) {
01096     if (starttime[id] > 0) {
01097       runtime[id] = runtime[id] + omp_get_wtime() - starttime[id];
01098       starttime[id] = -1;
01099     } else
01100       ERRMSG("Timer not started!");
01101   }
01102
01103   /* Print timer... */
01104   else if (mode == 3)
01105     printf("%s = %g s\n", name, runtime[id]);
01106 }
01107
01108 /*****************************************************************************/
01109
01110 double tropopause(
01111   double t,
01112   double lat) {
01113
01114   static double doys[12]
01115   = { 1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 };
01116
01117   static double lats[73]
01118     = { -90, -87.5, -85, -82.5, -80, -77.5, -75, -72.5, -70, -67.5,
01119     -65, -62.5, -60, -57.5, -55, -52.5, -50, -47.5, -45, -42.5,
01120     -40, -37.5, -35, -32.5, -30, -27.5, -25, -22.5, -20, -17.5,
01121     -15, -12.5, -10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5, 10, 12.5,
01122     15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40, 42.5,
01123     45, 47.5, 50, 52.5, 55, 57.5, 60, 62.5, 65, 67.5, 70, 72.5,
01124     75, 77.5, 80, 82.5, 85, 87.5, 90
01125   };
01126
01127   static double tps[12][73]
01128     = { {324.1, 325.6, 325, 324.3, 322.5, 319.7, 314, 307.2, 301.8, 299.6,
01129          297.1, 292.2, 285.6, 276.1, 264, 248.9, 231.9, 213.5, 194.4,
01130          175.3, 157, 140.4, 126.7, 116.3, 109.5, 105.4, 103, 101.4, 100.4,
01131          99.69, 99.19, 98.84, 98.56, 98.39, 98.39, 98.42, 98.44, 98.54,
01132          98.68, 98.81, 98.89, 98.96, 99.12, 99.65, 101.4, 105.4, 113.5, 128,
01133          152.1, 184.7, 214, 234.1, 247.3, 255.8, 262.6, 267.7, 271.7, 275,
```

```
01134          277.2, 279, 280.1, 280.4, 280.6, 280.1, 279.3, 278.3, 276.8, 275.8,
01135          275.3, 275.6, 275.4, 274.1, 273.5},
01136    {337.3, 338.7, 337.8, 336.4, 333, 328.8, 321.1, 312.6, 306.6, 303.7,
01137     300.2, 293.8, 285.4, 273.8, 259.6, 242.7, 224.4, 205.2, 186, 167.5,
01138     150.3, 135, 122.8, 113.9, 108.2, 104.7, 102.5, 101.1, 100.2, 99.42,
01139     98.88, 98.52, 98.25, 98.09, 98.07, 98.1, 98.12, 98.2, 98.25, 98.27,
01140     98.26, 98.27, 98.36, 98.79, 100.2, 104.2, 113.7, 131.2, 159.5, 193,
01141     220.4, 238.1, 250.2, 258.1, 264.7, 269.7, 273.7, 277.3, 280.2, 282.8,
01142     284.9, 286.5, 288.1, 288.8, 289, 288.5, 287.2, 286.3, 286.1, 287.2,
01143     287.5, 286.2, 285.8},
01144    {335, 336, 335.7, 335.1, 332.3, 328.1, 320.6, 311.8, 305.1, 301.9,
01145     297.6, 290, 280.4, 268.3, 254.6, 239.6, 223.9, 207.9, 192.2, 176.9,
01146     161.7, 146.4, 132.2, 120.6, 112.3, 107.2, 104.3, 102.4, 101.3,
01147     100.4, 99.86, 99.47, 99.16, 98.97, 98.94, 98.97, 99, 99.09, 99.2,
01148     99.31, 99.35, 99.41, 99.51, 99.86, 101.1, 104.9, 114.3, 131, 156.8,
01149     186.3, 209.3, 224.6, 236.8, 246.3, 254.9, 262.3, 268.8, 274.8,
01150     279.9, 284.6, 288.6, 291.6, 294.9, 297.5, 299.8, 301.8, 303.1,
01151     304.3, 304.9, 306, 306.6, 306.2, 306},
01152    {306.2, 306.7, 305.7, 307.1, 307.3, 306.4, 301.8, 296.2, 292.4,
01153     290.3, 287.1, 280.9, 273.4, 264.3, 254.1, 242.8, 231, 219, 207.2,
01154     195.5, 183.3, 169.7, 154.7, 138.7, 124.1, 113.6, 107.8, 104.7,
01155     102.8, 101.7, 100.9, 100.4, 100, 99.79, 99.7, 99.66, 99.68, 99.79,
01156     99.94, 100.2, 100.5, 100.9, 101.4, 102.1, 103.4, 107, 115.2, 129.1,
01157     148.7, 171, 190.8, 205.6, 218.4, 229.4, 239.6, 248.6, 256.5,
01158     263.7, 270.3, 276.6, 282.6, 288.1, 294.5, 300.4, 306.3, 311.4,
01159     315.1, 318.3, 320.3, 322.2, 322.8, 321.5, 321.1},
01160    {266.5, 264.9, 260.8, 261, 262, 263, 261.3, 259.7, 259.2, 259.8,
01161     260.1, 258.6, 256.7, 253.6, 249.5, 243.9, 237.4, 230, 222.1, 213.9,
01162     205, 194.4, 180.4, 161.8, 140.7, 122.9, 112.1, 106.7, 104.1, 102.7,
01163     101.8, 101.4, 101.1, 101, 101, 101, 101.1, 101.2, 101.5, 101.9,
01164     102.4, 103, 103.8, 104.9, 106.8, 110.1, 115.6, 124, 135.2, 148.9,
01165     165.2, 181.3, 198, 211.8, 223.5, 233.8, 242.9, 251.5, 259, 266.2,
01166     273.1, 279.2, 286.2, 292.8, 299.6, 306, 311.1, 315.5, 318.8, 322.6,
01167     325.3, 325.8, 325.8},
01168    {220.1, 218.1, 210.8, 207.2, 207.6, 210.5, 211.4, 213.5, 217.3,
01169     222.4, 227.9, 232.8, 237.4, 240.8, 242.8, 243, 241.5, 238.6, 234.2,
01170     228.5, 221, 210.7, 195.1, 172.9, 147.8, 127.6, 115.6, 109.9, 107.1,
01171     105.7, 105, 104.8, 104.8, 104.9, 105, 105.1, 105.3, 105.5, 105.8,
01172     106.4, 107, 107.6, 108.1, 108.8, 110, 111.8, 114.2, 117.4, 121.6,
01173     127.9, 137.3, 151.2, 169.5, 189, 205.8, 218.9, 229.1, 237.8, 245,
01174     251.5, 257.1, 262.3, 268.2, 274, 280.4, 286.7, 292.4, 297.9, 302.9,
01175     308.5, 312.2, 313.1, 313.3},
01176    {187.4, 184.5, 173.3, 166.1, 165.4, 167.8, 169.6, 173.6, 179.6,
01177     187.9, 198.9, 210, 220.5, 229.2, 235.7, 239.9, 241.8, 241.6, 239.6,
01178     235.8, 229.4, 218.6, 200.9, 175.9, 149.4, 129.4, 118.3, 113.1,
01179     110.8, 109.7, 109.3, 109.4, 109.7, 110, 110.2, 110.4, 110.5, 110.7,
01180     111, 111.4, 111.8, 112.1, 112.3, 112.7, 113.2, 113.9, 115, 116.4,
01181     117.9, 120.4, 124.1, 130.9, 142.2, 159.6, 179.6, 198.5, 212.9,
01182     224.2, 232.7, 239.1, 243.8, 247.7, 252.4, 257.3, 263.2, 269.5,
01183     275.4, 281.1, 286.3, 292, 296.3, 298.2, 298.8},
01184    {166, 166.4, 155.7, 148.3, 147.1, 149, 152.1, 157, 163.6, 172.4,
01185     185.3, 199.2, 212.6, 224, 233.2, 239.6, 243.3, 244.6, 243.6, 240.3,
01186     233.9, 222.6, 203.7, 177, 149.5, 129.7, 119, 114, 111.7, 110.7,
01187     110.3, 110.3, 110.6, 110.9, 111.1, 111.3, 111.5, 111.6, 111.9,
01188     112.2, 112.5, 112.6, 112.8, 113, 113.4, 114, 115.1, 116.5, 118.3,
01189     120.9, 124.4, 130.2, 139.4, 154.6, 173.8, 193.1, 208.1, 220.4,
01190     230.1, 238.2, 244.7, 249.5, 254.5, 259.3, 264.5, 269.4, 273.7,
01191     278.2, 282.6, 287.4, 290.9, 292.5, 293},
01192    {171.9, 172.8, 166.2, 162.3, 161.4, 162.5, 165.2, 169.6, 175.3,
01193     183.1, 193.8, 205.9, 218.3, 229.6, 238.5, 244.3, 246.9, 246.7,
01194     243.8, 238.4, 230.2, 217.9, 199.6, 174.9, 148.9, 129.8, 119.5,
01195     114.8, 112.3, 110.9, 110.3, 110.1, 110.2, 110.3, 110.4, 110.5,
01196     110.6, 110.8, 111, 111.4, 111.8, 112, 112.2, 112.4, 112.9, 113.6,
01197     114.7, 116.3, 118.4, 121.9, 127.1, 136.1, 149.8, 168.4, 186.9,
01198     203.3, 217, 229.1, 238.7, 247, 254, 259.3, 264.3, 268.3, 272.5,
01199     276.6, 280.4, 284.4, 288.4, 293.3, 297.2, 298.7, 299.1},
01200    {191.6, 192.2, 189, 188.1, 190.2, 193.7, 197.8, 202.9, 208.5,
01201     215.6, 224.2, 233.1, 241.2, 247.3, 250.8, 251.3, 248.9, 244.2,
01202     237.3, 228.4, 217.2, 202.9, 184.5, 162.5, 140.7, 124.8, 116.2,
01203     111.8, 109.4, 107.9, 107, 106.7, 106.6, 106.6, 106.7, 106.7,
01204     106.8, 107, 107.4, 108, 108.7, 109.3, 109.8, 110.4, 111.2,
01205     112.4, 114.2, 116.9, 121.1, 127.9, 139.3, 155.2, 173.6, 190.7,
01206     206.1, 220.1, 232.3, 243, 251.8, 259.2, 265.7, 270.6, 275.3,
01207     279.3, 283.3, 286.9, 289.7, 292.8, 296.1, 300.5, 303.9, 304.8,
01208     305.1},
01209    {241.5, 239.6, 236.8, 237.4, 239.4, 242.3, 244.2, 246.4, 249.2,
01210     253.6, 258.6, 262.7, 264.8, 264.2, 260.6, 254.1, 245.5, 235.3,
01211     223.9, 211.7, 198.3, 183.1, 165.6, 147.1, 130.5, 118.7, 111.9,
01212     108.1, 105.8, 104.3, 103.4, 102.8, 102.5, 102.4, 102.5, 102.5,
01213     102.5, 102.7, 103.1, 103.8, 104.6, 105.4, 106.1, 107, 108.2,
01214     109.9, 112.8, 117.5, 126, 140.4, 161, 181.9, 201.2, 216.8, 230.4,
01215     241.8, 251.4, 259.9, 266.9, 272.8, 277.4, 280.4, 282.9, 284.6,
01216     286.1, 287.4, 288.3, 289.5, 290.9, 294.2, 296.9, 297.5, 297.6},
01217    {301.2, 300.3, 296.6, 295.4, 295, 294.3, 291.2, 287.4, 284.9, 284.7,
01218     284.1, 281.5, 277.1, 270.4, 261.7, 250.6, 237.6, 223.1, 207.9, 192,
01219     175.8, 158.8, 142.1, 127.6, 116.8, 109.9, 106, 103.6, 102.1, 101.1,
01220     100.4, 99.96, 99.6, 99.37, 99.32, 99.32, 99.31, 99.46, 99.77, 100.2,
```

```
01221    100.7, 101.3, 101.8, 102.7, 104.1, 106.8, 111.9, 121, 136.7, 160,
01222    186.9, 209.9, 228.1, 241.2, 251.5, 259.5, 265.7, 270.9, 274.8, 278,
01223    280.3, 281.8, 283, 283.3, 283.7, 283.8, 283, 282.2, 281.2, 281.4,
01224    281.7, 281.1, 281.2};
01225   };
01226
01227   double doy, p0, p1, pt;
01228
01229   int imon, ilat;
01230
01231   /* Get day of year... */
01232   doy = fmod(t / 86400., 365.25);
01233   while (doy < 0)
01234     doy += 365.25;
01235
01236   /* Get indices... */
01237   imon = locate(doys, 12, doy);
01238   ilat = locate(lats, 73, lat);
01239
01240   /* Get tropopause pressure... */
01241   p0 = LIN(lats[ilat], tps[imon][ilat],
01242            lats[ilat + 1], tps[imon][ilat + 1], lat);
01243   p1 = LIN(lats[ilat], tps[imon + 1][ilat],
01244            lats[ilat + 1], tps[imon + 1][ilat + 1], lat);
01245   pt = LIN(doys[imon], p0, doys[imon + 1], p1, doy);
01246
01247   /* Return tropopause pressure... */
01248   return pt;
01249 }
01250
01251 /*****************************************************************************/
01252
01253 void write_atm(
01254   const char *filename,
01255   ctl_t * ctl,
01256   atm_t * atm,
01257   double t) {
01258
01259   FILE *in, *out;
01260
01261   char line[LEN];
01262
01263   double r, t0, t1;
01264
01265   int ip, iq, year, mon, day, hour, min, sec;
01266
01267   /* Set time interval for output... */
01268   t0 = t - 0.5 * ctl->dt_mod;
01269   t1 = t + 0.5 * ctl->dt_mod;
01270
01271   /* Check if gnuplot output is requested... */
01272   if (ctl->atm_gpfile[0] != '-') {
01273
01274     /* Write info... */
01275     printf("Plot atmospheric data: %s.png\n", filename);
01276
01277     /* Create gnuplot pipe... */
01278     if (!(out = popen("gnuplot", "w")))
01279       ERRMSG("Cannot create pipe to gnuplot!");
01280
01281     /* Set plot filename... */
01282     fprintf(out, "set out \"%s.png\"\n", filename);
01283
01284     /* Set time string... */
01285     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01286     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01287             year, mon, day, hour, min);
01288
01289     /* Dump gnuplot file to pipe... */
01290     if (!(in = fopen(ctl->atm_gpfile, "r")))
01291       ERRMSG("Cannot open file!");
01292     while (fgets(line, LEN, in))
01293       fprintf(out, "%s", line);
01294     fclose(in);
01295   }
01296
01297   else {
01298
01299     /* Write info... */
01300     printf("Write atmospheric data: %s\n", filename);
01301
01302     /* Create file... */
01303     if (!(out = fopen(filename, "w")))
01304       ERRMSG("Cannot create file!");
01305   }
01306
01307   /* Write header... */
```

```
01308   fprintf(out,
01309           "# $1 = time [s]\n"
01310           "# $2 = altitude [km]\n"
01311           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01312   for (iq = 0; iq < ctl->nq; iq++)
01313     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl->qnt_name[iq],
01314             ctl->qnt_unit[iq]);
01315   fprintf(out, "\n");
01316
01317   /* Write data... */
01318   for (ip = 0; ip < atm->np; ip++) {
01319
01320     /* Check time... */
01321     if (ctl->atm_filter && (atm->time[ip] < t0 || atm->time[ip] > t1))
01322       continue;
01323
01324     /* Write output... */
01325     fprintf(out, "%.2f %g %g %g", atm->time[ip], Z(atm->p[ip]),
01326             atm->lon[ip], atm->lat[ip]);
01327     for (iq = 0; iq < ctl->nq; iq++) {
01328       fprintf(out, " ");
01329       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
01330     }
01331     fprintf(out, "\n");
01332   }
01333
01334   /* Close file... */
01335   fclose(out);
01336 }
01337
01338 /*****************************************************************************/
01339
01340 void write_csi(
01341   const char *filename,
01342   ctl_t * ctl,
01343   atm_t * atm,
01344   double t) {
01345
01346   static FILE *in, *out;
01347
01348   static char line[LEN];
01349
01350   static double modmean[GX][GY][GZ], obsmean[GX][GY][GZ],
01351     rt, rz, rlon, rlat, robs, t0, t1, area, dlon, dlat, lat;
01352
01353   static int init, obscount[GX][GY][GZ], cx, cy, cz, ip, ix, iy, iz;
01354
01355   /* Init... */
01356   if (!init) {
01357     init = 1;
01358
01359     /* Check quantity index for mass... */
01360     if (ctl->qnt_m < 0)
01361       ERRMSG("Need quantity mass to analyze CSI!");
01362
01363     /* Open observation data file... */
01364     printf("Read CSI observation data: %s\n", ctl->csi_obsfile);
01365     if (!(in = fopen(ctl->csi_obsfile, "r")))
01366       ERRMSG("Cannot open file!");
01367
01368     /* Create new file... */
01369     printf("Write CSI data: %s\n", filename);
01370     if (!(out = fopen(filename, "w")))
01371       ERRMSG("Cannot create file!");
01372
01373     /* Write header... */
01374     fprintf(out,
01375             "# $1 = time [s]\n"
01376             "# $2 = number of hits (cx)\n"
01377             "# $3 = number of misses (cy)\n"
01378             "# $4 = number of false alarms (cz)\n"
01379             "# $5 = number of observations (cx + cy)\n"
01380             "# $6 = number of forecasts (cx + cz)\n"
01381             "# $7 = bias (forecasts/observations) [%%]\n"
01382             "# $8 = probability of detection (POD) [%%]\n"
01383             "# $9 = false alarm rate (FAR) [%%]\n"
01384             "# $10 = critical success index (CSI) [%%]\n\n");
01385   }
01386
01387   /* Set time interval... */
01388   t0 = t - 0.5 * ctl->dt_mod;
01389   t1 = t + 0.5 * ctl->dt_mod;
01390
01391   /* Initialize grid cells... */
01392   for (ix = 0; ix < ctl->csi_nx; ix++)
01393     for (iy = 0; iy < ctl->csi_ny; iy++)
01394       for (iz = 0; iz < ctl->csi_nz; iz++)
```

```
01395         modmean[ix][iy][iz] = obsmean[ix][iy][iz] = obscount[ix][iy][iz] = 0;
01396
01397     /* Read data... */
01398     while (fgets(line, LEN, in)) {
01399
01400       /* Read data... */
01401       if (sscanf(line, "%lg %lg %lg %lg %lg", &rt, &rz, &rlon, &rlat, &robs) !=
01402           5)
01403         continue;
01404
01405       /* Check time... */
01406       if (rt < t0)
01407         continue;
01408       if (rt > t1)
01409         break;
01410
01411       /* Calculate indices... */
01412       ix = (int) ((rlon - ctl->csi_lon0)
01413                   / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01414       iy = (int) ((rlat - ctl->csi_lat0)
01415                   / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01416       iz = (int) ((rz - ctl->csi_z0)
01417                   / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01418
01419       /* Check indices... */
01420       if (ix < 0 || ix >= ctl->csi_nx ||
01421           iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01422         continue;
01423
01424       /* Get mean observation index... */
01425       obsmean[ix][iy][iz] += robs;
01426       obscount[ix][iy][iz]++;
01427     }
01428
01429     /* Analyze model data... */
01430     for (ip = 0; ip < atm->np; ip++) {
01431
01432       /* Check time... */
01433       if (atm->time[ip] < t0 || atm->time[ip] > t1)
01434         continue;
01435
01436       /* Get indices... */
01437       ix = (int) ((atm->lon[ip] - ctl->csi_lon0)
01438                   / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01439       iy = (int) ((atm->lat[ip] - ctl->csi_lat0)
01440                   / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01441       iz = (int) ((Z(atm->p[ip]) - ctl->csi_z0)
01442                   / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01443
01444       /* Check indices... */
01445       if (ix < 0 || ix >= ctl->csi_nx ||
01446           iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01447         continue;
01448
01449       /* Get total mass in grid cell... */
01450       modmean[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01451     }
01452
01453     /* Analyze all grid cells... */
01454     for (ix = 0; ix < ctl->csi_nx; ix++)
01455       for (iy = 0; iy < ctl->csi_ny; iy++)
01456         for (iz = 0; iz < ctl->csi_nz; iz++) {
01457
01458           /* Calculate mean observation index... */
01459           if (obscount[ix][iy][iz] > 0)
01460             obsmean[ix][iy][iz] /= obscount[ix][iy][iz];
01461
01462           /* Calculate column density... */
01463           if (modmean[ix][iy][iz] > 0) {
01464             dlon = (ctl->csi_lon1 - ctl->csi_lon0) / ctl->csi_nx;
01465             dlat = (ctl->csi_lat1 - ctl->csi_lat0) / ctl->csi_ny;
01466             lat = ctl->csi_lat0 + dlat * (iy + 0.5);
01467             area = dlat * M_PI * RE / 180. * dlon * M_PI * RE / 180.
01468               * cos(lat * M_PI / 180.);
01469             modmean[ix][iy][iz] /= (1e6 * area);
01470           }
01471
01472           /* Calculate CSI... */
01473           if (obscount[ix][iy][iz] > 0) {
01474             if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01475                 modmean[ix][iy][iz] >= ctl->csi_modmin)
01476               cx++;
01477             else if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01478                      modmean[ix][iy][iz] < ctl->csi_modmin)
01479               cy++;
01480             else if (obsmean[ix][iy][iz] < ctl->csi_obsmin &&
01481                      modmean[ix][iy][iz] >= ctl->csi_modmin)
```

```
01482             cz++;
01483         }
01484       }
01485
01486   /* Write output... */
01487   if (fmod(t, ctl->csi_dt_out) == 0) {
01488
01489     /* Write... */
01490     fprintf(out, "%.2f %d %d %d %d %d %g %g %g %g\n",
01491             t, cx, cy, cz, cx + cy, cx + cz,
01492             (cx + cy > 0) ? 100. * (cx + cz) / (cx + cy) : GSL_NAN,
01493             (cx + cy > 0) ? (100. * cx) / (cx + cy) : GSL_NAN,
01494             (cx + cz > 0) ? (100. * cz) / (cx + cz) : GSL_NAN,
01495             (cx + cy + cz > 0) ? (100. * cx) / (cx + cy + cz) : GSL_NAN);
01496
01497     /* Set counters to zero... */
01498     cx = cy = cz = 0;
01499   }
01500
01501   /* Close file... */
01502   if (t == ctl->t_stop)
01503     fclose(out);
01504 }
01505
01506 /*****************************************************************************/
01507
01508 void write_ens(
01509   const char *filename,
01510   ctl_t * ctl,
01511   atm_t * atm,
01512   double t) {
01513
01514   static FILE *out;
01515
01516   static double dummy, ens, lat, lon, p[NENS], q[NQ][NENS],
01517     t0, t1, x[NENS][3], xm[3];
01518
01519   static int init, ip, iq;
01520
01521   static size_t i, n;
01522
01523   /* Init... */
01524   if (!init) {
01525     init = 1;
01526
01527     /* Check quantities... */
01528     if (ctl->qnt_ens < 0)
01529       ERRMSG("Missing ensemble IDs!");
01530
01531     /* Create new file... */
01532     printf("Write ensemble data: %s\n", filename);
01533     if (!(out = fopen(filename, "w")))
01534       ERRMSG("Cannot create file!");
01535
01536     /* Write header... */
01537     fprintf(out,
01538             "# $1 = time [s]\n"
01539             "# $2 = altitude [km]\n"
01540             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01541     for (iq = 0; iq < ctl->nq; iq++)
01542       fprintf(out, "# $%d = %s (mean) [%s]\n", 5 + iq,
01543               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01544     for (iq = 0; iq < ctl->nq; iq++)
01545       fprintf(out, "# $%d = %s (sigma) [%s]\n", 5 + ctl->nq + iq,
01546               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01547     fprintf(out, "# $%d = number of members\n\n", 5 + 2 * ctl->nq);
01548   }
01549
01550   /* Set time interval... */
01551   t0 = t - 0.5 * ctl->dt_mod;
01552   t1 = t + 0.5 * ctl->dt_mod;
01553
01554   /* Init... */
01555   ens = GSL_NAN;
01556   n = 0;
01557
01558   /* Loop over air parcels... */
01559   for (ip = 0; ip < atm->np; ip++) {
01560
01561     /* Check time... */
01562     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01563       continue;
01564
01565     /* Check ensemble id... */
01566     if (atm->q[ctl->qnt_ens][ip] != ens) {
01567
01568       /* Write results... */
```

```
01569       if (n > 0) {
01570
01571         /* Get mean position... */
01572         xm[0] = xm[1] = xm[2] = 0;
01573         for (i = 0; i < n; i++) {
01574           xm[0] += x[i][0] / (double) n;
01575           xm[1] += x[i][1] / (double) n;
01576           xm[2] += x[i][2] / (double) n;
01577         }
01578         cart2geo(xm, &dummy, &lon, &lat);
01579         fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon,
01580                 lat);
01581
01582         /* Get quantity statistics... */
01583         for (iq = 0; iq < ctl->nq; iq++) {
01584           fprintf(out, " ");
01585           fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01586         }
01587         for (iq = 0; iq < ctl->nq; iq++) {
01588           fprintf(out, " ");
01589           fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01590         }
01591         fprintf(out, " %lu\n", n);
01592       }
01593
01594       /* Init new ensemble... */
01595       ens = atm->q[ctl->qnt_ens][ip];
01596       n = 0;
01597     }
01598
01599     /* Save data... */
01600     p[n] = atm->p[ip];
01601     geo2cart(0, atm->lon[ip], atm->lat[ip], x[n]);
01602     for (iq = 0; iq < ctl->nq; iq++)
01603       q[iq][n] = atm->q[iq][ip];
01604     if ((++n) >= NENS)
01605       ERRMSG("Too many data points!");
01606   }
01607
01608   /* Write results... */
01609   if (n > 0) {
01610
01611     /* Get mean position... */
01612     xm[0] = xm[1] = xm[2] = 0;
01613     for (i = 0; i < n; i++) {
01614       xm[0] += x[i][0] / (double) n;
01615       xm[1] += x[i][1] / (double) n;
01616       xm[2] += x[i][2] / (double) n;
01617     }
01618     cart2geo(xm, &dummy, &lon, &lat);
01619     fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon, lat);
01620
01621     /* Get quantity statistics... */
01622     for (iq = 0; iq < ctl->nq; iq++) {
01623       fprintf(out, " ");
01624       fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01625     }
01626     for (iq = 0; iq < ctl->nq; iq++) {
01627       fprintf(out, " ");
01628       fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01629     }
01630     fprintf(out, " %lu\n", n);
01631   }
01632
01633   /* Close file... */
01634   if (t == ctl->t_stop)
01635     fclose(out);
01636 }
01637
01638 /*****************************************************************************/
01639
01640 void write_grid(
01641   const char *filename,
01642   ctl_t * ctl,
01643   met_t * met0,
01644   met_t * met1,
01645   atm_t * atm,
01646   double t) {
01647
01648   FILE *in, *out;
01649
01650   char line[LEN];
01651
01652   static double grid_m[GX][GY][GZ], z, dz, lon, dlon, lat, dlat,
01653     area, rho_air, press, temp, cd, mmr, t0, t1, r;
01654
01655   static int ip, ix, iy, iz, year, mon, day, hour, min, sec;
```

```
01656
01657   /* Check dimensions... */
01658   if (ctl->grid_nx > GX || ctl->grid_ny > GY || ctl->grid_nz > GZ)
01659     ERRMSG("Grid dimensions too large!");
01660
01661   /* Check quantity index for mass... */
01662   if (ctl->qnt_m < 0)
01663     ERRMSG("Need quantity mass to write grid data!");
01664
01665   /* Set time interval for output... */
01666   t0 = t - 0.5 * ctl->dt_mod;
01667   t1 = t + 0.5 * ctl->dt_mod;
01668
01669   /* Set grid box size... */
01670   dz = (ctl->grid_z1 - ctl->grid_z0) / ctl->grid_nz;
01671   dlon = (ctl->grid_lon1 - ctl->grid_lon0) / ctl->grid_nx;
01672   dlat = (ctl->grid_lat1 - ctl->grid_lat0) / ctl->grid_ny;
01673
01674   /* Initialize grid... */
01675   for (ix = 0; ix < ctl->grid_nx; ix++)
01676     for (iy = 0; iy < ctl->grid_ny; iy++)
01677       for (iz = 0; iz < ctl->grid_nz; iz++)
01678         grid_m[ix][iy][iz] = 0;
01679
01680   /* Average data... */
01681   for (ip = 0; ip < atm->np; ip++)
01682     if (atm->time[ip] >= t0 && atm->time[ip] <= t1) {
01683
01684       /* Get index... */
01685       ix = (int) ((atm->lon[ip] - ctl->grid_lon0) / dlon);
01686       iy = (int) ((atm->lat[ip] - ctl->grid_lat0) / dlat);
01687       iz = (int) ((Z(atm->p[ip]) - ctl->grid_z0) / dz);
01688
01689       /* Check indices... */
01690       if (ix < 0 || ix >= ctl->grid_nx ||
01691           iy < 0 || iy >= ctl->grid_ny || iz < 0 || iz >= ctl->grid_nz)
01692         continue;
01693
01694       /* Add mass... */
01695       grid_m[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01696     }
01697
01698   /* Check if gnuplot output is requested... */
01699   if (ctl->grid_gpfile[0] != '-') {
01700
01701     /* Write info... */
01702     printf("Plot grid data: %s.png\n", filename);
01703
01704     /* Create gnuplot pipe... */
01705     if (!(out = popen("gnuplot", "w")))
01706       ERRMSG("Cannot create pipe to gnuplot!");
01707
01708     /* Set plot filename... */
01709     fprintf(out, "set out \"%s.png\"\n", filename);
01710
01711     /* Set time string... */
01712     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01713     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01714             year, mon, day, hour, min);
01715
01716     /* Dump gnuplot file to pipe... */
01717     if (!(in = fopen(ctl->grid_gpfile, "r")))
01718       ERRMSG("Cannot open file!");
01719     while (fgets(line, LEN, in))
01720       fprintf(out, "%s", line);
01721     fclose(in);
01722   }
01723
01724   else {
01725
01726     /* Write info... */
01727     printf("Write grid data: %s\n", filename);
01728
01729     /* Create file... */
01730     if (!(out = fopen(filename, "w")))
01731       ERRMSG("Cannot create file!");
01732   }
01733
01734   /* Write header... */
01735   fprintf(out,
01736           "# $1 = time [s]\n"
01737           "# $2 = altitude [km]\n"
01738           "# $3 = longitude [deg]\n"
01739           "# $4 = latitude [deg]\n"
01740           "# $5 = surface area [km^2]\n"
01741           "# $6 = layer width [km]\n"
01742           "# $7 = temperature [K]\n"
```

```
01743           "# $8 = column density [kg/m^2]\n"
01744           "# $9 = mass mixing ratio [1]\n\n");
01745
01746   /* Write data... */
01747   for (ix = 0; ix < ctl->grid_nx; ix++) {
01748     if (ix > 0 && ctl->grid_ny > 1 && !ctl->grid_sparse)
01749       fprintf(out, "\n");
01750     for (iy = 0; iy < ctl->grid_ny; iy++) {
01751       if (iy > 0 && ctl->grid_nz > 1 && !ctl->grid_sparse)
01752         fprintf(out, "\n");
01753       for (iz = 0; iz < ctl->grid_nz; iz++)
01754         if (!ctl->grid_sparse
01755             || ix == 0 || iy == 0 || iz == 0 || grid_m[ix][iy][iz] > 0) {
01756
01757           /* Set coordinates... */
01758           z = ctl->grid_z0 + dz * (iz + 0.5);
01759           lon = ctl->grid_lon0 + dlon * (ix + 0.5);
01760           lat = ctl->grid_lat0 + dlat * (iy + 0.5);
01761
01762           /* Get pressure and temperature... */
01763           press = P(z);
01764           intpol_met_time(met0, met1, t, press, lon, lat,
01765                           NULL, &temp, NULL, NULL, NULL, NULL, NULL);
01766
01767           /* Calculate surface area... */
01768           area = dlat * dlon * gsl_pow_2(RE * M_PI / 180.)
01769             * cos(lat * M_PI / 180.);
01770
01771           /* Calculate column density... */
01772           cd = grid_m[ix][iy][iz] / (1e6 * area);
01773
01774           /* Calculate mass mixing ratio... */
01775           rho_air = 100. * press / (287.058 * temp);
01776           mmr = grid_m[ix][iy][iz] / (rho_air * 1e6 * area * 1e3 * dz);
01777
01778           /* Write output... */
01779           fprintf(out, "%.2f %g %g %g %g %g %g %g %g\n",
01780                   t, z, lon, lat, area, dz, temp, cd, mmr);
01781         }
01782     }
01783   }
01784
01785   /* Close file... */
01786   fclose(out);
01787 }
01788
01789 /*****************************************************************************/
01790
01791 void write_prof(
01792   const char *filename,
01793   ctl_t * ctl,
01794   met_t * met0,
01795   met_t * met1,
01796   atm_t * atm,
01797   double t) {
01798
01799   static FILE *in, *out;
01800
01801   static char line[LEN];
01802
01803   static double mass[GX][GY][GZ], obsmean[GX][GY], tmean[GX][GY],
01804     rt, rlon, rlat, robs, t0, t1, area, dz, dlon, dlat, lon, lat, z,
01805     press, temp, rho_air, mmr, h2o, o3;
01806
01807   static int init, obscount[GX][GY], ip, ix, iy, iz;
01808
01809   /* Init... */
01810   if (!init) {
01811     init = 1;
01812
01813     /* Check quantity index for mass... */
01814     if (ctl->qnt_m < 0)
01815       ERRMSG("Need quantity mass!");
01816
01817     /* Check dimensions... */
01818     if (ctl->prof_nx > GX || ctl->prof_ny > GY || ctl->prof_nz > GZ)
01819       ERRMSG("Grid dimensions too large!");
01820
01821     /* Open observation data file... */
01822     printf("Read profile observation data: %s\n", ctl->prof_obsfile);
01823     if (!(in = fopen(ctl->prof_obsfile, "r")))
01824       ERRMSG("Cannot open file!");
01825
01826     /* Create new file... */
01827     printf("Write profile data: %s\n", filename);
01828     if (!(out = fopen(filename, "w")))
01829       ERRMSG("Cannot create file!");
```

```
01830
01831     /* Write header... */
01832     fprintf(out,
01833             "# $1  = time [s]\n"
01834             "# $2  = altitude [km]\n"
01835             "# $3  = longitude [deg]\n"
01836             "# $4  = latitude [deg]\n"
01837             "# $5  = pressure [hPa]\n"
01838             "# $6  = temperature [K]\n"
01839             "# $7  = mass mixing ratio [1]\n"
01840             "# $8  = H2O volume mixing ratio [1]\n"
01841             "# $9  = O3 volume mixing ratio [1]\n"
01842             "# $10 = mean BT index [K]\n");
01843
01844     /* Set grid box size... */
01845     dz = (ctl->prof_z1 - ctl->prof_z0) / ctl->prof_nz;
01846     dlon = (ctl->prof_lon1 - ctl->prof_lon0) / ctl->prof_nx;
01847     dlat = (ctl->prof_lat1 - ctl->prof_lat0) / ctl->prof_ny;
01848   }
01849
01850   /* Set time interval... */
01851   t0 = t - 0.5 * ctl->dt_mod;
01852   t1 = t + 0.5 * ctl->dt_mod;
01853
01854   /* Initialize... */
01855   for (ix = 0; ix < ctl->prof_nx; ix++)
01856     for (iy = 0; iy < ctl->prof_ny; iy++) {
01857       obsmean[ix][iy] = 0;
01858       obscount[ix][iy] = 0;
01859       tmean[ix][iy] = 0;
01860       for (iz = 0; iz < ctl->prof_nz; iz++)
01861         mass[ix][iy][iz] = 0;
01862     }
01863
01864   /* Read data... */
01865   while (fgets(line, LEN, in)) {
01866
01867     /* Read data... */
01868     if (sscanf(line, "%lg %lg %lg %lg", &rt, &rlon, &rlat, &robs) != 4)
01869       continue;
01870
01871     /* Check time... */
01872     if (rt < t0)
01873       continue;
01874     if (rt > t1)
01875       break;
01876
01877     /* Calculate indices... */
01878     ix = (int) ((rlon - ctl->prof_lon0) / dlon);
01879     iy = (int) ((rlat - ctl->prof_lat0) / dlat);
01880
01881     /* Check indices... */
01882     if (ix < 0 || ix >= ctl->prof_nx || iy < 0 || iy >= ctl->prof_ny)
01883       continue;
01884
01885     /* Get mean observation index... */
01886     obsmean[ix][iy] += robs;
01887     tmean[ix][iy] += rt;
01888     obscount[ix][iy]++;
01889   }
01890
01891   /* Analyze model data... */
01892   for (ip = 0; ip < atm->np; ip++) {
01893
01894     /* Check time... */
01895     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01896       continue;
01897
01898     /* Get indices... */
01899     ix = (int) ((atm->lon[ip] - ctl->prof_lon0) / dlon);
01900     iy = (int) ((atm->lat[ip] - ctl->prof_lat0) / dlat);
01901     iz = (int) ((Z(atm->p[ip]) - ctl->prof_z0) / dz);
01902
01903     /* Check indices... */
01904     if (ix < 0 || ix >= ctl->prof_nx ||
01905         iy < 0 || iy >= ctl->prof_ny || iz < 0 || iz >= ctl->prof_nz)
01906       continue;
01907
01908     /* Get total mass in grid cell... */
01909     mass[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01910   }
01911
01912   /* Extract profiles... */
01913   for (ix = 0; ix < ctl->prof_nx; ix++)
01914     for (iy = 0; iy < ctl->prof_ny; iy++)
01915       if (obscount[ix][iy] > 0) {
01916
```

```
01917         /* Write output... */
01918         fprintf(out, "\n");
01919
01920         /* Loop over altitudes... */
01921         for (iz = 0; iz < ctl->prof_nz; iz++) {
01922
01923           /* Set coordinates... */
01924           z = ctl->prof_z0 + dz * (iz + 0.5);
01925           lon = ctl->prof_lon0 + dlon * (ix + 0.5);
01926           lat = ctl->prof_lat0 + dlat * (iy + 0.5);
01927
01928           /* Get meteorological data... */
01929           press = P(z);
01930           intpol_met_time(met0, met1, t, press, lon, lat,
01931                           NULL, &temp, NULL, NULL, NULL, &h2o, &o3);
01932
01933           /* Calculate mass mixing ratio... */
01934           rho_air = 100. * press / (287.058 * temp);
01935           area = dlat * dlon * gsl_pow_2(M_PI * RE / 180.)
01936             * cos(lat * M_PI / 180.);
01937           mmr = mass[ix][iy][iz] / (rho_air * area * dz * 1e9);
01938
01939           /* Write output... */
01940           fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g\n",
01941                   tmean[ix][iy] / obscount[ix][iy],
01942                   z, lon, lat, press, temp, mmr, h2o, o3,
01943                   obsmean[ix][iy] / obscount[ix][iy]);
01944         }
01945       }
01946
01947   /* Close file... */
01948   if (t == ctl->t_stop)
01949     fclose(out);
01950 }
01951
01952 /*****************************************************************************/
01953
01954 void write_station(
01955   const char *filename,
01956   ctl_t * ctl,
01957   atm_t * atm,
01958   double t) {
01959
01960   static FILE *out;
01961
01962   static double rmax2, t0, t1, x0[3], x1[3];
01963
01964   static int init, ip, iq;
01965
01966   /* Init... */
01967   if (!init) {
01968     init = 1;
01969
01970     /* Write info... */
01971     printf("Write station data: %s\n", filename);
01972
01973     /* Create new file... */
01974     if (!(out = fopen(filename, "w")))
01975       ERRMSG("Cannot create file!");
01976
01977     /* Write header... */
01978     fprintf(out,
01979             "# $1 = time [s]\n"
01980             "# $2 = altitude [km]\n"
01981             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01982     for (iq = 0; iq < ctl->nq; iq++)
01983       fprintf(out, "# $%i = %s [%s]\n", (iq + 5),
01984               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01985     fprintf(out, "\n");
01986
01987     /* Set geolocation and search radius... */
01988     geo2cart(0, ctl->stat_lon, ctl->stat_lat, x0);
01989     rmax2 = gsl_pow_2(ctl->stat_r);
01990   }
01991
01992   /* Set time interval for output... */
01993   t0 = t - 0.5 * ctl->dt_mod;
01994   t1 = t + 0.5 * ctl->dt_mod;
01995
01996   /* Loop over air parcels... */
01997   for (ip = 0; ip < atm->np; ip++) {
01998
01999     /* Check time... */
02000     if (atm->time[ip] < t0 || atm->time[ip] > t1)
02001       continue;
02002
02003     /* Check station flag... */
```

```
02004     if (ctl->qnt_stat >= 0)
02005       if (atm->q[ctl->qnt_stat][ip])
02006         continue;
02007
02008     /* Get Cartesian coordinates... */
02009     geo2cart(0, atm->lon[ip], atm->lat[ip], x1);
02010
02011     /* Check horizontal distance... */
02012     if (DIST2(x0, x1) > rmax2)
02013       continue;
02014
02015     /* Set station flag... */
02016     if (ctl->qnt_stat >= 0)
02017       atm->q[ctl->qnt_stat][ip] = 1;
02018
02019     /* Write data... */
02020     fprintf(out, "%.2f %g %g %g",
02021             atm->time[ip], Z(atm->p[ip]), atm->lon[ip], atm->lat[ip]);
02022     for (iq = 0; iq < ctl->nq; iq++) {
02023       fprintf(out, " ");
02024       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
02025     }
02026     fprintf(out, "\n");
02027   }
02028
02029   /* Close file... */
02030   if (t == ctl->t_stop)
02031     fclose(out);
02032 }
```

5.13 libtrac.h File Reference

MPTRAC library declarations.

Data Structures

- struct ctl_t

  Control parameters.
- struct atm_t

  Atmospheric data.
- struct met_t

  Meteorological data.

Functions

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

  Convert Cartesian coordinates to geolocation.
- double deg2dx (double dlon, double lat)

  Convert degrees to horizontal distance.
- double deg2dy (double dlat)

  Convert degrees to horizontal distance.
- double dp2dz (double dp, double p)

  Convert pressure to vertical distance.
- double dx2deg (double dx, double lat)

  Convert horizontal distance to degrees.
- double dy2deg (double dy)

  Convert horizontal distance to degrees.
- double dz2dp (double dz, double p)

  Convert vertical distance to pressure.
- void geo2cart (double z, double lon, double lat, double *x)

Convert geolocation to Cartesian coordinates.

- void get_met (ctl_t *ctl, 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 *t, double *u, double *v, double *w, 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 *t, double *u, double *v, double *w, double *h2o, double *o3)

  Temporal interpolation of meteorological data.
- void jsec2time (double jsec, int *year, int *mon, int *day, int *hour, int *min, int *sec, double *remain)

  Convert seconds to date.
- int locate (double *xx, int n, double x)

  Find array index.
- void read_atm (const char *filename, ctl_t *ctl, atm_t *atm)

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

  Read control parameters.
- void read_met (ctl_t *ctl, 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_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_ml2pl (ctl_t *ctl, met_t *met, float var[EX][EY][EP])

  Convert meteorological data from model levels to pressure levels.
- void read_met_periodic (met_t *met)

  Create meteorological data with periodic boundary conditions.
- double scan_ctl (const char *filename, int argc, char *argv[ ], const char *varname, int arridx, const char *defvalue, char *value)

  Read a control parameter from file or command line.
- 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, int id, int mode)

  Measure wall-clock time.
- double tropopause (double t, double lat)
- void write_atm (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write atmospheric data.
- void write_csi (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write CSI data.
- void write_ens (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write ensemble data.
- void write_grid (const char *filename, ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, double t)

  Write gridded data.
- void write_prof (const char *filename, ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, double t)

  Write profile data.
- void write_station (const char *filename, ctl_t *ctl, atm_t *atm, double t)

  Write station data.

5.13.1 Detailed Description

MPTRAC library declarations.

Definition in file libtrac.h.

5.13.2 Function Documentation

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

Convert Cartesian coordinates to geolocation.

Definition at line 29 of file libtrac.c.

```
00033                {
00034
00035   double radius;
00036
00037   radius = NORM(x);
00038   *lat = asin(x[2] / radius) * 180 / M_PI;
00039   *lon = atan2(x[1], x[0]) * 180 / M_PI;
00040   *z = radius - RE;
00041 }
```

5.13.2.2 double deg2dx (double dlon, double lat)

Convert degrees to horizontal distance.

Definition at line 45 of file libtrac.c.

```
00047               {
00048
00049   return dlon * M_PI * RE / 180. * cos(lat / 180. * M_PI);
00050 }
```

5.13.2.3 double deg2dy (double dlat)

Convert degrees to horizontal distance.

Definition at line 54 of file libtrac.c.

```
00055                {
00056
00057   return dlat * M_PI * RE / 180.;
00058 }
```

5.13.2.4 double dp2dz (double dp, double p)

Convert pressure to vertical distance.

Definition at line 62 of file libtrac.c.

```
00064             {
00065
00066   return -dp * H0 / p;
00067 }
```

5.13.2.5 double dx2deg (double *dx*, double *lat*)

Convert horizontal distance to degrees.

Definition at line 71 of file libtrac.c.

```
00073               {
00074
00075   /* Avoid singularity at poles... */
00076   if (lat < -89.999 || lat > 89.999)
00077     return 0;
00078   else
00079     return dx * 180. / (M_PI * RE * cos(lat / 180. * M_PI));
00080 }
```

5.13.2.6 double dy2deg (double *dy*)

Convert horizontal distance to degrees.

Definition at line 84 of file libtrac.c.

```
00085               {
00086
00087   return dy * 180. / (M_PI * RE);
00088 }
```

5.13.2.7 double dz2dp (double *dz*, double *p*)

Convert vertical distance to pressure.

Definition at line 92 of file libtrac.c.

```
00094             {
00095
00096   return -dz * p / H0;
00097 }
```

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

Convert geolocation to Cartesian coordinates.

Definition at line 101 of file libtrac.c.

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

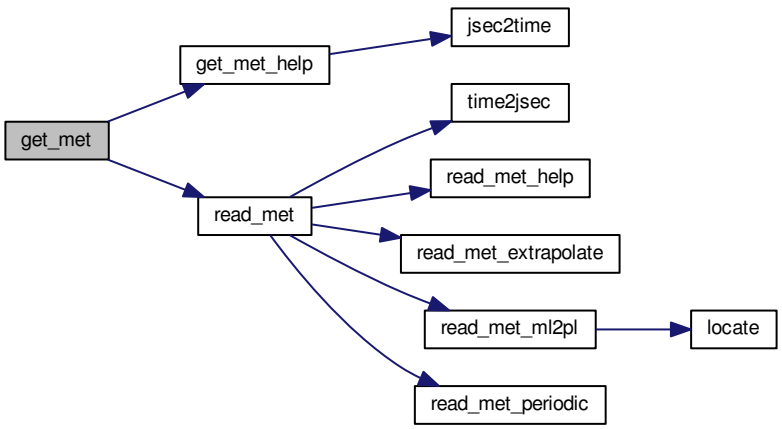
5.13.2.9 void get_met (ctl_t * *ctl,* char * *metbase,* double *t,* met_t * *met0,* met_t * *met1*)

Get meteorological data for given timestep.

Definition at line 117 of file libtrac.c.

```
00122                 {
00123
00124   char filename[LEN];
00125
00126   static int init;
00127
00128   /* Init... */
00129   if (!init) {
00130     init = 1;
00131
00132     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00133     read_met(ctl, filename, met0);
00134
00135     get_met_help(t + 1.0 * ctl->direction, 1, metbase, ctl->
      dt_met, filename);
00136     read_met(ctl, filename, met1);
00137   }
00138
00139   /* Read new data for forward trajectories... */
00140   if (t > met1->time && ctl->direction == 1) {
00141     memcpy(met0, met1, sizeof(met_t));
00142     get_met_help(t, 1, metbase, ctl->dt_met, filename);
00143     read_met(ctl, filename, met1);
00144   }
00145
00146   /* Read new data for backward trajectories... */
00147   if (t < met0->time && ctl->direction == -1) {
00148     memcpy(met1, met0, sizeof(met_t));
00149     get_met_help(t, -1, metbase, ctl->dt_met, filename);
00150     read_met(ctl, filename, met0);
00151   }
00152 }
```

Here is the call graph for this function:



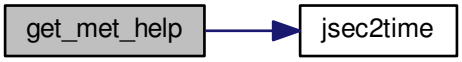
5.13.2.10 void get_met_help (double *t,* int *direct,* char * *metbase,* double *dt_met,* char * *filename*)

Get meteorological data for timestep.

Definition at line 156 of file libtrac.c.

```
00161                   {
00162
00163   double t6, r;
00164
00165   int year, mon, day, hour, min, sec;
00166
00167   /* Round time to fixed intervals... */
00168   if (direct == -1)
00169     t6 = floor(t / dt_met) * dt_met;
00170   else
00171     t6 = ceil(t / dt_met) * dt_met;
00172
00173   /* Decode time... */
00174   jsec2time(t6, &year, &mon, &day, &hour, &min, &sec, &r);
00175
00176   /* Set filename... */
00177   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", metbase, year, mon, day, hour);
00178 }
```

Here is the call graph for this function:



5.13.2.11 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 182 of file libtrac.c.

```
00188               {
00189
00190   double aux00, aux01, aux10, aux11;
00191
00192   /* Set variables... */
00193   aux00 = array[ix][iy];
00194   aux01 = array[ix][iy + 1];
00195   aux10 = array[ix + 1][iy];
00196   aux11 = array[ix + 1][iy + 1];
00197
00198   /* Interpolate horizontally... */
00199   aux00 = wy * (aux00 - aux01) + aux01;
00200   aux11 = wy * (aux10 - aux11) + aux11;
00201   *var = wx * (aux00 - aux11) + aux11;
00202 }
```

5.13.2.12 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 206 of file libtrac.c.

```
00214               {
00215
00216   double aux00, aux01, aux10, aux11;
00217
00218   /* Interpolate vertically... */
00219   aux00 = wp * (array[ix][iy][ip] - array[ix][iy][ip + 1])
00220     + array[ix][iy][ip + 1];
00221   aux01 = wp * (array[ix][iy + 1][ip] - array[ix][iy + 1][ip + 1])
00222     + array[ix][iy + 1][ip + 1];
00223   aux10 = wp * (array[ix + 1][iy][ip] - array[ix + 1][iy][ip + 1])
00224     + array[ix + 1][iy][ip + 1];
00225   aux11 = wp * (array[ix + 1][iy + 1][ip] - array[ix + 1][iy + 1][ip + 1])
00226     + array[ix + 1][iy + 1][ip + 1];
00227
00228   /* Interpolate horizontally... */
00229   aux00 = wy * (aux00 - aux01) + aux01;
00230   aux11 = wy * (aux10 - aux11) + aux11;
00231   *var = wx * (aux00 - aux11) + aux11;
00232 }
```

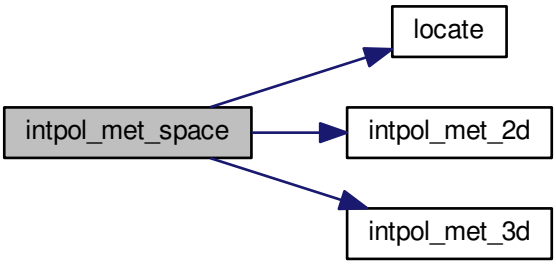
5.13.2.13 void intpol_met_space (met_t * *met*, double *p*, double *lon*, double *lat*, double * *ps*, double * *t*, double * *u*, double * *v*, double * *w*, double * *h2o*, double * *o3*)

Spatial interpolation of meteorological data.

Definition at line 236 of file libtrac.c.

```
00247               {
00248
00249   double wp, wx, wy;
00250
00251   int ip, ix, iy;
00252
00253   /* Check longitude... */
00254   if (met->lon[met->nx - 1] > 180 && lon < 0)
00255     lon += 360;
00256
00257   /* Get indices... */
00258   ip = locate(met->p, met->np, p);
00259   ix = locate(met->lon, met->nx, lon);
00260   iy = locate(met->lat, met->ny, lat);
00261
00262   /* Get weights... */
00263   wp = (met->p[ip + 1] - p) / (met->p[ip + 1] - met->p[ip]);
00264   wx = (met->lon[ix + 1] - lon) / (met->lon[ix + 1] - met->lon[ix]);
00265   wy = (met->lat[iy + 1] - lat) / (met->lat[iy + 1] - met->lat[iy]);
00266
00267   /* Interpolate... */
00268   if (ps != NULL)
00269     intpol_met_2d(met->ps, ix, iy, wx, wy, ps);
00270   if (t != NULL)
00271     intpol_met_3d(met->t, ip, ix, iy, wp, wx, wy, t);
00272   if (u != NULL)
00273     intpol_met_3d(met->u, ip, ix, iy, wp, wx, wy, u);
00274   if (v != NULL)
00275     intpol_met_3d(met->v, ip, ix, iy, wp, wx, wy, v);
00276   if (w != NULL)
00277     intpol_met_3d(met->w, ip, ix, iy, wp, wx, wy, w);
00278   if (h2o != NULL)
00279     intpol_met_3d(met->h2o, ip, ix, iy, wp, wx, wy, h2o);
00280   if (o3 != NULL)
00281     intpol_met_3d(met->o3, ip, ix, iy, wp, wx, wy, o3);
00282 }
```

Here is the call graph for this function:



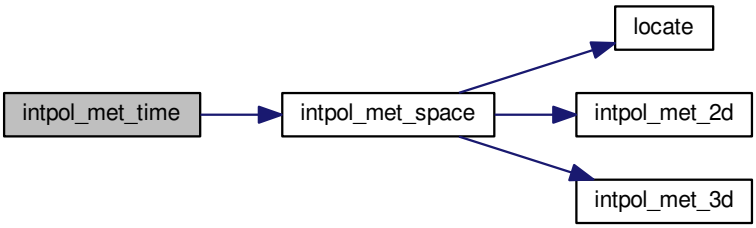
5.13.2.14 void intpol_met_time (met_t * *met0*, met_t * *met1*, double *ts*, double *p*, double *lon*, double *lat*, double * *ps*, double * *t*, double * *u*, double * *v*, double * *w*, double * *h2o*, double * *o3*)

Temporal interpolation of meteorological data.

Definition at line 286 of file libtrac.c.

```
00299              {
00300
00301   double h2o0, h2o1, o30, o31, ps0, ps1, t0, t1, u0, u1, v0, v1, w0, w1, wt;
00302
00303   /* Spatial interpolation... */
00304   intpol_met_space(met0, p, lon, lat,
00305                    ps == NULL ? NULL : &ps0,
00306                    t == NULL ? NULL : &t0,
00307                    u == NULL ? NULL : &u0,
00308                    v == NULL ? NULL : &v0,
00309                    w == NULL ? NULL : &w0,
00310                    h2o == NULL ? NULL : &h2o0, o3 == NULL ? NULL : &o30);
00311   intpol_met_space(met1, p, lon, lat,
00312                    ps == NULL ? NULL : &ps1,
00313                    t == NULL ? NULL : &t1,
00314                    u == NULL ? NULL : &u1,
00315                    v == NULL ? NULL : &v1,
00316                    w == NULL ? NULL : &w1,
00317                    h2o == NULL ? NULL : &h2o1, o3 == NULL ? NULL : &o31);
00318
00319   /* Get weighting factor... */
00320   wt = (met1->time - ts) / (met1->time - met0->time);
00321
00322   /* Interpolate... */
00323   if (ps != NULL)
00324     *ps = wt * (ps0 - ps1) + ps1;
00325   if (t != NULL)
00326     *t = wt * (t0 - t1) + t1;
00327   if (u != NULL)
00328     *u = wt * (u0 - u1) + u1;
00329   if (v != NULL)
00330     *v = wt * (v0 - v1) + v1;
00331   if (w != NULL)
00332     *w = wt * (w0 - w1) + w1;
00333   if (h2o != NULL)
00334     *h2o = wt * (h2o0 - h2o1) + h2o1;
00335   if (o3 != NULL)
00336     *o3 = wt * (o30 - o31) + o31;
00337 }
```

Here is the call graph for this function:



5.13.2.15 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 341 of file libtrac.c.

```
00349                   {
00350
00351   struct tm t0, *t1;
00352
00353   time_t jsec0;
00354
00355   t0.tm_year = 100;
00356   t0.tm_mon = 0;
00357   t0.tm_mday = 1;
00358   t0.tm_hour = 0;
00359   t0.tm_min = 0;
00360   t0.tm_sec = 0;
00361
00362   jsec0 = (time_t) jsec + timegm(&t0);
00363   t1 = gmtime(&jsec0);
00364
00365   *year = t1->tm_year + 1900;
00366   *mon = t1->tm_mon + 1;
00367   *day = t1->tm_mday;
00368   *hour = t1->tm_hour;
00369   *min = t1->tm_min;
00370   *sec = t1->tm_sec;
00371   *remain = jsec - floor(jsec);
00372 }
```

5.13.2.16 int locate (double * *xx*, int *n*, double *x*)

Find array index.

Definition at line 376 of file libtrac.c.

```
00379             {
00380
00381   int i, ilo, ihi;
00382
00383   ilo = 0;
00384   ihi = n - 1;
00385   i = (ihi + ilo) >> 1;
00386
00387   if (xx[i] < xx[i + 1])
00388     while (ihi > ilo + 1) {
00389       i = (ihi + ilo) >> 1;
00390       if (xx[i] > x)
```

```
00391         ihi = i;
00392       else
00393         ilo = i;
00394     } else
00395       while (ihi > ilo + 1) {
00396         i = (ihi + ilo) >> 1;
00397         if (xx[i] <= x)
00398           ihi = i;
00399         else
00400           ilo = i;
00401       }
00402
00403     return ilo;
00404   }
```

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

Read atmospheric data.

Definition at line 408 of file libtrac.c.

```
00411                 {
00412
00413     FILE *in;
00414
00415     char line[LEN], *tok;
00416
00417     int iq;
00418
00419     /* Init... */
00420     atm->np = 0;
00421
00422     /* Write info... */
00423     printf("Read atmospheric data: %s\n", filename);
00424
00425     /* Open file... */
00426     if (!(in = fopen(filename, "r")))
00427       ERRMSG("Cannot open file!");
00428
00429     /* Read line... */
00430     while (fgets(line, LEN, in)) {
00431
00432       /* Read data... */
00433       TOK(line, tok, "%lg", atm->time[atm->np]);
00434       TOK(NULL, tok, "%lg", atm->p[atm->np]);
00435       TOK(NULL, tok, "%lg", atm->lon[atm->np]);
00436       TOK(NULL, tok, "%lg", atm->lat[atm->np]);
00437       for (iq = 0; iq < ctl->nq; iq++)
00438         TOK(NULL, tok, "%lg", atm->q[iq][atm->np]);
00439
00440       /* Convert altitude to pressure... */
00441       atm->p[atm->np] = P(atm->p[atm->np]);
00442
00443       /* Increment data point counter... */
00444       if ((++atm->np) > NP)
00445         ERRMSG("Too many data points!");
00446     }
00447
00448     /* Close file... */
00449     fclose(in);
00450
00451     /* Check number of points... */
00452     if (atm->np < 1)
00453       ERRMSG("Can not read any data!");
00454   }
```

5.13.2.18 void read_ctl (const char * *filename*, int *argc*, char * *argv[]*, ctl_t * *ctl*)

Read control parameters.

Definition at line 458 of file libtrac.c.

```
00462               {
00463
00464   int ip, iq;
00465
00466   /* Write info... */
00467   printf("\nMassive-Parallel Trajectory Calculations (MPTRAC)\n"
00468          "(executable: %s | compiled: %s, %s)\n\n",
00469          argv[0], __DATE__, __TIME__);
00470
00471   /* Initialize quantity indices... */
00472   ctl->qnt_ens = -1;
00473   ctl->qnt_m = -1;
00474   ctl->qnt_r = -1;
00475   ctl->qnt_rho = -1;
00476   ctl->qnt_ps = -1;
00477   ctl->qnt_p = -1;
00478   ctl->qnt_t = -1;
00479   ctl->qnt_u = -1;
00480   ctl->qnt_v = -1;
00481   ctl->qnt_w = -1;
00482   ctl->qnt_h2o = -1;
00483   ctl->qnt_o3 = -1;
00484   ctl->qnt_theta = -1;
00485   ctl->qnt_pv = -1;
00486   ctl->qnt_tice = -1;
00487   ctl->qnt_tsts = -1;
00488   ctl->qnt_tnat = -1;
00489   ctl->qnt_gw_wind = -1;
00490   ctl->qnt_gw_sso = -1;
00491   ctl->qnt_gw_var = -1;
00492   ctl->qnt_stat = -1;
00493
00494   /* Read quantities... */
00495   ctl->nq = (int) scan_ctl(filename, argc, argv, "NQ", -1, "0", NULL);
00496   if (ctl->nq > NQ)
00497     ERRMSG("Too many quantities!");
00498   for (iq = 0; iq < ctl->nq; iq++) {
00499
00500     /* Read quantity name and format... */
00501     scan_ctl(filename, argc, argv, "QNT_NAME", iq, "", ctl->qnt_name[iq]);
00502     scan_ctl(filename, argc, argv, "QNT_FORMAT", iq, "%g",
00503              ctl->qnt_format[iq]);
00504
00505     /* Try to identify quantity... */
00506     if (strcmp(ctl->qnt_name[iq], "ens") == 0) {
00507       ctl->qnt_ens = iq;
00508       sprintf(ctl->qnt_unit[iq], "-");
00509     } else if (strcmp(ctl->qnt_name[iq], "m") == 0) {
00510       ctl->qnt_m = iq;
00511       sprintf(ctl->qnt_unit[iq], "kg");
00512     } else if (strcmp(ctl->qnt_name[iq], "r") == 0) {
00513       ctl->qnt_r = iq;
00514       sprintf(ctl->qnt_unit[iq], "m");
00515     } else if (strcmp(ctl->qnt_name[iq], "rho") == 0) {
00516       ctl->qnt_rho = iq;
00517       sprintf(ctl->qnt_unit[iq], "kg/m^3");
00518     } else if (strcmp(ctl->qnt_name[iq], "ps") == 0) {
00519       ctl->qnt_ps = iq;
00520       sprintf(ctl->qnt_unit[iq], "hPa");
00521     } else if (strcmp(ctl->qnt_name[iq], "p") == 0) {
00522       ctl->qnt_p = iq;
00523       sprintf(ctl->qnt_unit[iq], "hPa");
00524     } else if (strcmp(ctl->qnt_name[iq], "t") == 0) {
00525       ctl->qnt_t = iq;
00526       sprintf(ctl->qnt_unit[iq], "K");
00527     } else if (strcmp(ctl->qnt_name[iq], "u") == 0) {
00528       ctl->qnt_u = iq;
00529       sprintf(ctl->qnt_unit[iq], "m/s");
00530     } else if (strcmp(ctl->qnt_name[iq], "v") == 0) {
00531       ctl->qnt_v = iq;
00532       sprintf(ctl->qnt_unit[iq], "m/s");
00533     } else if (strcmp(ctl->qnt_name[iq], "w") == 0) {
00534       ctl->qnt_w = iq;
00535       sprintf(ctl->qnt_unit[iq], "hPa/s");
00536     } else if (strcmp(ctl->qnt_name[iq], "h2o") == 0) {
00537       ctl->qnt_h2o = iq;
00538       sprintf(ctl->qnt_unit[iq], "1");
00539     } else if (strcmp(ctl->qnt_name[iq], "o3") == 0) {
00540       ctl->qnt_o3 = iq;
00541       sprintf(ctl->qnt_unit[iq], "1");
00542     } else if (strcmp(ctl->qnt_name[iq], "theta") == 0) {
00543       ctl->qnt_theta = iq;
00544       sprintf(ctl->qnt_unit[iq], "K");
00545     } else if (strcmp(ctl->qnt_name[iq], "pv") == 0) {
00546       ctl->qnt_pv = iq;
00547       sprintf(ctl->qnt_unit[iq], "PVU");
00548     } else if (strcmp(ctl->qnt_name[iq], "tice") == 0) {
```

```
00549       ctl->qnt_tice = iq;
00550       sprintf(ctl->qnt_unit[iq], "K");
00551     } else if (strcmp(ctl->qnt_name[iq], "tsts") == 0) {
00552       ctl->qnt_tsts = iq;
00553       sprintf(ctl->qnt_unit[iq], "K");
00554     } else if (strcmp(ctl->qnt_name[iq], "tnat") == 0) {
00555       ctl->qnt_tnat = iq;
00556       sprintf(ctl->qnt_unit[iq], "K");
00557     } else if (strcmp(ctl->qnt_name[iq], "gw_wind") == 0) {
00558       ctl->qnt_gw_wind = iq;
00559       sprintf(ctl->qnt_unit[iq], "m/s");
00560     } else if (strcmp(ctl->qnt_name[iq], "gw_sso") == 0) {
00561       ctl->qnt_gw_sso = iq;
00562       sprintf(ctl->qnt_unit[iq], "m^2");
00563     } else if (strcmp(ctl->qnt_name[iq], "gw_var") == 0) {
00564       ctl->qnt_gw_var = iq;
00565       sprintf(ctl->qnt_unit[iq], "K^2");
00566     } else if (strcmp(ctl->qnt_name[iq], "stat") == 0) {
00567       ctl->qnt_stat = iq;
00568       sprintf(ctl->qnt_unit[iq], "-");
00569     } else
00570       scan_ctl(filename, argc, argv, "QNT_UNIT", iq, "", ctl->qnt_unit[iq]);
00571   }
00572
00573   /* Time steps of simulation... */
00574   ctl->direction =
00575     (int) scan_ctl(filename, argc, argv, "DIRECTION", -1, "1", NULL);
00576   if (ctl->direction != -1 && ctl->direction != 1)
00577     ERRMSG("Set DIRECTION to -1 or 1!");
00578   ctl->t_start =
00579     scan_ctl(filename, argc, argv, "T_START", -1, "-1e100", NULL);
00580   ctl->t_stop = scan_ctl(filename, argc, argv, "T_STOP", -1, "-1e100", NULL);
00581   ctl->dt_mod = scan_ctl(filename, argc, argv, "DT_MOD", -1, "600", NULL);
00582
00583   /* Meteorological data... */
00584   ctl->dt_met = scan_ctl(filename, argc, argv, "DT_MET", -1, "21600", NULL);
00585   ctl->met_np = (int) scan_ctl(filename, argc, argv, "MET_NP", -1, "0", NULL);
00586   if (ctl->met_np > EP)
00587     ERRMSG("Too many levels!");
00588   for (ip = 0; ip < ctl->met_np; ip++)
00589     ctl->met_p[ip] = scan_ctl(filename, argc, argv, "MET_P", ip, "", NULL);
00590
00591   /* Isosurface parameters... */
00592   ctl->isosurf
00593     = (int) scan_ctl(filename, argc, argv, "ISOSURF", -1, "0", NULL);
00594   scan_ctl(filename, argc, argv, "BALLOON", -1, "-", ctl->balloon);
00595
00596   /* Diffusion parameters... */
00597   ctl->turb_dx_trop
00598     = scan_ctl(filename, argc, argv, "TURB_DX_TROP", -1, "50.0", NULL);
00599   ctl->turb_dx_strat
00600     = scan_ctl(filename, argc, argv, "TURB_DX_STRAT", -1, "0.0", NULL);
00601   ctl->turb_dz_trop
00602     = scan_ctl(filename, argc, argv, "TURB_DZ_TROP", -1, "0.0", NULL);
00603   ctl->turb_dz_strat
00604     = scan_ctl(filename, argc, argv, "TURB_DZ_STRAT", -1, "0.1", NULL);
00605   ctl->turb_meso =
00606     scan_ctl(filename, argc, argv, "TURB_MESO", -1, "0.16", NULL);
00607
00608   /* Life time of particles... */
00609   ctl->tdec_trop = scan_ctl(filename, argc, argv, "TDEC_TROP", -1, "0", NULL);
00610   ctl->tdec_strat =
00611     scan_ctl(filename, argc, argv, "TDEC_STRAT", -1, "0", NULL);
00612
00613   /* PSC analysis... */
00614   ctl->psc_h2o = scan_ctl(filename, argc, argv, "PSC_H2O", -1, "4e-6", NULL);
00615   ctl->psc_hno3 =
00616     scan_ctl(filename, argc, argv, "PSC_HNO3", -1, "9e-9", NULL);
00617
00618   /* Gravity wave analysis... */
00619   scan_ctl(filename, argc, argv, "GW_BASENAME", -1, "-", ctl->
      gw_basename);
00620
00621   /* Output of atmospheric data... */
00622   scan_ctl(filename, argc, argv, "ATM_BASENAME", -1, "-", ctl->
      atm_basename);
00623   scan_ctl(filename, argc, argv, "ATM_GPFILE", -1, "-", ctl->atm_gpfile);
00624   ctl->atm_dt_out =
00625     scan_ctl(filename, argc, argv, "ATM_DT_OUT", -1, "86400", NULL);
00626   ctl->atm_filter =
00627     (int) scan_ctl(filename, argc, argv, "ATM_FILTER", -1, "0", NULL);
00628
00629   /* Output of CSI data... */
00630   scan_ctl(filename, argc, argv, "CSI_BASENAME", -1, "-", ctl->
      csi_basename);
00631   ctl->csi_dt_out =
00632     scan_ctl(filename, argc, argv, "CSI_DT_OUT", -1, "86400", NULL);
```

```
00633   scan_ctl(filename, argc, argv, "CSI_OBSFILE", -1, "obs.tab",
00634            ctl->csi_obsfile);
00635   ctl->csi_obsmin =
00636     scan_ctl(filename, argc, argv, "CSI_OBSMIN", -1, "0", NULL);
00637   ctl->csi_modmin =
00638     scan_ctl(filename, argc, argv, "CSI_MODMIN", -1, "0", NULL);
00639   ctl->csi_z0 = scan_ctl(filename, argc, argv, "CSI_Z0", -1, "0", NULL);
00640   ctl->csi_z1 = scan_ctl(filename, argc, argv, "CSI_Z1", -1, "100", NULL);
00641   ctl->csi_nz = (int) scan_ctl(filename, argc, argv, "CSI_NZ", -1, "1", NULL);
00642   ctl->csi_lon0 =
00643     scan_ctl(filename, argc, argv, "CSI_LON0", -1, "-180", NULL);
00644   ctl->csi_lon1 = scan_ctl(filename, argc, argv, "CSI_LON1", -1, "180", NULL);
00645   ctl->csi_nx =
00646     (int) scan_ctl(filename, argc, argv, "CSI_NX", -1, "360", NULL);
00647   ctl->csi_lat0 = scan_ctl(filename, argc, argv, "CSI_LAT0", -1, "-90", NULL);
00648   ctl->csi_lat1 = scan_ctl(filename, argc, argv, "CSI_LAT1", -1, "90", NULL);
00649   ctl->csi_ny =
00650     (int) scan_ctl(filename, argc, argv, "CSI_NY", -1, "180", NULL);
00651
00652   /* Output of ensemble data... */
00653   scan_ctl(filename, argc, argv, "ENS_BASENAME", -1, "-", ctl->
      ens_basename);
00654
00655   /* Output of grid data... */
00656   scan_ctl(filename, argc, argv, "GRID_BASENAME", -1, "-",
00657            ctl->grid_basename);
00658   scan_ctl(filename, argc, argv, "GRID_GPFILE", -1, "-", ctl->
      grid_gpfile);
00659   ctl->grid_dt_out =
00660     scan_ctl(filename, argc, argv, "GRID_DT_OUT", -1, "86400", NULL);
00661   ctl->grid_sparse =
00662     (int) scan_ctl(filename, argc, argv, "GRID_SPARSE", -1, "0", NULL);
00663   ctl->grid_z0 = scan_ctl(filename, argc, argv, "GRID_Z0", -1, "0", NULL);
00664   ctl->grid_z1 = scan_ctl(filename, argc, argv, "GRID_Z1", -1, "100", NULL);
00665   ctl->grid_nz =
00666     (int) scan_ctl(filename, argc, argv, "GRID_NZ", -1, "1", NULL);
00667   ctl->grid_lon0 =
00668     scan_ctl(filename, argc, argv, "GRID_LON0", -1, "-180", NULL);
00669   ctl->grid_lon1 =
00670     scan_ctl(filename, argc, argv, "GRID_LON1", -1, "180", NULL);
00671   ctl->grid_nx =
00672     (int) scan_ctl(filename, argc, argv, "GRID_NX", -1, "360", NULL);
00673   ctl->grid_lat0 =
00674     scan_ctl(filename, argc, argv, "GRID_LAT0", -1, "-90", NULL);
00675   ctl->grid_lat1 =
00676     scan_ctl(filename, argc, argv, "GRID_LAT1", -1, "90", NULL);
00677   ctl->grid_ny =
00678     (int) scan_ctl(filename, argc, argv, "GRID_NY", -1, "180", NULL);
00679
00680   /* Output of profile data... */
00681   scan_ctl(filename, argc, argv, "PROF_BASENAME", -1, "-",
00682            ctl->prof_basename);
00683   scan_ctl(filename, argc, argv, "PROF_OBSFILE", -1, "-", ctl->
      prof_obsfile);
00684   ctl->prof_z0 = scan_ctl(filename, argc, argv, "PROF_Z0", -1, "0", NULL);
00685   ctl->prof_z1 = scan_ctl(filename, argc, argv, "PROF_Z1", -1, "60", NULL);
00686   ctl->prof_nz =
00687     (int) scan_ctl(filename, argc, argv, "PROF_NZ", -1, "60", NULL);
00688   ctl->prof_lon0 =
00689     scan_ctl(filename, argc, argv, "PROF_LON0", -1, "-180", NULL);
00690   ctl->prof_lon1 =
00691     scan_ctl(filename, argc, argv, "PROF_LON1", -1, "180", NULL);
00692   ctl->prof_nx =
00693     (int) scan_ctl(filename, argc, argv, "PROF_NX", -1, "360", NULL);
00694   ctl->prof_lat0 =
00695     scan_ctl(filename, argc, argv, "PROF_LAT0", -1, "-90", NULL);
00696   ctl->prof_lat1 =
00697     scan_ctl(filename, argc, argv, "PROF_LAT1", -1, "90", NULL);
00698   ctl->prof_ny =
00699     (int) scan_ctl(filename, argc, argv, "PROF_NY", -1, "180", NULL);
00700
00701   /* Output of station data... */
00702   scan_ctl(filename, argc, argv, "STAT_BASENAME", -1, "-",
00703            ctl->stat_basename);
00704   ctl->stat_lon = scan_ctl(filename, argc, argv, "STAT_LON", -1, "0", NULL);
00705   ctl->stat_lat = scan_ctl(filename, argc, argv, "STAT_LAT", -1, "0", NULL);
00706   ctl->stat_r = scan_ctl(filename, argc, argv, "STAT_R", -1, "50", NULL);
00707 }
```

Here is the call graph for this function:

5.13.2.19 void read_met (ctl_t * *ctl*, char * *filename*, met_t * *met*)

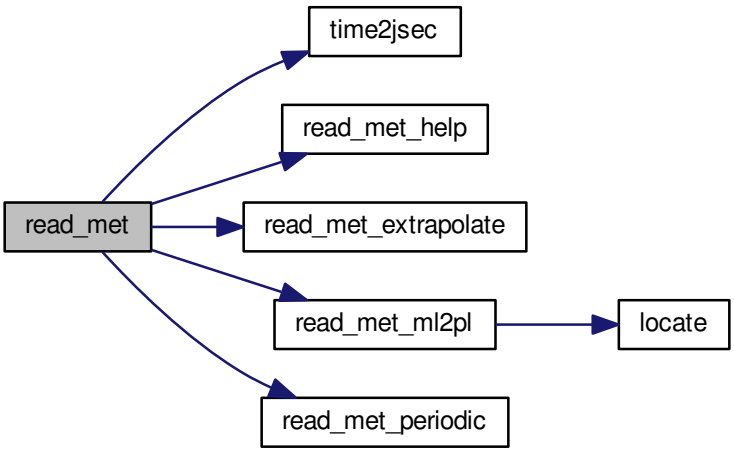
Read meteorological data file.

Definition at line 711 of file libtrac.c.

```
00714                {
00715
00716   char tstr[10];
00717
00718   static float help[EX * EY];
00719
00720   int ix, iy, ip, dimid, ncid, varid, year, mon, day, hour;
00721
00722   size_t np, nx, ny;
00723
00724   /* Write info... */
00725   printf("Read meteorological data: %s\n", filename);
00726
00727   /* Get time from filename... */
00728   sprintf(tstr, "%.4s", &filename[strlen(filename) - 16]);
00729   year = atoi(tstr);
00730   sprintf(tstr, "%.2s", &filename[strlen(filename) - 11]);
00731   mon = atoi(tstr);
00732   sprintf(tstr, "%.2s", &filename[strlen(filename) - 8]);
00733   day = atoi(tstr);
00734   sprintf(tstr, "%.2s", &filename[strlen(filename) - 5]);
00735   hour = atoi(tstr);
00736   time2jsec(year, mon, day, hour, 0, 0, 0, &met->time);
00737
00738   /* Open netCDF file... */
00739   NC(nc_open(filename, NC_NOWRITE, &ncid));
00740
00741   /* Get dimensions... */
00742   NC(nc_inq_dimid(ncid, "lon", &dimid));
00743   NC(nc_inq_dimlen(ncid, dimid, &nx));
00744   if (nx > EX)
00745     ERRMSG("Too many longitudes!");
00746
00747   NC(nc_inq_dimid(ncid, "lat", &dimid));
00748   NC(nc_inq_dimlen(ncid, dimid, &ny));
00749   if (ny > EY)
00750     ERRMSG("Too many latitudes!");
00751
00752   NC(nc_inq_dimid(ncid, "lev", &dimid));
00753   NC(nc_inq_dimlen(ncid, dimid, &np));
00754   if (np > EP)
00755     ERRMSG("Too many levels!");
00756
00757   /* Store dimensions... */
00758   met->np = (int) np;
00759   met->nx = (int) nx;
00760   met->ny = (int) ny;
00761
00762   /* Get horizontal grid... */
00763   NC(nc_inq_varid(ncid, "lon", &varid));
00764   NC(nc_get_var_double(ncid, varid, met->lon));
00765   NC(nc_inq_varid(ncid, "lat", &varid));
00766   NC(nc_get_var_double(ncid, varid, met->lat));
00767
00768   /* Read meteorological data... */
00769   read_met_help(ncid, "t", "T", met, met->t, 1.0);
00770   read_met_help(ncid, "u", "U", met, met->u, 1.0);
```

```
00771   read_met_help(ncid, "v", "V", met, met->v, 1.0);
00772   read_met_help(ncid, "w", "W", met, met->w, 0.01f);
00773   read_met_help(ncid, "q", "Q", met, met->h2o, 1.608f);
00774   read_met_help(ncid, "o3", "O3", met, met->o3, 0.602f);
00775
00776   /* Meteo data on pressure levels... */
00777   if (ctl->met_np <= 0) {
00778
00779     /* Read pressure levels from file... */
00780     NC(nc_inq_varid(ncid, "lev", &varid));
00781     NC(nc_get_var_double(ncid, varid, met->p));
00782     for (ip = 0; ip < met->np; ip++)
00783       met->p[ip] /= 100.;
00784
00785     /* Extrapolate data for lower boundary... */
00786     read_met_extrapolate(met);
00787   }
00788
00789   /* Meteo data on model levels... */
00790   else {
00791
00792     /* Read pressure data from file... */
00793     read_met_help(ncid, "pl", "PL", met, met->pl, 0.01f);
00794
00795     /* Interpolate from model levels to pressure levels... */
00796     read_met_ml2pl(ctl, met, met->t);
00797     read_met_ml2pl(ctl, met, met->u);
00798     read_met_ml2pl(ctl, met, met->v);
00799     read_met_ml2pl(ctl, met, met->w);
00800     read_met_ml2pl(ctl, met, met->h2o);
00801     read_met_ml2pl(ctl, met, met->o3);
00802
00803     /* Set pressure levels... */
00804     met->np = ctl->met_np;
00805     for (ip = 0; ip < met->np; ip++)
00806       met->p[ip] = ctl->met_p[ip];
00807   }
00808
00809   /* Check ordering of pressure levels... */
00810   for (ip = 1; ip < met->np; ip++)
00811     if (met->p[ip - 1] < met->p[ip])
00812       ERRMSG("Pressure levels must be descending!");
00813
00814   /* Read surface pressure... */
00815   if (nc_inq_varid(ncid, "PS", &varid) == NC_NOERR) {
00816     NC(nc_get_var_float(ncid, varid, help));
00817     for (iy = 0; iy < met->ny; iy++)
00818       for (ix = 0; ix < met->nx; ix++)
00819         met->ps[ix][iy] = help[iy * met->nx + ix] / 100.;
00820   } else if (nc_inq_varid(ncid, "LNSP", &varid) == NC_NOERR) {
00821     NC(nc_get_var_float(ncid, varid, help));
00822     for (iy = 0; iy < met->ny; iy++)
00823       for (ix = 0; ix < met->nx; ix++)
00824         met->ps[ix][iy] = exp(help[iy * met->nx + ix]) / 100.;
00825   } else
00826     for (ix = 0; ix < met->nx; ix++)
00827       for (iy = 0; iy < met->ny; iy++)
00828         met->ps[ix][iy] = met->p[0];
00829
00830   /* Create periodic boundary conditions... */
00831   read_met_periodic(met);
00832
00833   /* Close file... */
00834   NC(nc_close(ncid));
00835 }
```

Here is the call graph for this function:



5.13.2.20 void read_met_extrapolate (met_t * *met*)

Extrapolate meteorological data at lower boundary.

Definition at line 839 of file libtrac.c.

```
00840                {
00841
00842   int ip, ip0, ix, iy;
00843
00844   /* Loop over columns... */
00845   for (ix = 0; ix < met->nx; ix++)
00846     for (iy = 0; iy < met->ny; iy++) {
00847
00848       /* Find lowest valid data point... */
00849       for (ip0 = met->np - 1; ip0 >= 0; ip0--)
00850         if (!gsl_finite(met->t[ix][iy][ip0])
00851             || !gsl_finite(met->u[ix][iy][ip0])
00852             || !gsl_finite(met->v[ix][iy][ip0])
00853             || !gsl_finite(met->w[ix][iy][ip0]))
00854           break;
00855
00856       /* Extrapolate... */
00857       for (ip = ip0; ip >= 0; ip--) {
00858         met->t[ix][iy][ip] = met->t[ix][iy][ip + 1];
00859         met->u[ix][iy][ip] = met->u[ix][iy][ip + 1];
00860         met->v[ix][iy][ip] = met->v[ix][iy][ip + 1];
00861         met->w[ix][iy][ip] = met->w[ix][iy][ip + 1];
00862         met->h2o[ix][iy][ip] = met->h2o[ix][iy][ip + 1];
00863         met->o3[ix][iy][ip] = met->o3[ix][iy][ip + 1];
00864       }
00865     }
00866 }
```

5.13.2.21 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 870 of file libtrac.c.

```
00876             {
00877
00878   static float help[EX * EY * EP];
00879
00880   int ip, ix, iy, n = 0, varid;
00881
00882   /* Check if variable exists... */
00883   if (nc_inq_varid(ncid, varname, &varid) != NC_NOERR)
00884     if (nc_inq_varid(ncid, varname2, &varid) != NC_NOERR)
00885       return;
00886
00887   /* Read data... */
00888   NC(nc_get_var_float(ncid, varid, help));
00889
00890   /* Copy and check data... */
00891   for (ip = 0; ip < met->np; ip++)
00892     for (iy = 0; iy < met->ny; iy++)
00893       for (ix = 0; ix < met->nx; ix++) {
00894         dest[ix][iy][ip] = scl * help[n++];
00895         if (fabs(dest[ix][iy][ip] / scl) > 1e14)
00896           dest[ix][iy][ip] = GSL_NAN;
00897       }
00898 }
```

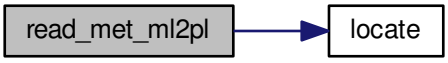
5.13.2.22 void read_met_ml2pl (ctl_t * *ctl*, met_t * *met*, float *var[EX][EY][EP]*)

Convert meteorological data from model levels to pressure levels.

Definition at line 902 of file libtrac.c.

```
00905                         {
00906
00907   double aux[EP], p[EP], pt;
00908
00909   int ip, ip2, ix, iy;
00910
00911   /* Loop over columns... */
00912   for (ix = 0; ix < met->nx; ix++)
00913     for (iy = 0; iy < met->ny; iy++) {
00914
00915       /* Copy pressure profile... */
00916       for (ip = 0; ip < met->np; ip++)
00917         p[ip] = met->pl[ix][iy][ip];
00918
00919       /* Interpolate... */
00920       for (ip = 0; ip < ctl->met_np; ip++) {
00921         pt = ctl->met_p[ip];
00922         if ((pt > p[0] && p[0] > p[1]) || (pt < p[0] && p[0] < p[1]))
00923           pt = p[0];
00924         else if ((pt > p[met->np - 1] && p[1] > p[0])
00925                  || (pt < p[met->np - 1] && p[1] < p[0]))
00926           pt = p[met->np - 1];
00927         ip2 = locate(p, met->np, pt);
00928         aux[ip] = LIN(p[ip2], var[ix][iy][ip2],
00929                       p[ip2 + 1], var[ix][iy][ip2 + 1], pt);
00930       }
00931
00932       /* Copy data... */
00933       for (ip = 0; ip < ctl->met_np; ip++)
00934         var[ix][iy][ip] = (float) aux[ip];
00935     }
00936 }
```

Here is the call graph for this function:



5.13.2.23 void read_met_periodic (met_t * *met*)

Create meteorological data with periodic boundary conditions.

Definition at line 940 of file libtrac.c.

```
00941                {
00942
00943   int ip, iy;
00944
00945   /* Check longitudes... */
00946   if (!(fabs(met->lon[met->nx - 1] - met->lon[0]
00947              + met->lon[1] - met->lon[0] - 360) < 0.01))
00948     return;
00949
00950   /* Increase longitude counter... */
00951   if ((++met->nx) > EX)
00952     ERRMSG("Cannot create periodic boundary conditions!");
00953
00954   /* Set longitude... */
00955   met->lon[met->nx - 1] = met->lon[met->nx - 2] + met->lon[1] - met->
      lon[0];
00956
00957   /* Loop over latitudes and pressure levels... */
00958   for (iy = 0; iy < met->ny; iy++)
00959     for (ip = 0; ip < met->np; ip++) {
00960       met->ps[met->nx - 1][iy] = met->ps[0][iy];
00961       met->u[met->nx - 1][iy][ip] = met->u[0][iy][ip];
00962       met->v[met->nx - 1][iy][ip] = met->v[0][iy][ip];
00963       met->w[met->nx - 1][iy][ip] = met->w[0][iy][ip];
00964       met->t[met->nx - 1][iy][ip] = met->t[0][iy][ip];
00965       met->h2o[met->nx - 1][iy][ip] = met->h2o[0][iy][ip];
00966       met->o3[met->nx - 1][iy][ip] = met->o3[0][iy][ip];
00967     }
00968 }
```

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

Read a control parameter from file or command line.

Definition at line 972 of file libtrac.c.

```
00979                {
00980
00981   FILE *in = NULL;
00982
00983   char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
00984     msg[LEN], rvarname[LEN], rval[LEN];
00985
00986   int contain = 0, i;
00987
00988   /* Open file... */
00989   if (filename[strlen(filename) - 1] != '-')
00990     if (!(in = fopen(filename, "r")))
00991       ERRMSG("Cannot open file!");
00992
00993   /* Set full variable name... */
00994   if (arridx >= 0) {
00995     sprintf(fullname1, "%s[%d]", varname, arridx);
00996     sprintf(fullname2, "%s[*]", varname);
00997   } else {
00998     sprintf(fullname1, "%s", varname);
00999     sprintf(fullname2, "%s", varname);
01000   }
01001
01002   /* Read data... */
01003   if (in != NULL)
01004     while (fgets(line, LEN, in))
01005       if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
01006         if (strcasecmp(rvarname, fullname1) == 0 ||
01007             strcasecmp(rvarname, fullname2) == 0) {
01008           contain = 1;
01009           break;
01010         }
01011   for (i = 1; i < argc - 1; i++)
```

```
01012     if (strcasecmp(argv[i], fullname1) == 0 ||
01013         strcasecmp(argv[i], fullname2) == 0) {
01014       sprintf(rval, "%s", argv[i + 1]);
01015       contain = 1;
01016       break;
01017     }
01018
01019   /* Close file... */
01020   if (in != NULL)
01021     fclose(in);
01022
01023   /* Check for missing variables... */
01024   if (!contain) {
01025     if (strlen(defvalue) > 0)
01026       sprintf(rval, "%s", defvalue);
01027     else {
01028       sprintf(msg, "Missing variable %s!\n", fullname1);
01029       ERRMSG(msg);
01030     }
01031   }
01032
01033   /* Write info... */
01034   printf("%s = %s\n", fullname1, rval);
01035
01036   /* Return values... */
01037   if (value != NULL)
01038     sprintf(value, "%s", rval);
01039   return atof(rval);
01040 }
```

5.13.2.25 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 1044 of file libtrac.c.

```
01052                 {
01053
01054   struct tm t0, t1;
01055
01056   t0.tm_year = 100;
01057   t0.tm_mon = 0;
01058   t0.tm_mday = 1;
01059   t0.tm_hour = 0;
01060   t0.tm_min = 0;
01061   t0.tm_sec = 0;
01062
01063   t1.tm_year = year - 1900;
01064   t1.tm_mon = mon - 1;
01065   t1.tm_mday = day;
01066   t1.tm_hour = hour;
01067   t1.tm_min = min;
01068   t1.tm_sec = sec;
01069
01070   *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
01071 }
```

5.13.2.26 void timer (const char * *name*, int *id*, int *mode*)

Measure wall-clock time.

Definition at line 1075 of file libtrac.c.

```
01078             {
01079
01080   static double starttime[NTIMER], runtime[NTIMER];
01081
01082   /* Check id... */
01083   if (id < 0 || id >= NTIMER)
01084     ERRMSG("Too many timers!");
01085
01086   /* Start timer... */
01087   if (mode == 1) {
01088     if (starttime[id] <= 0)
```

```
01089       starttime[id] = omp_get_wtime();
01090     else
01091       ERRMSG("Timer already started!");
01092   }
01093
01094   /* Stop timer... */
01095   else if (mode == 2) {
01096     if (starttime[id] > 0) {
01097       runtime[id] = runtime[id] + omp_get_wtime() - starttime[id];
01098       starttime[id] = -1;
01099     } else
01100       ERRMSG("Timer not started!");
01101   }
01102
01103   /* Print timer... */
01104   else if (mode == 3)
01105     printf("%s = %g s\n", name, runtime[id]);
01106 }
```

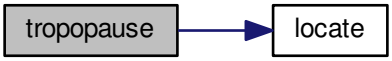
5.13.2.27 double tropopause (double *t*, double *lat*)

Definition at line 1110 of file libtrac.c.

```
01112               {
01113
01114   static double doys[12]
01115   = { 1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 };
01116
01117   static double lats[73]
01118     = { -90, -87.5, -85, -82.5, -80, -77.5, -75, -72.5, -70, -67.5,
01119     -65, -62.5, -60, -57.5, -55, -52.5, -50, -47.5, -45, -42.5,
01120     -40, -37.5, -35, -32.5, -30, -27.5, -25, -22.5, -20, -17.5,
01121     -15, -12.5, -10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5, 10, 12.5,
01122     15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40, 42.5,
01123     45, 47.5, 50, 52.5, 55, 57.5, 60, 62.5, 65, 67.5, 70, 72.5,
01124     75, 77.5, 80, 82.5, 85, 87.5, 90
01125   };
01126
01127   static double tps[12][73]
01128     = { {324.1, 325.6, 325, 324.3, 322.5, 319.7, 314, 307.2, 301.8, 299.6,
01129          297.1, 292.2, 285.6, 276.1, 264, 248.9, 231.9, 213.5, 194.4,
01130          175.3, 157, 140.4, 126.7, 116.3, 109.5, 105.4, 103, 101.4, 100.4,
01131          99.69, 99.19, 98.84, 98.56, 98.39, 98.39, 98.42, 98.44, 98.54,
01132          98.68, 98.81, 98.89, 98.96, 99.12, 99.65, 101.4, 105.4, 113.5, 128,
01133          152.1, 184.7, 214, 234.1, 247.3, 255.8, 262.6, 267.7, 271.7, 275,
01134          277.2, 279, 280.1, 280.4, 280.6, 280.1, 279.3, 278.3, 276.8, 275.8,
01135          275.3, 275.6, 275.4, 274.1, 273.5},
01136   {337.3, 338.7, 337.8, 336.4, 333, 328.8, 321.1, 312.6, 306.6, 303.7,
01137    300.2, 293.8, 285.4, 273.8, 259.6, 242.7, 224.4, 205.2, 186, 167.5,
01138    150.3, 135, 122.8, 113.9, 108.2, 104.7, 102.5, 101.1, 100.2, 99.42,
01139    98.88, 98.52, 98.25, 98.09, 98.07, 98.1, 98.12, 98.2, 98.25, 98.27,
01140    98.26, 98.27, 98.36, 98.79, 100.2, 104.2, 113.7, 131.2, 159.5, 193,
01141    220.4, 238.1, 250.2, 258.1, 264.7, 269.7, 273.7, 277.3, 280.2, 282.8,
01142    284.9, 286.5, 288.1, 288.8, 289, 288.5, 287.2, 286.3, 286.1, 287.2,
01143    287.5, 286.2, 285.8},
01144   {335, 336, 335.7, 335.1, 332.3, 328.1, 320.6, 311.8, 305.1, 301.9,
01145    297.6, 290, 280.4, 268.3, 254.6, 239.6, 223.9, 207.9, 192.2, 176.9,
01146    161.7, 146.4, 132.2, 120.6, 112.3, 107.2, 104.3, 102.4, 101.3,
01147    100.4, 99.86, 99.47, 99.16, 98.97, 98.94, 98.97, 99, 99.09, 99.2,
01148    99.31, 99.35, 99.41, 99.51, 99.86, 101.1, 104.9, 114.3, 131, 156.8,
01149    186.3, 209.3, 224.6, 236.8, 246.3, 254.9, 262.3, 268.8, 274.8,
01150    279.9, 284.6, 288.6, 291.6, 294.9, 297.5, 299.8, 301.8, 303.1,
01151    304.3, 304.9, 306, 306.6, 306.2, 306},
01152   {306.2, 306.7, 305.7, 307.1, 307.3, 306.4, 301.8, 296.2, 292.4,
01153    290.3, 287.1, 280.9, 273.4, 264.3, 254.1, 242.8, 231, 219, 207.2,
01154    195.5, 183.3, 169.7, 154.7, 138.7, 124.1, 113.6, 107.8, 104.7,
01155    102.8, 101.7, 100.9, 100.4, 100, 99.79, 99.7, 99.66, 99.68, 99.79,
01156    99.94, 100.2, 100.5, 100.9, 101.4, 102.1, 103.4, 107, 115.2, 129.1,
01157    148.7, 171, 190.8, 205.6, 218.4, 229.4, 239.6, 248.6, 256.5,
01158    263.7, 270.3, 276.6, 282.6, 288.1, 294.5, 300.4, 306.3, 311.4,
01159    315.1, 318.3, 320.3, 322.2, 322.8, 321.5, 321.1},
01160   {266.5, 264.9, 260.8, 261, 262, 263, 261.3, 259.7, 259.2, 259.8,
01161    260.1, 258.6, 256.7, 253.6, 249.5, 243.9, 237.4, 230, 222.1, 213.9,
01162    205, 194.4, 180.4, 161.8, 140.7, 122.9, 112.1, 106.7, 104.1, 102.7,
01163    101.8, 101.4, 101.1, 101, 101, 101, 101.1, 101.2, 101.5, 101.9,
01164    102.4, 103, 103.8, 104.9, 106.8, 110.1, 115.6, 124, 135.2, 148.9,
01165    165.2, 181.3, 198, 211.8, 223.5, 233.8, 242.9, 251.5, 259, 266.2,
01166    273.1, 279.2, 286.2, 292.8, 299.6, 306, 311.1, 315.5, 318.8, 322.6,
01167    325.3, 325.8, 325.8},
01168   {220.1, 218.1, 210.8, 207.2, 207.6, 210.5, 211.4, 213.5, 217.3,
01169    222.4, 227.9, 232.8, 237.4, 240.8, 242.8, 243, 241.5, 238.6, 234.2,
```

```
01170     228.5, 221, 210.7, 195.1, 172.9, 147.8, 127.6, 115.6, 109.9, 107.1,
01171     105.7, 105, 104.8, 104.8, 104.9, 105, 105.1, 105.3, 105.5, 105.8,
01172     106.4, 107, 107.6, 108.1, 108.8, 110, 111.8, 114.2, 117.4, 121.6,
01173     127.9, 137.3, 151.2, 169.5, 189, 205.8, 218.9, 229.1, 237.8, 245,
01174     251.5, 257.1, 262.3, 268.2, 274, 280.4, 286.7, 292.4, 297.9, 302.9,
01175     308.5, 312.2, 313.1, 313.3},
01176    {187.4, 184.5, 173.3, 166.1, 165.4, 167.8, 169.6, 173.6, 179.6,
01177     187.9, 198.9, 210, 220.5, 229.2, 235.7, 239.9, 241.8, 241.6, 239.6,
01178     235.8, 229.4, 218.6, 200.9, 175.9, 149.4, 129.4, 118.3, 113.1,
01179     110.8, 109.7, 109.3, 109.4, 109.7, 110, 110.2, 110.4, 110.5, 110.7,
01180     111, 111.4, 111.8, 112.1, 112.3, 112.7, 113.2, 113.9, 115, 116.4,
01181     117.9, 120.4, 124.1, 130.9, 142.2, 159.6, 179.6, 198.5, 212.9,
01182     224.2, 232.7, 239.1, 243.8, 247.7, 252.4, 257.3, 263.2, 269.5,
01183     275.4, 281.1, 286.3, 292, 296.3, 298.2, 298.8},
01184    {166, 166.4, 155.7, 148.3, 147.1, 149, 152.1, 157, 163.6, 172.4,
01185     185.3, 199.2, 212.6, 224, 233.2, 239.6, 243.3, 244.6, 243.6, 240.3,
01186     233.9, 222.6, 203.7, 177, 149.5, 129.7, 119, 114, 111.7, 110.7,
01187     110.3, 110.3, 110.6, 110.9, 111.1, 111.3, 111.5, 111.6, 111.9,
01188     112.2, 112.5, 112.6, 112.8, 113, 113.4, 114, 115.1, 116.5, 118.3,
01189     120.9, 124.4, 130.2, 139.4, 154.6, 173.8, 193.1, 208.1, 220.4,
01190     230.1, 238.2, 244.7, 249.5, 254.5, 259.3, 264.5, 269.4, 273.7,
01191     278.2, 282.6, 287.4, 290.9, 292.5, 293},
01192    {171.9, 172.8, 166.2, 162.3, 161.4, 162.5, 165.2, 169.6, 175.3,
01193     183.1, 193.8, 205.9, 218.3, 229.6, 238.5, 244.3, 246.9, 246.7,
01194     243.8, 238.4, 230.2, 217.9, 199.6, 174.9, 148.9, 129.8, 119.5,
01195     114.8, 112.3, 110.9, 110.3, 110.1, 110.2, 110.3, 110.4, 110.5,
01196     110.6, 110.8, 111, 111.4, 111.8, 112, 112.2, 112.4, 112.9, 113.6,
01197     114.7, 116.3, 118.4, 121.9, 127.1, 136.1, 149.8, 168.4, 186.9,
01198     203.3, 217, 229.1, 238.7, 247, 254, 259.3, 264.3, 268.3, 272.5,
01199     276.6, 280.4, 284.4, 288.4, 293.3, 297.2, 298.7, 299.1},
01200    {191.6, 192.2, 189, 188.1, 190.2, 193.7, 197.8, 202.9, 208.5,
01201     215.6, 224.2, 233.1, 241.2, 247.3, 250.8, 251.3, 248.9, 244.2,
01202     237.3, 228.4, 217.2, 202.9, 184.5, 162.5, 140.7, 124.8, 116.2,
01203     111.8, 109.4, 107.9, 107, 106.7, 106.6, 106.6, 106.7, 106.7,
01204     106.8, 107, 107.4, 108, 108.7, 109.3, 109.8, 110.4, 111.2,
01205     112.4, 114.2, 116.9, 121.1, 127.9, 139.3, 155.2, 173.6, 190.7,
01206     206.1, 220.1, 232.3, 243, 251.8, 259.2, 265.7, 270.6, 275.3,
01207     279.3, 283.3, 286.9, 289.7, 292.8, 296.1, 300.5, 303.9, 304.8,
01208     305.1},
01209    {241.5, 239.6, 236.8, 237.4, 239.4, 242.3, 244.2, 246.4, 249.2,
01210     253.6, 258.6, 262.7, 264.8, 264.2, 260.6, 254.1, 245.5, 235.3,
01211     223.9, 211.7, 198.3, 183.1, 165.6, 147.1, 130.5, 118.7, 111.9,
01212     108.1, 105.8, 104.3, 103.4, 102.8, 102.5, 102.4, 102.5, 102.5,
01213     102.5, 102.7, 103.1, 103.8, 104.6, 105.4, 106.1, 107, 108.2,
01214     109.9, 112.8, 117.5, 126, 140.4, 161, 181.9, 201.2, 216.8, 230.4,
01215     241.8, 251.4, 259.9, 266.9, 272.8, 277.4, 280.4, 282.9, 284.6,
01216     286.1, 287.4, 288.3, 289.5, 290.9, 294.2, 296.9, 297.5, 297.6},
01217    {301.2, 300.3, 296.6, 295.4, 295, 294.3, 291.2, 287.4, 284.9, 284.7,
01218     284.1, 281.5, 277.1, 270.4, 261.7, 250.6, 237.6, 223.1, 207.9, 192,
01219     175.8, 158.8, 142.1, 127.6, 116.8, 109.9, 106, 103.6, 102.1, 101.1,
01220     100.4, 99.96, 99.6, 99.37, 99.32, 99.32, 99.31, 99.46, 99.77, 100.2,
01221     100.7, 101.3, 101.8, 102.7, 104.1, 106.8, 111.9, 121, 136.7, 160,
01222     186.9, 209.9, 228.1, 241.2, 251.5, 259.5, 265.7, 270.9, 274.8, 278,
01223     280.3, 281.8, 283, 283.3, 283.7, 283.8, 283, 282.2, 281.2, 281.4,
01224     281.7, 281.1, 281.2}
01225    };
01226
01227    double doy, p0, p1, pt;
01228
01229    int imon, ilat;
01230
01231    /* Get day of year... */
01232    doy = fmod(t / 86400., 365.25);
01233    while (doy < 0)
01234      doy += 365.25;
01235
01236    /* Get indices... */
01237    imon = locate(doys, 12, doy);
01238    ilat = locate(lats, 73, lat);
01239
01240    /* Get tropopause pressure... */
01241    p0 = LIN(lats[ilat], tps[imon][ilat],
01242             lats[ilat + 1], tps[imon][ilat + 1], lat);
01243    p1 = LIN(lats[ilat], tps[imon + 1][ilat],
01244             lats[ilat + 1], tps[imon + 1][ilat + 1], lat);
01245    pt = LIN(doys[imon], p0, doys[imon + 1], p1, doy);
01246
01247    /* Return tropopause pressure... */
01248    return pt;
01249 }
```

Here is the call graph for this function:



5.13.2.28 void write_atm (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write atmospheric data.

Definition at line 1253 of file libtrac.c.

```
01257            {
01258
01259   FILE *in, *out;
01260
01261   char line[LEN];
01262
01263   double r, t0, t1;
01264
01265   int ip, iq, year, mon, day, hour, min, sec;
01266
01267   /* Set time interval for output... */
01268   t0 = t - 0.5 * ctl->dt_mod;
01269   t1 = t + 0.5 * ctl->dt_mod;
01270
01271   /* Check if gnuplot output is requested... */
01272   if (ctl->atm_gpfile[0] != '-') {
01273
01274     /* Write info... */
01275     printf("Plot atmospheric data: %s.png\n", filename);
01276
01277     /* Create gnuplot pipe... */
01278     if (!(out = popen("gnuplot", "w")))
01279       ERRMSG("Cannot create pipe to gnuplot!");
01280
01281     /* Set plot filename... */
01282     fprintf(out, "set out \"%s.png\"\n", filename);
01283
01284     /* Set time string... */
01285     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01286     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01287             year, mon, day, hour, min);
01288
01289     /* Dump gnuplot file to pipe... */
01290     if (!(in = fopen(ctl->atm_gpfile, "r")))
01291       ERRMSG("Cannot open file!");
01292     while (fgets(line, LEN, in))
01293       fprintf(out, "%s", line);
01294     fclose(in);
01295   }
01296
01297   else {
01298
01299     /* Write info... */
01300     printf("Write atmospheric data: %s\n", filename);
01301
01302     /* Create file... */
01303     if (!(out = fopen(filename, "w")))
01304       ERRMSG("Cannot create file!");
01305   }
01306
01307   /* Write header... */
01308   fprintf(out,
01309           "# $1 = time [s]\n"
01310           "# $2 = altitude [km]\n"
01311           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01312   for (iq = 0; iq < ctl->nq; iq++)
01313     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl->qnt_name[iq],
```

```
01314             ctl->qnt_unit[iq]);
01315   fprintf(out, "\n");
01316
01317   /* Write data... */
01318   for (ip = 0; ip < atm->np; ip++) {
01319
01320     /* Check time... */
01321     if (ctl->atm_filter && (atm->time[ip] < t0 || atm->time[ip] > t1))
01322       continue;
01323
01324     /* Write output... */
01325     fprintf(out, "%.2f %g %g %g", atm->time[ip], Z(atm->p[ip]),
01326             atm->lon[ip], atm->lat[ip]);
01327     for (iq = 0; iq < ctl->nq; iq++) {
01328       fprintf(out, " ");
01329       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
01330     }
01331     fprintf(out, "\n");
01332   }
01333
01334   /* Close file... */
01335   fclose(out);
01336 }
```

Here is the call graph for this function:

5.13.2.29 void write_csi (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write CSI data.

Definition at line 1340 of file libtrac.c.

```
01344             {
01345
01346   static FILE *in, *out;
01347
01348   static char line[LEN];
01349
01350   static double modmean[GX][GY][GZ], obsmean[GX][GY][GZ],
01351     rt, rz, rlon, rlat, robs, t0, t1, area, dlon, dlat, lat;
01352
01353   static int init, obscount[GX][GY][GZ], cx, cy, cz, ip, ix, iy, iz;
01354
01355   /* Init... */
01356   if (!init) {
01357     init = 1;
01358
01359     /* Check quantity index for mass... */
01360     if (ctl->qnt_m < 0)
01361       ERRMSG("Need quantity mass to analyze CSI!");
01362
01363     /* Open observation data file... */
01364     printf("Read CSI observation data: %s\n", ctl->csi_obsfile);
01365     if (!(in = fopen(ctl->csi_obsfile, "r")))
01366       ERRMSG("Cannot open file!");
01367
01368     /* Create new file... */
01369     printf("Write CSI data: %s\n", filename);
01370     if (!(out = fopen(filename, "w")))
01371       ERRMSG("Cannot create file!");
01372
01373     /* Write header... */
```

```
01374     fprintf(out,
01375             "# $1 = time [s]\n"
01376             "# $2 = number of hits (cx)\n"
01377             "# $3 = number of misses (cy)\n"
01378             "# $4 = number of false alarms (cz)\n"
01379             "# $5 = number of observations (cx + cy)\n"
01380             "# $6 = number of forecasts (cx + cz)\n"
01381             "# $7 = bias (forecasts/observations) [%%]\n"
01382             "# $8 = probability of detection (POD) [%%]\n"
01383             "# $9 = false alarm rate (FAR) [%%]\n"
01384             "# $10 = critical success index (CSI) [%%]\n\n");
01385   }
01386
01387   /* Set time interval... */
01388   t0 = t - 0.5 * ctl->dt_mod;
01389   t1 = t + 0.5 * ctl->dt_mod;
01390
01391   /* Initialize grid cells... */
01392   for (ix = 0; ix < ctl->csi_nx; ix++)
01393     for (iy = 0; iy < ctl->csi_ny; iy++)
01394       for (iz = 0; iz < ctl->csi_nz; iz++)
01395         modmean[ix][iy][iz] = obsmean[ix][iy][iz] = obscount[ix][iy][iz] = 0;
01396
01397   /* Read data... */
01398   while (fgets(line, LEN, in)) {
01399
01400     /* Read data... */
01401     if (sscanf(line, "%lg %lg %lg %lg %lg", &rt, &rz, &rlon, &rlat, &robs) !=
01402         5)
01403       continue;
01404
01405     /* Check time... */
01406     if (rt < t0)
01407       continue;
01408     if (rt > t1)
01409       break;
01410
01411     /* Calculate indices... */
01412     ix = (int) ((rlon - ctl->csi_lon0)
01413                 / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01414     iy = (int) ((rlat - ctl->csi_lat0)
01415                 / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01416     iz = (int) ((rz - ctl->csi_z0)
01417                 / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01418
01419     /* Check indices... */
01420     if (ix < 0 || ix >= ctl->csi_nx ||
01421         iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01422       continue;
01423
01424     /* Get mean observation index... */
01425     obsmean[ix][iy][iz] += robs;
01426     obscount[ix][iy][iz]++;
01427   }
01428
01429   /* Analyze model data... */
01430   for (ip = 0; ip < atm->np; ip++) {
01431
01432     /* Check time... */
01433     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01434       continue;
01435
01436     /* Get indices... */
01437     ix = (int) ((atm->lon[ip] - ctl->csi_lon0)
01438                 / (ctl->csi_lon1 - ctl->csi_lon0) * ctl->csi_nx);
01439     iy = (int) ((atm->lat[ip] - ctl->csi_lat0)
01440                 / (ctl->csi_lat1 - ctl->csi_lat0) * ctl->csi_ny);
01441     iz = (int) ((Z(atm->p[ip]) - ctl->csi_z0)
01442                 / (ctl->csi_z1 - ctl->csi_z0) * ctl->csi_nz);
01443
01444     /* Check indices... */
01445     if (ix < 0 || ix >= ctl->csi_nx ||
01446         iy < 0 || iy >= ctl->csi_ny || iz < 0 || iz >= ctl->csi_nz)
01447       continue;
01448
01449     /* Get total mass in grid cell... */
01450     modmean[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01451   }
01452
01453   /* Analyze all grid cells... */
01454   for (ix = 0; ix < ctl->csi_nx; ix++)
01455     for (iy = 0; iy < ctl->csi_ny; iy++)
01456       for (iz = 0; iz < ctl->csi_nz; iz++) {
01457
01458         /* Calculate mean observation index... */
01459         if (obscount[ix][iy][iz] > 0)
01460           obsmean[ix][iy][iz] /= obscount[ix][iy][iz];
```

```
01461
01462         /* Calculate column density... */
01463         if (modmean[ix][iy][iz] > 0) {
01464           dlon = (ctl->csi_lon1 - ctl->csi_lon0) / ctl->csi_nx;
01465           dlat = (ctl->csi_lat1 - ctl->csi_lat0) / ctl->csi_ny;
01466           lat = ctl->csi_lat0 + dlat * (iy + 0.5);
01467           area = dlat * M_PI * RE / 180. * dlon * M_PI * RE / 180.
01468             * cos(lat * M_PI / 180.);
01469           modmean[ix][iy][iz] /= (1e6 * area);
01470         }
01471
01472         /* Calculate CSI... */
01473         if (obscount[ix][iy][iz] > 0) {
01474           if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01475               modmean[ix][iy][iz] >= ctl->csi_modmin)
01476             cx++;
01477           else if (obsmean[ix][iy][iz] >= ctl->csi_obsmin &&
01478                    modmean[ix][iy][iz] < ctl->csi_modmin)
01479             cy++;
01480           else if (obsmean[ix][iy][iz] < ctl->csi_obsmin &&
01481                    modmean[ix][iy][iz] >= ctl->csi_modmin)
01482             cz++;
01483         }
01484       }
01485
01486   /* Write output... */
01487   if (fmod(t, ctl->csi_dt_out) == 0) {
01488
01489     /* Write... */
01490     fprintf(out, "%.2f %d %d %d %d %d %g %g %g %g\n",
01491             t, cx, cy, cz, cx + cy, cx + cz,
01492             (cx + cy > 0) ? 100. * (cx + cz) / (cx + cy) : GSL_NAN,
01493             (cx + cy > 0) ? (100. * cx) / (cx + cy) : GSL_NAN,
01494             (cx + cz > 0) ? (100. * cz) / (cx + cz) : GSL_NAN,
01495             (cx + cy + cz > 0) ? (100. * cx) / (cx + cy + cz) : GSL_NAN);
01496
01497     /* Set counters to zero... */
01498     cx = cy = cz = 0;
01499   }
01500
01501   /* Close file... */
01502   if (t == ctl->t_stop)
01503     fclose(out);
01504 }
```

5.13.2.30 void write_ens (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

Write ensemble data.

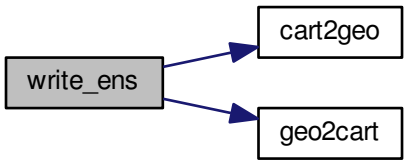
Definition at line 1508 of file libtrac.c.

```
01512             {
01513
01514   static FILE *out;
01515
01516   static double dummy, ens, lat, lon, p[NENS], q[NQ][NENS],
01517     t0, t1, x[NENS][3], xm[3];
01518
01519   static int init, ip, iq;
01520
01521   static size_t i, n;
01522
01523   /* Init... */
01524   if (!init) {
01525     init = 1;
01526
01527     /* Check quantities... */
01528     if (ctl->qnt_ens < 0)
01529       ERRMSG("Missing ensemble IDs!");
01530
01531     /* Create new file... */
01532     printf("Write ensemble data: %s\n", filename);
01533     if (!(out = fopen(filename, "w")))
01534       ERRMSG("Cannot create file!");
01535
01536     /* Write header... */
01537     fprintf(out,
01538             "# $1 = time [s]\n"
01539             "# $2 = altitude [km]\n"
01540             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
```

```
01541     for (iq = 0; iq < ctl->nq; iq++)
01542       fprintf(out, "# $%d = %s (mean) [%s]\n", 5 + iq,
01543               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01544     for (iq = 0; iq < ctl->nq; iq++)
01545       fprintf(out, "# $%d = %s (sigma) [%s]\n", 5 + ctl->nq + iq,
01546               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01547     fprintf(out, "# $%d = number of members\n\n", 5 + 2 * ctl->nq);
01548   }
01549
01550   /* Set time interval... */
01551   t0 = t - 0.5 * ctl->dt_mod;
01552   t1 = t + 0.5 * ctl->dt_mod;
01553
01554   /* Init... */
01555   ens = GSL_NAN;
01556   n = 0;
01557
01558   /* Loop over air parcels... */
01559   for (ip = 0; ip < atm->np; ip++) {
01560
01561     /* Check time... */
01562     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01563       continue;
01564
01565     /* Check ensemble id... */
01566     if (atm->q[ctl->qnt_ens][ip] != ens) {
01567
01568       /* Write results... */
01569       if (n > 0) {
01570
01571         /* Get mean position... */
01572         xm[0] = xm[1] = xm[2] = 0;
01573         for (i = 0; i < n; i++) {
01574           xm[0] += x[i][0] / (double) n;
01575           xm[1] += x[i][1] / (double) n;
01576           xm[2] += x[i][2] / (double) n;
01577         }
01578         cart2geo(xm, &dummy, &lon, &lat);
01579         fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon,
01580                 lat);
01581
01582         /* Get quantity statistics... */
01583         for (iq = 0; iq < ctl->nq; iq++) {
01584           fprintf(out, " ");
01585           fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01586         }
01587         for (iq = 0; iq < ctl->nq; iq++) {
01588           fprintf(out, " ");
01589           fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01590         }
01591         fprintf(out, " %lu\n", n);
01592       }
01593
01594       /* Init new ensemble... */
01595       ens = atm->q[ctl->qnt_ens][ip];
01596       n = 0;
01597     }
01598
01599     /* Save data... */
01600     p[n] = atm->p[ip];
01601     geo2cart(0, atm->lon[ip], atm->lat[ip], x[n]);
01602     for (iq = 0; iq < ctl->nq; iq++)
01603       q[iq][n] = atm->q[iq][ip];
01604     if ((++n) >= NENS)
01605       ERRMSG("Too many data points!");
01606   }
01607
01608   /* Write results... */
01609   if (n > 0) {
01610
01611     /* Get mean position... */
01612     xm[0] = xm[1] = xm[2] = 0;
01613     for (i = 0; i < n; i++) {
01614       xm[0] += x[i][0] / (double) n;
01615       xm[1] += x[i][1] / (double) n;
01616       xm[2] += x[i][2] / (double) n;
01617     }
01618     cart2geo(xm, &dummy, &lon, &lat);
01619     fprintf(out, "%.2f %g %g %g", t, Z(gsl_stats_mean(p, 1, n)), lon, lat);
01620
01621     /* Get quantity statistics... */
01622     for (iq = 0; iq < ctl->nq; iq++) {
01623       fprintf(out, " ");
01624       fprintf(out, ctl->qnt_format[iq], gsl_stats_mean(q[iq], 1, n));
01625     }
01626     for (iq = 0; iq < ctl->nq; iq++) {
01627       fprintf(out, " ");
```

```
01628       fprintf(out, ctl->qnt_format[iq], gsl_stats_sd(q[iq], 1, n));
01629     }
01630     fprintf(out, " %lu\n", n);
01631   }
01632
01633   /* Close file... */
01634   if (t == ctl->t_stop)
01635     fclose(out);
01636 }
```

Here is the call graph for this function:



5.13.2.31 void write_grid (const char * *filename*, ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, double *t*)

Write gridded data.

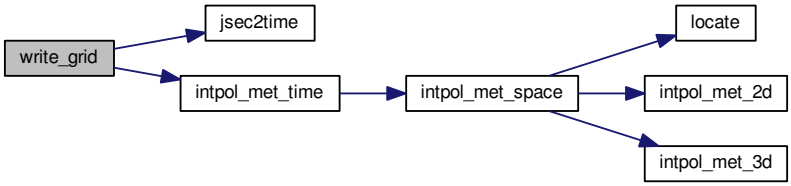
Definition at line 1640 of file libtrac.c.

```
01646             {
01647
01648   FILE *in, *out;
01649
01650   char line[LEN];
01651
01652   static double grid_m[GX][GY][GZ], z, dz, lon, dlon, lat, dlat,
01653     area, rho_air, press, temp, cd, mmr, t0, t1, r;
01654
01655   static int ip, ix, iy, iz, year, mon, day, hour, min, sec;
01656
01657   /* Check dimensions... */
01658   if (ctl->grid_nx > GX || ctl->grid_ny > GY || ctl->grid_nz > GZ)
01659     ERRMSG("Grid dimensions too large!");
01660
01661   /* Check quantity index for mass... */
01662   if (ctl->qnt_m < 0)
01663     ERRMSG("Need quantity mass to write grid data!");
01664
01665   /* Set time interval for output... */
01666   t0 = t - 0.5 * ctl->dt_mod;
01667   t1 = t + 0.5 * ctl->dt_mod;
01668
01669   /* Set grid box size... */
01670   dz = (ctl->grid_z1 - ctl->grid_z0) / ctl->grid_nz;
01671   dlon = (ctl->grid_lon1 - ctl->grid_lon0) / ctl->grid_nx;
01672   dlat = (ctl->grid_lat1 - ctl->grid_lat0) / ctl->grid_ny;
01673
01674   /* Initialize grid... */
01675   for (ix = 0; ix < ctl->grid_nx; ix++)
01676     for (iy = 0; iy < ctl->grid_ny; iy++)
01677       for (iz = 0; iz < ctl->grid_nz; iz++)
01678         grid_m[ix][iy][iz] = 0;
01679
01680   /* Average data... */
01681   for (ip = 0; ip < atm->np; ip++)
01682     if (atm->time[ip] >= t0 && atm->time[ip] <= t1) {
01683
01684       /* Get index... */
01685       ix = (int) ((atm->lon[ip] - ctl->grid_lon0) / dlon);
```

```
01686       iy = (int) ((atm->lat[ip] - ctl->grid_lat0) / dlat);
01687       iz = (int) ((Z(atm->p[ip]) - ctl->grid_z0) / dz);
01688
01689       /* Check indices... */
01690       if (ix < 0 || ix >= ctl->grid_nx ||
01691           iy < 0 || iy >= ctl->grid_ny || iz < 0 || iz >= ctl->grid_nz)
01692         continue;
01693
01694       /* Add mass... */
01695       grid_m[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01696     }
01697
01698   /* Check if gnuplot output is requested... */
01699   if (ctl->grid_gpfile[0] != '-') {
01700
01701     /* Write info... */
01702     printf("Plot grid data: %s.png\n", filename);
01703
01704     /* Create gnuplot pipe... */
01705     if (!(out = popen("gnuplot", "w")))
01706       ERRMSG("Cannot create pipe to gnuplot!");
01707
01708     /* Set plot filename... */
01709     fprintf(out, "set out \"%s.png\"\n", filename);
01710
01711     /* Set time string... */
01712     jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01713     fprintf(out, "timestr=\"%d-%02d-%02d, %02d:%02d UTC\"\n",
01714             year, mon, day, hour, min);
01715
01716     /* Dump gnuplot file to pipe... */
01717     if (!(in = fopen(ctl->grid_gpfile, "r")))
01718       ERRMSG("Cannot open file!");
01719     while (fgets(line, LEN, in))
01720       fprintf(out, "%s", line);
01721     fclose(in);
01722   }
01723
01724   else {
01725
01726     /* Write info... */
01727     printf("Write grid data: %s\n", filename);
01728
01729     /* Create file... */
01730     if (!(out = fopen(filename, "w")))
01731       ERRMSG("Cannot create file!");
01732   }
01733
01734   /* Write header... */
01735   fprintf(out,
01736           "# $1 = time [s]\n"
01737           "# $2 = altitude [km]\n"
01738           "# $3 = longitude [deg]\n"
01739           "# $4 = latitude [deg]\n"
01740           "# $5 = surface area [km^2]\n"
01741           "# $6 = layer width [km]\n"
01742           "# $7 = temperature [K]\n"
01743           "# $8 = column density [kg/m^2]\n"
01744           "# $9 = mass mixing ratio [1]\n\n");
01745
01746   /* Write data... */
01747   for (ix = 0; ix < ctl->grid_nx; ix++) {
01748     if (ix > 0 && ctl->grid_ny > 1 && !ctl->grid_sparse)
01749       fprintf(out, "\n");
01750     for (iy = 0; iy < ctl->grid_ny; iy++) {
01751       if (iy > 0 && ctl->grid_nz > 1 && !ctl->grid_sparse)
01752         fprintf(out, "\n");
01753       for (iz = 0; iz < ctl->grid_nz; iz++)
01754         if (!ctl->grid_sparse
01755             || ix == 0 || iy == 0 || iz == 0 || grid_m[ix][iy][iz] > 0) {
01756
01757           /* Set coordinates... */
01758           z = ctl->grid_z0 + dz * (iz + 0.5);
01759           lon = ctl->grid_lon0 + dlon * (ix + 0.5);
01760           lat = ctl->grid_lat0 + dlat * (iy + 0.5);
01761
01762           /* Get pressure and temperature... */
01763           press = P(z);
01764           intpol_met_time(met0, met1, t, press, lon, lat,
01765                           NULL, &temp, NULL, NULL, NULL, NULL, NULL);
01766
01767           /* Calculate surface area... */
01768           area = dlat * dlon * gsl_pow_2(RE * M_PI / 180.)
01769             * cos(lat * M_PI / 180.);
01770
01771           /* Calculate column density... */
01772           cd = grid_m[ix][iy][iz] / (1e6 * area);
```

```
01773
01774           /* Calculate mass mixing ratio... */
01775           rho_air = 100. * press / (287.058 * temp);
01776           mmr = grid_m[ix][iy][iz] / (rho_air * 1e6 * area * 1e3 * dz);
01777
01778           /* Write output... */
01779           fprintf(out, "%.2f %g %g %g %g %g %g %g %g\n",
01780                   t, z, lon, lat, area, dz, temp, cd, mmr);
01781         }
01782     }
01783   }
01784
01785   /* Close file... */
01786   fclose(out);
01787 }
```

Here is the call graph for this function:



5.13.2.32 void write_prof (const char * *filename*, ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, double *t*)

Write profile data.

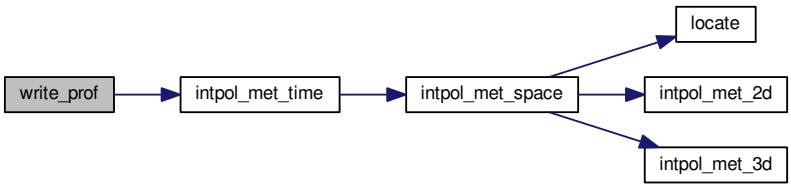
Definition at line 1791 of file libtrac.c.

```
01797           {
01798
01799   static FILE *in, *out;
01800
01801   static char line[LEN];
01802
01803   static double mass[GX][GY][GZ], obsmean[GX][GY], tmean[GX][GY],
01804     rt, rlon, rlat, robs, t0, t1, area, dz, dlon, dlat, lon, lat, z,
01805     press, temp, rho_air, mmr, h2o, o3;
01806
01807   static int init, obscount[GX][GY], ip, ix, iy, iz;
01808
01809   /* Init... */
01810   if (!init) {
01811     init = 1;
01812
01813     /* Check quantity index for mass... */
01814     if (ctl->qnt_m < 0)
01815       ERRMSG("Need quantity mass!");
01816
01817     /* Check dimensions... */
01818     if (ctl->prof_nx > GX || ctl->prof_ny > GY || ctl->prof_nz > GZ)
01819       ERRMSG("Grid dimensions too large!");
01820
01821     /* Open observation data file... */
01822     printf("Read profile observation data: %s\n", ctl->prof_obsfile);
01823     if (!(in = fopen(ctl->prof_obsfile, "r")))
01824       ERRMSG("Cannot open file!");
01825
01826     /* Create new file... */
01827     printf("Write profile data: %s\n", filename);
01828     if (!(out = fopen(filename, "w")))
01829       ERRMSG("Cannot create file!");
01830
01831     /* Write header... */
01832     fprintf(out,
```

```
01833             "# $1  = time [s]\n"
01834             "# $2  = altitude [km]\n"
01835             "# $3  = longitude [deg]\n"
01836             "# $4  = latitude [deg]\n"
01837             "# $5  = pressure [hPa]\n"
01838             "# $6  = temperature [K]\n"
01839             "# $7  = mass mixing ratio [1]\n"
01840             "# $8  = H2O volume mixing ratio [1]\n"
01841             "# $9  = O3 volume mixing ratio [1]\n"
01842             "# $10 = mean BT index [K]\n");
01843
01844     /* Set grid box size... */
01845     dz = (ctl->prof_z1 - ctl->prof_z0) / ctl->prof_nz;
01846     dlon = (ctl->prof_lon1 - ctl->prof_lon0) / ctl->prof_nx;
01847     dlat = (ctl->prof_lat1 - ctl->prof_lat0) / ctl->prof_ny;
01848   }
01849
01850   /* Set time interval... */
01851   t0 = t - 0.5 * ctl->dt_mod;
01852   t1 = t + 0.5 * ctl->dt_mod;
01853
01854   /* Initialize... */
01855   for (ix = 0; ix < ctl->prof_nx; ix++)
01856     for (iy = 0; iy < ctl->prof_ny; iy++) {
01857       obsmean[ix][iy] = 0;
01858       obscount[ix][iy] = 0;
01859       tmean[ix][iy] = 0;
01860       for (iz = 0; iz < ctl->prof_nz; iz++)
01861         mass[ix][iy][iz] = 0;
01862     }
01863
01864   /* Read data... */
01865   while (fgets(line, LEN, in)) {
01866
01867     /* Read data... */
01868     if (sscanf(line, "%lg %lg %lg %lg", &rt, &rlon, &rlat, &robs) != 4)
01869       continue;
01870
01871     /* Check time... */
01872     if (rt < t0)
01873       continue;
01874     if (rt > t1)
01875       break;
01876
01877     /* Calculate indices... */
01878     ix = (int) ((rlon - ctl->prof_lon0) / dlon);
01879     iy = (int) ((rlat - ctl->prof_lat0) / dlat);
01880
01881     /* Check indices... */
01882     if (ix < 0 || ix >= ctl->prof_nx || iy < 0 || iy >= ctl->prof_ny)
01883       continue;
01884
01885     /* Get mean observation index... */
01886     obsmean[ix][iy] += robs;
01887     tmean[ix][iy] += rt;
01888     obscount[ix][iy]++;
01889   }
01890
01891   /* Analyze model data... */
01892   for (ip = 0; ip < atm->np; ip++) {
01893
01894     /* Check time... */
01895     if (atm->time[ip] < t0 || atm->time[ip] > t1)
01896       continue;
01897
01898     /* Get indices... */
01899     ix = (int) ((atm->lon[ip] - ctl->prof_lon0) / dlon);
01900     iy = (int) ((atm->lat[ip] - ctl->prof_lat0) / dlat);
01901     iz = (int) ((Z(atm->p[ip]) - ctl->prof_z0) / dz);
01902
01903     /* Check indices... */
01904     if (ix < 0 || ix >= ctl->prof_nx ||
01905         iy < 0 || iy >= ctl->prof_ny || iz < 0 || iz >= ctl->prof_nz)
01906       continue;
01907
01908     /* Get total mass in grid cell... */
01909     mass[ix][iy][iz] += atm->q[ctl->qnt_m][ip];
01910   }
01911
01912   /* Extract profiles... */
01913   for (ix = 0; ix < ctl->prof_nx; ix++)
01914     for (iy = 0; iy < ctl->prof_ny; iy++)
01915       if (obscount[ix][iy] > 0) {
01916
01917         /* Write output... */
01918         fprintf(out, "\n");
01919
```

```
01920         /* Loop over altitudes... */
01921         for (iz = 0; iz < ctl->prof_nz; iz++) {
01922
01923           /* Set coordinates... */
01924           z = ctl->prof_z0 + dz * (iz + 0.5);
01925           lon = ctl->prof_lon0 + dlon * (ix + 0.5);
01926           lat = ctl->prof_lat0 + dlat * (iy + 0.5);
01927
01928           /* Get meteorological data... */
01929           press = P(z);
01930           intpol_met_time(met0, met1, t, press, lon, lat,
01931                           NULL, &temp, NULL, NULL, NULL, &h2o, &o3);
01932
01933           /* Calculate mass mixing ratio... */
01934           rho_air = 100. * press / (287.058 * temp);
01935           area = dlat * dlon * gsl_pow_2(M_PI * RE / 180.)
01936             * cos(lat * M_PI / 180.);
01937           mmr = mass[ix][iy][iz] / (rho_air * area * dz * 1e9);
01938
01939           /* Write output... */
01940           fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g\n",
01941                   tmean[ix][iy] / obscount[ix][iy],
01942                   z, lon, lat, press, temp, mmr, h2o, o3,
01943                   obsmean[ix][iy] / obscount[ix][iy]);
01944         }
01945       }
01946
01947   /* Close file... */
01948   if (t == ctl->t_stop)
01949     fclose(out);
01950 }
```

Here is the call graph for this function:



5.13.2.33 void write_station (const char * *filename*, ctl_t * *ctl*, atm_t * *atm*, double *t*)

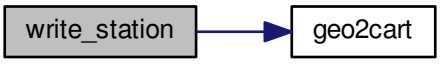
Write station data.

Definition at line 1954 of file libtrac.c.

```
01958             {
01959
01960   static FILE *out;
01961
01962   static double rmax2, t0, t1, x0[3], x1[3];
01963
01964   static int init, ip, iq;
01965
01966   /* Init... */
01967   if (!init) {
01968     init = 1;
01969
01970     /* Write info... */
01971     printf("Write station data: %s\n", filename);
01972
01973     /* Create new file... */
01974     if (!(out = fopen(filename, "w")))
01975       ERRMSG("Cannot create file!");
01976
01977     /* Write header... */
```

```
01978     fprintf(out,
01979             "# $1 = time [s]\n"
01980             "# $2 = altitude [km]\n"
01981             "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
01982     for (iq = 0; iq < ctl->nq; iq++)
01983       fprintf(out, "# $%i = %s [%s]\n", (iq + 5),
01984               ctl->qnt_name[iq], ctl->qnt_unit[iq]);
01985     fprintf(out, "\n");
01986
01987     /* Set geolocation and search radius... */
01988     geo2cart(0, ctl->stat_lon, ctl->stat_lat, x0);
01989     rmax2 = gsl_pow_2(ctl->stat_r);
01990   }
01991
01992   /* Set time interval for output... */
01993   t0 = t - 0.5 * ctl->dt_mod;
01994   t1 = t + 0.5 * ctl->dt_mod;
01995
01996   /* Loop over air parcels... */
01997   for (ip = 0; ip < atm->np; ip++) {
01998
01999     /* Check time... */
02000     if (atm->time[ip] < t0 || atm->time[ip] > t1)
02001       continue;
02002
02003     /* Check station flag... */
02004     if (ctl->qnt_stat >= 0)
02005       if (atm->q[ctl->qnt_stat][ip])
02006         continue;
02007
02008     /* Get Cartesian coordinates... */
02009     geo2cart(0, atm->lon[ip], atm->lat[ip], x1);
02010
02011     /* Check horizontal distance... */
02012     if (DIST2(x0, x1) > rmax2)
02013       continue;
02014
02015     /* Set station flag... */
02016     if (ctl->qnt_stat >= 0)
02017       atm->q[ctl->qnt_stat][ip] = 1;
02018
02019     /* Write data... */
02020     fprintf(out, "%.2f %g %g %g",
02021             atm->time[ip], Z(atm->p[ip]), atm->lon[ip], atm->lat[ip]);
02022     for (iq = 0; iq < ctl->nq; iq++) {
02023       fprintf(out, " ");
02024       fprintf(out, ctl->qnt_format[iq], atm->q[iq][ip]);
02025     }
02026     fprintf(out, "\n");
02027   }
02028
02029   /* Close file... */
02030   if (t == ctl->t_stop)
02031     fclose(out);
02032 }
```

Here is the call graph for this function:



5.14 libtrac.h

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00034 #include <ctype.h>
00035 #include <gsl/gsl_const_mksa.h>
00036 #include <gsl/gsl_math.h>
00037 #include <gsl/gsl_randist.h>
00038 #include <gsl/gsl_rng.h>
00039 #include <gsl/gsl_sort.h>
00040 #include <gsl/gsl_statistics.h>
00041 #include <math.h>
00042 #include <netcdf.h>
00043 #include <omp.h>
00044 #include <stdio.h>
00045 #include <stdlib.h>
00046 #include <string.h>
00047 #include <time.h>
00048 #include <sys/time.h>
00049
00050 /* ------------------------------------------------------------
00051    Macros...
00052    ------------------------------------------------------------ */
00053
00055 #define ALLOC(ptr, type, n)                             \
00056   if((ptr=calloc((size_t)(n), sizeof(type)))==NULL)      \
00057     ERRMSG("Out of memory!");
00058
00060 #define DIST(a, b) sqrt(DIST2(a, b))
00061
00063 #define DIST2(a, b)                                                     \
00064   ((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1])+(a[2]-b[2])*(a[2]-b[2]))
00065
00067 #define DOTP(a, b) (a[0]*b[0]+a[1]*b[1]+a[2]*b[2])
00068
00070 #define ERRMSG(msg) {                                                   \
00071     printf("\nError (%s, %s, l%d): %s\n\n",                             \
00072            __FILE__, __func__, __LINE__, msg);                          \
00073     exit(EXIT_FAILURE);                                                 \
00074   }
00075
00077 #define LIN(x0, y0, x1, y1, x)                  \
00078   ((y0)+((y1)-(y0))/((x1)-(x0))*((x)-(x0)))
00079
00081 #define NC(cmd) {                                       \
00082     if((cmd)!=NC_NOERR)                                 \
00083       ERRMSG(nc_strerror(cmd));                         \
00084   }
00085
00087 #define NORM(a) sqrt(DOTP(a, a))
00088
00090 #define PRINT(format, var)                                              \
00091   printf("Print (%s, %s, l%d): %s= "format"\n",                         \
00092          __FILE__, __func__, __LINE__, #var, var);
00093
00095 #define P(z) (P0*exp(-(z)/H0))
00096
00098 #define TOK(line, tok, format, var) {                                   \
00099     if(((tok)=strtok((line), " \t"))) {                                 \
00100       if(sscanf(tok, format, &(var))!=1) continue;                      \
00101     } else ERRMSG("Error while reading!");                              \
00102   }
00103
00105 #define Z(p) (H0*log(P0/(p)))
00106
00108 #define START_TIMER(id) timer(#id, id, 1)
00109
00111 #define STOP_TIMER(id) timer(#id, id, 2)
00112
00114 #define PRINT_TIMER(id) timer(#id, id, 3)
00115
00116 /* ------------------------------------------------------------
00117    Constants...
00118    ------------------------------------------------------------ */
00119
00121 #define G0 9.80665
00122
```

```
00124 #define H0 7.0
00125
00127 #define P0 1013.25
00128
00130 #define RE 6367.421
00131
00132 /* ------------------------------------------------------------
00133    Dimensions...
00134    ------------------------------------------------------------ */
00135
00137 #define LEN 5000
00138
00140 #define NP 10000000
00141
00143 #define NQ 12
00144
00146 #define EP 73
00147
00149 #define EX 721
00150
00152 #define EY 361
00153
00155 #define GX 720
00156
00158 #define GY 360
00159
00161 #define GZ 100
00162
00164 #define NENS 2000
00165
00167 #define NTHREADS 128
00168
00170 #define NTIMER 20
00171
00172 /* ------------------------------------------------------------
00173    Structs...
00174    ------------------------------------------------------------ */
00175
00177 typedef struct {
00178
00180   int nq;
00181
00183   char qnt_name[NQ][LEN];
00184
00186   char qnt_unit[NQ][LEN];
00187
00189   char qnt_format[NQ][LEN];
00190
00192   int qnt_ens;
00193
00195   int qnt_m;
00196
00198   int qnt_rho;
00199
00201   int qnt_r;
00202
00204   int qnt_ps;
00205
00207   int qnt_p;
00208
00210   int qnt_t;
00211
00213   int qnt_u;
00214
00216   int qnt_v;
00217
00219   int qnt_w;
00220
00222   int qnt_h2o;
00223
00225   int qnt_o3;
00226
00228   int qnt_theta;
00229
00231   int qnt_pv;
00232
00234   int qnt_tice;
00235
00237   int qnt_tsts;
00238
00240   int qnt_tnat;
00241
00243   int qnt_stat;
00244
00246   int qnt_gw_wind;
00247
00249   int qnt_gw_sso;
```

```
00250
00252   int qnt_gw_var;
00253
00255   int direction;
00256
00258   double t_start;
00259
00261   double t_stop;
00262
00264   double dt_mod;
00265
00267   double dt_met;
00268
00270   int met_np;
00271
00273   double met_p[EP];
00274
00277   int isosurf;
00278
00280   char balloon[LEN];
00281
00283   double turb_dx_trop;
00284
00286   double turb_dx_strat;
00287
00289   double turb_dz_trop;
00290
00292   double turb_dz_strat;
00293
00295   double turb_meso;
00296
00298   double tdec_trop;
00299
00301   double tdec_strat;
00302
00304   double psc_h2o;
00305
00307   double psc_hno3;
00308
00310   char gw_basename[LEN];
00311
00313   char atm_basename[LEN];
00314
00316   char atm_gpfile[LEN];
00317
00319   double atm_dt_out;
00320
00322   int atm_filter;
00323
00325   char csi_basename[LEN];
00326
00328   double csi_dt_out;
00329
00331   char csi_obsfile[LEN];
00332
00334   double csi_obsmin;
00335
00337   double csi_modmin;
00338
00340   int csi_nz;
00341
00343   double csi_z0;
00344
00346   double csi_z1;
00347
00349   int csi_nx;
00350
00352   double csi_lon0;
00353
00355   double csi_lon1;
00356
00358   int csi_ny;
00359
00361   double csi_lat0;
00362
00364   double csi_lat1;
00365
00367   char grid_basename[LEN];
00368
00370   char grid_gpfile[LEN];
00371
00373   double grid_dt_out;
00374
00376   int grid_sparse;
00377
00379   int grid_nz;
00380
```

```
00382   double grid_z0;
00383
00385   double grid_z1;
00386
00388   int grid_nx;
00389
00391   double grid_lon0;
00392
00394   double grid_lon1;
00395
00397   int grid_ny;
00398
00400   double grid_lat0;
00401
00403   double grid_lat1;
00404
00406   char prof_basename[LEN];
00407
00409   char prof_obsfile[LEN];
00410
00412   int prof_nz;
00413
00415   double prof_z0;
00416
00418   double prof_z1;
00419
00421   int prof_nx;
00422
00424   double prof_lon0;
00425
00427   double prof_lon1;
00428
00430   int prof_ny;
00431
00433   double prof_lat0;
00434
00436   double prof_lat1;
00437
00439   char ens_basename[LEN];
00440
00442   char stat_basename[LEN];
00443
00445   double stat_lon;
00446
00448   double stat_lat;
00449
00451   double stat_r;
00452
00453 } ctl_t;
00454
00456 typedef struct {
00457
00459   int np;
00460
00462   double time[NP];
00463
00465   double p[NP];
00466
00468   double lon[NP];
00469
00471   double lat[NP];
00472
00474   double q[NQ][NP];
00475
00477   double up[NP];
00478
00480   double vp[NP];
00481
00483   double wp[NP];
00484
00485 } atm_t;
00486
00488 typedef struct {
00489
00491   double time;
00492
00494   int nx;
00495
00497   int ny;
00498
00500   int np;
00501
00503   double lon[EX];
00504
00506   double lat[EY];
00507
00509   double p[EP];
```

```
00510
00512   double ps[EX][EY];
00513
00515   float pl[EX][EY][EP];
00516
00518   float t[EX][EY][EP];
00519
00521   float u[EX][EY][EP];
00522
00524   float v[EX][EY][EP];
00525
00527   float w[EX][EY][EP];
00528
00530   float h2o[EX][EY][EP];
00531
00533   float o3[EX][EY][EP];
00534
00535 } met_t;
00536
00537 /* ------------------------------------------------------------
00538    Functions...
00539    ------------------------------------------------------------ */
00540
00542 void cart2geo(
00543   double *x,
00544   double *z,
00545   double *lon,
00546   double *lat);
00547
00549 double deg2dx(
00550   double dlon,
00551   double lat);
00552
00554 double deg2dy(
00555   double dlat);
00556
00558 double dp2dz(
00559   double dp,
00560   double p);
00561
00563 double dx2deg(
00564   double dx,
00565   double lat);
00566
00568 double dy2deg(
00569   double dy);
00570
00572 double dz2dp(
00573   double dz,
00574   double p);
00575
00577 void geo2cart(
00578   double z,
00579   double lon,
00580   double lat,
00581   double *x);
00582
00584 void get_met(
00585   ctl_t * ctl,
00586   char *metbase,
00587   double t,
00588   met_t * met0,
00589   met_t * met1);
00590
00592 void get_met_help(
00593   double t,
00594   int direct,
00595   char *metbase,
00596   double dt_met,
00597   char *filename);
00598
00600 void intpol_met_2d(
00601   double array[EX][EY],
00602   int ix,
00603   int iy,
00604   double wx,
00605   double wy,
00606   double *var);
00607
00609 void intpol_met_3d(
00610   float array[EX][EY][EP],
00611   int ip,
00612   int ix,
00613   int iy,
00614   double wp,
00615   double wx,
00616   double wy,
```

```
00617   double *var);
00618
00620 void intpol_met_space(
00621   met_t * met,
00622   double p,
00623   double lon,
00624   double lat,
00625   double *ps,
00626   double *t,
00627   double *u,
00628   double *v,
00629   double *w,
00630   double *h2o,
00631   double *o3);
00632
00634 void intpol_met_time(
00635   met_t * met0,
00636   met_t * met1,
00637   double ts,
00638   double p,
00639   double lon,
00640   double lat,
00641   double *ps,
00642   double *t,
00643   double *u,
00644   double *v,
00645   double *w,
00646   double *h2o,
00647   double *o3);
00648
00650 void jsec2time(
00651   double jsec,
00652   int *year,
00653   int *mon,
00654   int *day,
00655   int *hour,
00656   int *min,
00657   int *sec,
00658   double *remain);
00659
00661 int locate(
00662   double *xx,
00663   int n,
00664   double x);
00665
00667 void read_atm(
00668   const char *filename,
00669   ctl_t * ctl,
00670   atm_t * atm);
00671
00673 void read_ctl(
00674   const char *filename,
00675   int argc,
00676   char *argv[],
00677   ctl_t * ctl);
00678
00680 void read_met(
00681   ctl_t * ctl,
00682   char *filename,
00683   met_t * met);
00684
00686 void read_met_extrapolate(
00687   met_t * met);
00688
00690 void read_met_help(
00691   int ncid,
00692   char *varname,
00693   char *varname2,
00694   met_t * met,
00695   float dest[EX][EY][EP],
00696   float scl);
00697
00699 void read_met_ml2pl(
00700   ctl_t * ctl,
00701   met_t * met,
00702   float var[EX][EY][EP]);
00703
00705 void read_met_periodic(
00706   met_t * met);
00707
00709 double scan_ctl(
00710   const char *filename,
00711   int argc,
00712   char *argv[],
00713   const char *varname,
00714   int arridx,
00715   const char *defvalue,
```

```
00716   char *value);
00717
00719 void time2jsec(
00720   int year,
00721   int mon,
00722   int day,
00723   int hour,
00724   int min,
00725   int sec,
00726   double remain,
00727   double *jsec);
00728
00730 void timer(
00731   const char *name,
00732   int id,
00733   int mode);
00734
00735 /* Get tropopause pressure... */
00736 double tropopause(
00737   double t,
00738   double lat);
00739
00741 void write_atm(
00742   const char *filename,
00743   ctl_t * ctl,
00744   atm_t * atm,
00745   double t);
00746
00748 void write_csi(
00749   const char *filename,
00750   ctl_t * ctl,
00751   atm_t * atm,
00752   double t);
00753
00755 void write_ens(
00756   const char *filename,
00757   ctl_t * ctl,
00758   atm_t * atm,
00759   double t);
00760
00762 void write_grid(
00763   const char *filename,
00764   ctl_t * ctl,
00765   met_t * met0,
00766   met_t * met1,
00767   atm_t * atm,
00768   double t);
00769
00771 void write_prof(
00772   const char *filename,
00773   ctl_t * ctl,
00774   met_t * met0,
00775   met_t * met1,
00776   atm_t * atm,
00777   double t);
00778
00780 void write_station(
00781   const char *filename,
00782   ctl_t * ctl,
00783   atm_t * atm,
00784   double t);
```

5.15 match.c File Reference

Calculate deviations between two trajectories.

Functions

- int main (int argc, char ∗argv[ ])

5.15.1 Detailed Description

Calculate deviations between two trajectories.

Definition in file match.c.

5.15.2 Function Documentation

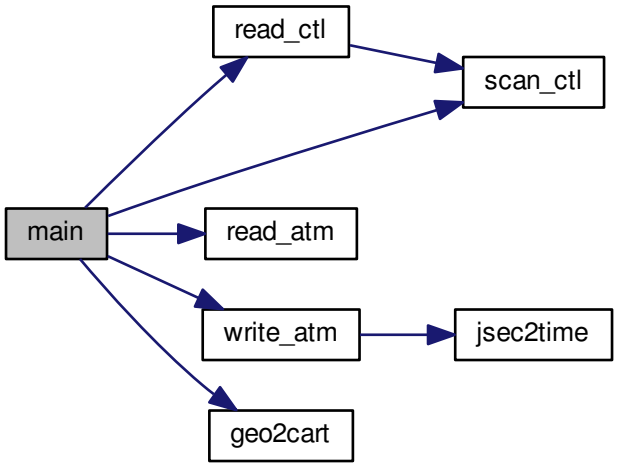
5.15.2.1 int main (int *argc*, char * *argv[]*)

Definition at line 28 of file match.c.

```
00030                {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm1, *atm2, *atm3;
00035
00036   FILE *out;
00037
00038   char filename[LEN];
00039
00040   double filter_dt, x1[3], x2[3], dh, dq[NQ], dv, lh = 0, lt = 0, lv = 0;
00041
00042   int filter, ip1, ip2, iq, n;
00043
00044   /* Allocate... */
00045   ALLOC(atm1, atm_t, 1);
00046   ALLOC(atm2, atm_t, 1);
00047   ALLOC(atm3, atm_t, 1);
00048
00049   /* Check arguments... */
00050   if (argc < 5)
00051     ERRMSG("Give parameters: <ctl> <atm_test> <atm_ref> <outfile>");
00052
00053   /* Read control parameters... */
00054   read_ctl(argv[1], argc, argv, &ctl);
00055   filter = (int) scan_ctl(argv[1], argc, argv, "FILTER", -1, "0", NULL);
00056   filter_dt = scan_ctl(argv[1], argc, argv, "FILTER_DT", -1, "0", NULL);
00057
00058   /* Read atmospheric data... */
00059   read_atm(argv[2], &ctl, atm1);
00060   read_atm(argv[3], &ctl, atm2);
00061
00062   /* Write info... */
00063   printf("Write transport deviations: %s\n", argv[4]);
00064
00065   /* Create output file... */
00066   if (!(out = fopen(argv[4], "w")))
00067     ERRMSG("Cannot create file!");
00068
00069   /* Write header... */
00070   fprintf(out,
00071           "# $1 = time [s]\n"
00072           "# $2 = altitude [km]\n"
00073           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
00074   for (iq = 0; iq < ctl.nq; iq++)
00075     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl.qnt_name[iq],
00076             ctl.qnt_unit[iq]);
00077   fprintf(out,
00078           "# $%d = trajectory time [s]\n"
00079           "# $%d = vertical length of trajectory [km]\n"
00080           "# $%d = horizontal length of trajectory [km]\n"
00081           "# $%d = vertical deviation [km]\n"
00082           "# $%d = horizontal deviation [km]\n",
00083           5 + ctl.nq, 6 + ctl.nq, 7 + ctl.nq, 8 + ctl.nq, 9 + ctl.nq);
00084   for (iq = 0; iq < ctl.nq; iq++)
00085     fprintf(out, "# $%d = %s deviation [%s]\n", ctl.nq + iq + 10,
00086             ctl.qnt_name[iq], ctl.qnt_unit[iq]);
00087   fprintf(out, "\n");
00088
00089   /* Filtering of reference time series... */
00090   if (filter) {
00091
00092     /* Copy data... */
00093     memcpy(atm3, atm2, sizeof(atm_t));
00094
00095     /* Loop over data points... */
00096     for (ip1 = 0; ip1 < atm2->np; ip1++) {
00097       n = 0;
00098       atm2->p[ip1] = 0;
00099       for (iq = 0; iq < ctl.nq; iq++)
00100         atm2->q[iq][ip1] = 0;
00101       for (ip2 = 0; ip2 < atm2->np; ip2++)
00102         if (fabs(atm2->time[ip1] - atm2->time[ip2]) < filter_dt) {
00103           atm2->p[ip1] += atm3->p[ip2];
00104           for (iq = 0; iq < ctl.nq; iq++)
00105             atm2->q[iq][ip1] += atm3->q[iq][ip2];
```

```
00106           n++;
00107         }
00108       atm2->p[ip1] /= n;
00109       for (iq = 0; iq < ctl.nq; iq++)
00110         atm2->q[iq][ip1] /= n;
00111     }
00112
00113     /* Write filtered data... */
00114     sprintf(filename, "%s.filt", argv[3]);
00115     write_atm(filename, &ctl, atm2, 0);
00116   }
00117
00118   /* Loop over air parcels (reference data)... */
00119   for (ip2 = 0; ip2 < atm2->np; ip2++) {
00120
00121     /* Get trajectory length... */
00122     if (ip2 > 0) {
00123       geo2cart(0, atm2->lon[ip2 - 1], atm2->lat[ip2 - 1], x1);
00124       geo2cart(0, atm2->lon[ip2], atm2->lat[ip2], x2);
00125       lh += DIST(x1, x2);
00126       lv += fabs(Z(atm2->p[ip2 - 1]) - Z(atm2->p[ip2]));
00127       lt = fabs(atm2->time[ip2] - atm2->time[0]);
00128     }
00129
00130     /* Init... */
00131     n = 0;
00132     dh = 0;
00133     dv = 0;
00134     for (iq = 0; iq < ctl.nq; iq++)
00135       dq[iq] = 0;
00136     geo2cart(0, atm2->lon[ip2], atm2->lat[ip2], x2);
00137
00138     /* Find corresponding time step (test data)... */
00139     for (ip1 = 0; ip1 < atm1->np; ip1++)
00140       if (fabs(atm1->time[ip1] - atm2->time[ip2])
00141           < (filter ? filter_dt : 0.1)) {
00142
00143         /* Calculate deviations... */
00144         geo2cart(0, atm1->lon[ip1], atm1->lat[ip1], x1);
00145         dh += DIST(x1, x2);
00146         dv += Z(atm1->p[ip1]) - Z(atm2->p[ip2]);
00147         for (iq = 0; iq < ctl.nq; iq++)
00148           dq[iq] += atm1->q[iq][ip1] - atm2->q[iq][ip2];
00149         n++;
00150       }
00151
00152     /* Write output... */
00153     if (n > 0) {
00154       fprintf(out, "%.2f %.4f %.4f %.4f",
00155               atm2->time[ip2], Z(atm2->p[ip2]),
00156               atm2->lon[ip2], atm2->lat[ip2]);
00157       for (iq = 0; iq < ctl.nq; iq++) {
00158         fprintf(out, " ");
00159         fprintf(out, ctl.qnt_format[iq], atm2->q[iq][ip2]);
00160       }
00161       fprintf(out, " %.2f %g %g %g %g", lt, lv, lh, dv / n, dh / n);
00162       for (iq = 0; iq < ctl.nq; iq++) {
00163         fprintf(out, " ");
00164         fprintf(out, ctl.qnt_format[iq], dq[iq] / n);
00165       }
00166       fprintf(out, "\n");
00167     }
00168   }
00169
00170   /* Close file... */
00171   fclose(out);
00172
00173   /* Free... */
00174   free(atm1);
00175   free(atm2);
00176   free(atm3);
00177
00178   return EXIT_SUCCESS;
00179 }
```

Here is the call graph for this function:



5.16 match.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026 #include <gsl/gsl_sort.h>
00027
00028 int main(
00029   int argc,
00030   char *argv[]) {
00031
00032   ctl_t ctl;
00033
00034   atm_t *atm1, *atm2, *atm3;
00035
00036   FILE *out;
00037
00038   char filename[LEN];
00039
00040   double filter_dt, x1[3], x2[3], dh, dq[NQ], dv, lh = 0, lt = 0, lv = 0;
00041
00042   int filter, ip1, ip2, iq, n;
00043
00044   /* Allocate... */
00045   ALLOC(atm1, atm_t, 1);
00046   ALLOC(atm2, atm_t, 1);
00047   ALLOC(atm3, atm_t, 1);
00048
00049   /* Check arguments... */
```

```
00050   if (argc < 5)
00051     ERRMSG("Give parameters: <ctl> <atm_test> <atm_ref> <outfile>");
00052
00053   /* Read control parameters... */
00054   read_ctl(argv[1], argc, argv, &ctl);
00055   filter = (int) scan_ctl(argv[1], argc, argv, "FILTER", -1, "0", NULL);
00056   filter_dt = scan_ctl(argv[1], argc, argv, "FILTER_DT", -1, "0", NULL);
00057
00058   /* Read atmospheric data... */
00059   read_atm(argv[2], &ctl, atm1);
00060   read_atm(argv[3], &ctl, atm2);
00061
00062   /* Write info... */
00063   printf("Write transport deviations: %s\n", argv[4]);
00064
00065   /* Create output file... */
00066   if (!(out = fopen(argv[4], "w")))
00067     ERRMSG("Cannot create file!");
00068
00069   /* Write header... */
00070   fprintf(out,
00071           "# $1 = time [s]\n"
00072           "# $2 = altitude [km]\n"
00073           "# $3 = longitude [deg]\n" "# $4 = latitude [deg]\n");
00074   for (iq = 0; iq < ctl.nq; iq++)
00075     fprintf(out, "# $%i = %s [%s]\n", iq + 5, ctl.qnt_name[iq],
00076             ctl.qnt_unit[iq]);
00077   fprintf(out,
00078           "# $%d = trajectory time [s]\n"
00079           "# $%d = vertical length of trajectory [km]\n"
00080           "# $%d = horizontal length of trajectory [km]\n"
00081           "# $%d = vertical deviation [km]\n"
00082           "# $%d = horizontal deviation [km]\n",
00083           5 + ctl.nq, 6 + ctl.nq, 7 + ctl.nq, 8 + ctl.nq, 9 + ctl.nq);
00084   for (iq = 0; iq < ctl.nq; iq++)
00085     fprintf(out, "# $%d = %s deviation [%s]\n", ctl.nq + iq + 10,
00086             ctl.qnt_name[iq], ctl.qnt_unit[iq]);
00087   fprintf(out, "\n");
00088
00089   /* Filtering of reference time series... */
00090   if (filter) {
00091
00092     /* Copy data... */
00093     memcpy(atm3, atm2, sizeof(atm_t));
00094
00095     /* Loop over data points... */
00096     for (ip1 = 0; ip1 < atm2->np; ip1++) {
00097       n = 0;
00098       atm2->p[ip1] = 0;
00099       for (iq = 0; iq < ctl.nq; iq++)
00100         atm2->q[iq][ip1] = 0;
00101       for (ip2 = 0; ip2 < atm2->np; ip2++)
00102         if (fabs(atm2->time[ip1] - atm2->time[ip2]) < filter_dt) {
00103           atm2->p[ip1] += atm3->p[ip2];
00104           for (iq = 0; iq < ctl.nq; iq++)
00105             atm2->q[iq][ip1] += atm3->q[iq][ip2];
00106           n++;
00107         }
00108       atm2->p[ip1] /= n;
00109       for (iq = 0; iq < ctl.nq; iq++)
00110         atm2->q[iq][ip1] /= n;
00111     }
00112
00113     /* Write filtered data... */
00114     sprintf(filename, "%s.filt", argv[3]);
00115     write_atm(filename, &ctl, atm2, 0);
00116   }
00117
00118   /* Loop over air parcels (reference data)... */
00119   for (ip2 = 0; ip2 < atm2->np; ip2++) {
00120
00121     /* Get trajectory length... */
00122     if (ip2 > 0) {
00123       geo2cart(0, atm2->lon[ip2 - 1], atm2->lat[ip2 - 1], x1);
00124       geo2cart(0, atm2->lon[ip2], atm2->lat[ip2], x2);
00125       lh += DIST(x1, x2);
00126       lv += fabs(Z(atm2->p[ip2 - 1]) - Z(atm2->p[ip2]));
00127       lt = fabs(atm2->time[ip2] - atm2->time[0]);
00128     }
00129
00130     /* Init... */
00131     n = 0;
00132     dh = 0;
00133     dv = 0;
00134     for (iq = 0; iq < ctl.nq; iq++)
00135       dq[iq] = 0;
00136     geo2cart(0, atm2->lon[ip2], atm2->lat[ip2], x2);
```

```
00137
00138     /* Find corresponding time step (test data)... */
00139     for (ip1 = 0; ip1 < atm1->np; ip1++)
00140       if (fabs(atm1->time[ip1] - atm2->time[ip2])
00141           < (filter ? filter_dt : 0.1)) {
00142
00143         /* Calculate deviations... */
00144         geo2cart(0, atm1->lon[ip1], atm1->lat[ip1], x1);
00145         dh += DIST(x1, x2);
00146         dv += Z(atm1->p[ip1]) - Z(atm2->p[ip2]);
00147         for (iq = 0; iq < ctl.nq; iq++)
00148           dq[iq] += atm1->q[iq][ip1] - atm2->q[iq][ip2];
00149         n++;
00150       }
00151
00152     /* Write output... */
00153     if (n > 0) {
00154       fprintf(out, "%.2f %.4f %.4f %.4f",
00155               atm2->time[ip2], Z(atm2->p[ip2]),
00156               atm2->lon[ip2], atm2->lat[ip2]);
00157       for (iq = 0; iq < ctl.nq; iq++) {
00158         fprintf(out, " ");
00159         fprintf(out, ctl.qnt_format[iq], atm2->q[iq][ip2]);
00160       }
00161       fprintf(out, " %.2f %g %g %g %g", lt, lv, lh, dv / n, dh / n);
00162       for (iq = 0; iq < ctl.nq; iq++) {
00163         fprintf(out, " ");
00164         fprintf(out, ctl.qnt_format[iq], dq[iq] / n);
00165       }
00166       fprintf(out, "\n");
00167     }
00168   }
00169
00170   /* Close file... */
00171   fclose(out);
00172
00173   /* Free... */
00174   free(atm1);
00175   free(atm2);
00176   free(atm3);
00177
00178   return EXIT_SUCCESS;
00179 }
```

5.17 met_map.c File Reference

Extract global map from meteorological data.

Functions

- int main (int argc, char ∗argv[ ])

5.17.1 Detailed Description

Extract global map from meteorological data.

Definition in file met_map.c.

5.17.2 Function Documentation

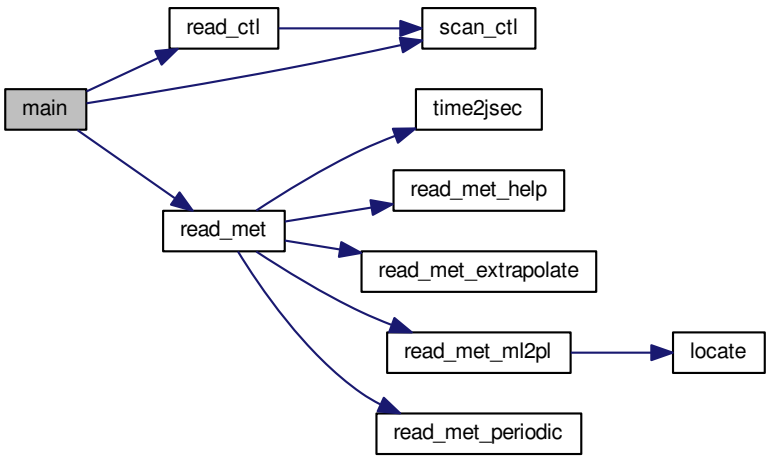
5.17.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 27 of file met_map.c.

```
00029                {
00030
00031   ctl_t ctl;
00032
00033   met_t *met;
00034
00035   FILE *in, *out;
00036
00037   static double dz, dzmin = 1e10, z, timem[EX][EY], psm[EX][EY], tm[EX][EY],
00038     um[EX][EY], vm[EX][EY], wm[EX][EY], h2om[EX][EY], o3m[EX][EY];
00039
00040   static int i, ip, ip2, ix, iy, np[EX][EY];
00041
00042   /* Allocate... */
00043   ALLOC(met, met_t, 1);
00044
00045   /* Check arguments... */
00046   if (argc < 4)
00047     ERRMSG("Give parameters: <ctl> <map.tab> <met0> [ <met1> ... ]");
00048
00049   /* Read control parameters... */
00050   read_ctl(argv[1], argc, argv, &ctl);
00051   z = scan_ctl(argv[1], argc, argv, "Z", -1, "", NULL);
00052
00053   /* Loop over files... */
00054   for (i = 3; i < argc; i++) {
00055
00056     /* Read meteorological data... */
00057     if (!(in = fopen(argv[i], "r")))
00058       continue;
00059     else
00060       fclose(in);
00061     read_met(&ctl, argv[i], met);
00062
00063     /* Find nearest pressure level... */
00064     for (ip2 = 0; ip2 < met->np; ip2++) {
00065       dz = fabs(Z(met->p[ip2]) - z);
00066       if (dz < dzmin) {
00067         dzmin = dz;
00068         ip = ip2;
00069       }
00070     }
00071
00072     /* Average data... */
00073     for (ix = 0; ix < met->nx; ix++)
00074       for (iy = 0; iy < met->ny; iy++) {
00075         timem[ix][iy] += met->time;
00076         tm[ix][iy] += met->t[ix][iy][ip];
00077         um[ix][iy] += met->u[ix][iy][ip];
00078         vm[ix][iy] += met->v[ix][iy][ip];
00079         wm[ix][iy] += met->w[ix][iy][ip];
00080         h2om[ix][iy] += met->h2o[ix][iy][ip];
00081         o3m[ix][iy] += met->o3[ix][iy][ip];
00082         psm[ix][iy] += met->ps[ix][iy];
00083         np[ix][iy]++;
00084       }
00085   }
00086
00087   /* Create output file... */
00088   printf("Write meteorological data file: %s\n", argv[2]);
00089   if (!(out = fopen(argv[2], "w")))
00090     ERRMSG("Cannot create file!");
00091
00092   /* Write header... */
00093   fprintf(out,
00094           "# $1  = time [s]\n"
00095           "# $2  = altitude [km]\n"
00096           "# $3  = longitude [deg]\n"
00097           "# $4  = latitude [deg]\n"
00098           "# $5  = pressure [hPa]\n"
00099           "# $6  = temperature [K]\n"
00100           "# $7  = zonal wind [m/s]\n"
00101           "# $8  = meridional wind [m/s]\n"
00102           "# $9  = vertical wind [hPa/s]\n"
00103           "# $10 = H2O volume mixing ratio [1]\n"
00104           "# $11 = O3 volume mixing ratio [1]\n"
00105           "# $12 = surface pressure [hPa]\n");
00106
00107   /* Write data... */
00108   for (iy = 0; iy < met->ny; iy++) {
00109     fprintf(out, "\n");
00110     for (ix = 0; ix < met->nx; ix++)
00111       if (met->lon[ix] >= 180)
00112         fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00113                 timem[ix][iy] / np[ix][iy], Z(met->p[ip]),
00114                 met->lon[ix] - 360.0, met->lat[iy], met->p[ip],
00115                 tm[ix][iy] / np[ix][iy], um[ix][iy] / np[ix][iy],
```

```
00116                 vm[ix][iy] / np[ix][iy], wm[ix][iy] / np[ix][iy],
00117                 h2om[ix][iy] / np[ix][iy], o3m[ix][iy] / np[ix][iy],
00118                 psm[ix][iy] / np[ix][iy]);
00119     for (ix = 0; ix < met->nx; ix++)
00120       if (met->lon[ix] <= 180)
00121         fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00122                 timem[ix][iy] / np[ix][iy], Z(met->p[ip]),
00123                 met->lon[ix], met->lat[iy], met->p[ip],
00124                 tm[ix][iy] / np[ix][iy], um[ix][iy] / np[ix][iy],
00125                 vm[ix][iy] / np[ix][iy], wm[ix][iy] / np[ix][iy],
00126                 h2om[ix][iy] / np[ix][iy], o3m[ix][iy] / np[ix][iy],
00127                 psm[ix][iy] / np[ix][iy]);
00128   }
00129
00130   /* Close file... */
00131   fclose(out);
00132
00133   /* Free... */
00134   free(met);
00135
00136   return EXIT_SUCCESS;
00137 }
```

Here is the call graph for this function:



5.18 met_map.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
```

```
00027 int main(
00028   int argc,
00029   char *argv[]) {
00030
00031   ctl_t ctl;
00032
00033   met_t *met;
00034
00035   FILE *in, *out;
00036
00037   static double dz, dzmin = 1e10, z, timem[EX][EY], psm[EX][EY], tm[EX][EY],
00038     um[EX][EY], vm[EX][EY], wm[EX][EY], h2om[EX][EY], o3m[EX][EY];
00039
00040   static int i, ip, ip2, ix, iy, np[EX][EY];
00041
00042   /* Allocate... */
00043   ALLOC(met, met_t, 1);
00044
00045   /* Check arguments... */
00046   if (argc < 4)
00047     ERRMSG("Give parameters: <ctl> <map.tab> <met0> [ <met1> ... ]");
00048
00049   /* Read control parameters... */
00050   read_ctl(argv[1], argc, argv, &ctl);
00051   z = scan_ctl(argv[1], argc, argv, "Z", -1, "", NULL);
00052
00053   /* Loop over files... */
00054   for (i = 3; i < argc; i++) {
00055
00056     /* Read meteorological data... */
00057     if (!(in = fopen(argv[i], "r")))
00058       continue;
00059     else
00060       fclose(in);
00061     read_met(&ctl, argv[i], met);
00062
00063     /* Find nearest pressure level... */
00064     for (ip2 = 0; ip2 < met->np; ip2++) {
00065       dz = fabs(Z(met->p[ip2]) - z);
00066       if (dz < dzmin) {
00067         dzmin = dz;
00068         ip = ip2;
00069       }
00070     }
00071
00072     /* Average data... */
00073     for (ix = 0; ix < met->nx; ix++)
00074       for (iy = 0; iy < met->ny; iy++) {
00075         timem[ix][iy] += met->time;
00076         tm[ix][iy] += met->t[ix][iy][ip];
00077         um[ix][iy] += met->u[ix][iy][ip];
00078         vm[ix][iy] += met->v[ix][iy][ip];
00079         wm[ix][iy] += met->w[ix][iy][ip];
00080         h2om[ix][iy] += met->h2o[ix][iy][ip];
00081         o3m[ix][iy] += met->o3[ix][iy][ip];
00082         psm[ix][iy] += met->ps[ix][iy];
00083         np[ix][iy]++;
00084       }
00085   }
00086
00087   /* Create output file... */
00088   printf("Write meteorological data file: %s\n", argv[2]);
00089   if (!(out = fopen(argv[2], "w")))
00090     ERRMSG("Cannot create file!");
00091
00092   /* Write header... */
00093   fprintf(out,
00094           "# $1  = time [s]\n"
00095           "# $2  = altitude [km]\n"
00096           "# $3  = longitude [deg]\n"
00097           "# $4  = latitude [deg]\n"
00098           "# $5  = pressure [hPa]\n"
00099           "# $6  = temperature [K]\n"
00100           "# $7  = zonal wind [m/s]\n"
00101           "# $8  = meridional wind [m/s]\n"
00102           "# $9  = vertical wind [hPa/s]\n"
00103           "# $10 = H2O volume mixing ratio [1]\n"
00104           "# $11 = O3 volume mixing ratio [1]\n"
00105           "# $12 = surface pressure [hPa]\n");
00106
00107   /* Write data... */
00108   for (iy = 0; iy < met->ny; iy++) {
00109     fprintf(out, "\n");
00110     for (ix = 0; ix < met->nx; ix++)
00111       if (met->lon[ix] >= 180)
00112         fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00113                 timem[ix][iy] / np[ix][iy], Z(met->p[ip]),
```

```
00114                 met->lon[ix] - 360.0, met->lat[iy], met->p[ip],
00115                 tm[ix][iy] / np[ix][iy], um[ix][iy] / np[ix][iy],
00116                 vm[ix][iy] / np[ix][iy], wm[ix][iy] / np[ix][iy],
00117                 h2om[ix][iy] / np[ix][iy], o3m[ix][iy] / np[ix][iy],
00118                 psm[ix][iy] / np[ix][iy]);
00119     for (ix = 0; ix < met->nx; ix++)
00120       if (met->lon[ix] <= 180)
00121         fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00122                 timem[ix][iy] / np[ix][iy], Z(met->p[ip]),
00123                 met->lon[ix], met->lat[iy], met->p[ip],
00124                 tm[ix][iy] / np[ix][iy], um[ix][iy] / np[ix][iy],
00125                 vm[ix][iy] / np[ix][iy], wm[ix][iy] / np[ix][iy],
00126                 h2om[ix][iy] / np[ix][iy], o3m[ix][iy] / np[ix][iy],
00127                 psm[ix][iy] / np[ix][iy]);
00128   }
00129
00130   /* Close file... */
00131   fclose(out);
00132
00133   /* Free... */
00134   free(met);
00135
00136   return EXIT_SUCCESS;
00137 }
```

5.19 met_prof.c File Reference

Extract vertical profile from meteorological data.

Functions

- int main (int argc, char ∗argv[ ])

5.19.1 Detailed Description

Extract vertical profile from meteorological data.

Definition in file met_prof.c.

5.19.2 Function Documentation

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

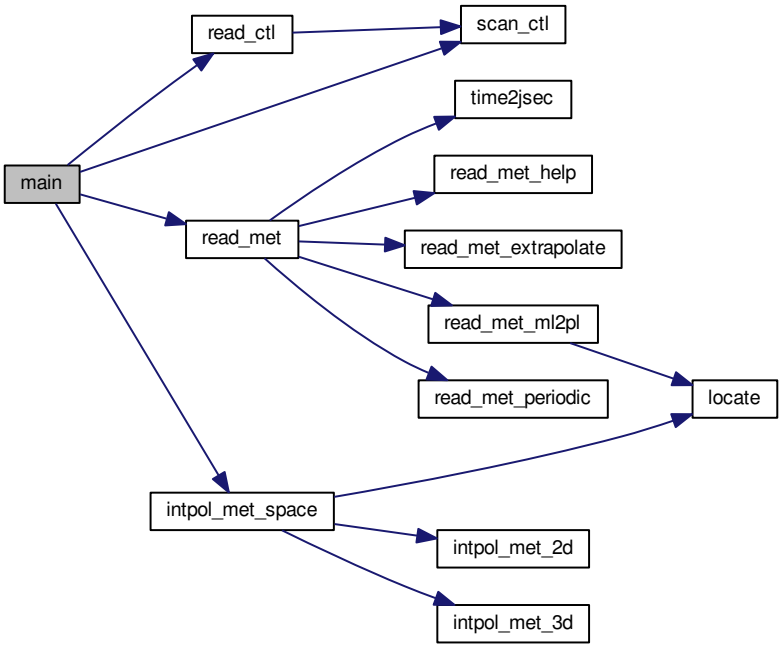
Definition at line 38 of file met_prof.c.

```
00040                 {
00041
00042   ctl_t ctl;
00043
00044   met_t *met;
00045
00046   FILE *in, *out;
00047
00048   static double timem[NZ], z, z0, z1, dz, lon, lon0, lon1, dlon, lonm[NZ],
00049     lat, lat0, lat1, dlat, latm[NZ], t, tm[NZ], u, um[NZ], v, vm[NZ],
00050     w, wm[NZ], h2o, h2om[NZ], o3, o3m[NZ], ps, psm[NZ];
00051
00052   static int i, iz, np[NZ];
00053
00054   /* Allocate... */
00055   ALLOC(met, met_t, 1);
00056
00057   /* Check arguments... */
00058   if (argc < 4)
00059     ERRMSG("Give parameters: <ctl> <prof.tab> <met0> [ <met1> ... ]");
```

```
00060
00061   /* Read control parameters... */
00062   read_ctl(argv[1], argc, argv, &ctl);
00063   z0 = scan_ctl(argv[1], argc, argv, "Z0", -1, "0", NULL);
00064   z1 = scan_ctl(argv[1], argc, argv, "Z1", -1, "60", NULL);
00065   dz = scan_ctl(argv[1], argc, argv, "DZ", -1, "1", NULL);
00066   lon0 = scan_ctl(argv[1], argc, argv, "LON0", -1, "0", NULL);
00067   lon1 = scan_ctl(argv[1], argc, argv, "LON1", -1, "0", NULL);
00068   dlon = scan_ctl(argv[1], argc, argv, "DLON", -1, "1", NULL);
00069   lat0 = scan_ctl(argv[1], argc, argv, "LAT0", -1, "0", NULL);
00070   lat1 = scan_ctl(argv[1], argc, argv, "LAT1", -1, "0", NULL);
00071   dlat = scan_ctl(argv[1], argc, argv, "DLAT", -1, "1", NULL);
00072
00073   /* Loop over input files... */
00074   for (i = 3; i < argc; i++) {
00075
00076     /* Read meteorological data... */
00077     if (!(in = fopen(argv[i], "r")))
00078       continue;
00079     else
00080       fclose(in);
00081     read_met(&ctl, argv[i], met);
00082
00083     /* Average... */
00084     for (z = z0; z <= z1; z += dz) {
00085       iz = (int) ((z - z0) / dz);
00086       if (iz < 0 || iz > NZ)
00087         ERRMSG("Too many altitudes!");
00088       for (lon = lon0; lon <= lon1; lon += dlon)
00089         for (lat = lat0; lat <= lat1; lat += dlat) {
00090           intpol_met_space(met, P(z), lon, lat, &ps,
00091                            &t, &u, &v, &w, &h2o, &o3);
00092           if (gsl_finite(t) && gsl_finite(u)
00093               && gsl_finite(v) && gsl_finite(w)) {
00094             timem[iz] += met->time;
00095             lonm[iz] += lon;
00096             latm[iz] += lat;
00097             tm[iz] += t;
00098             um[iz] += u;
00099             vm[iz] += v;
00100             wm[iz] += w;
00101             h2om[iz] += h2o;
00102             o3m[iz] += o3;
00103             psm[iz] += ps;
00104             np[iz]++;
00105           }
00106         }
00107     }
00108   }
00109
00110   /* Normalize... */
00111   for (z = z0; z <= z1; z += dz) {
00112     iz = (int) ((z - z0) / dz);
00113     if (np[iz] > 0) {
00114       timem[iz] /= np[iz];
00115       lonm[iz] /= np[iz];
00116       latm[iz] /= np[iz];
00117       tm[iz] /= np[iz];
00118       um[iz] /= np[iz];
00119       vm[iz] /= np[iz];
00120       wm[iz] /= np[iz];
00121       h2om[iz] /= np[iz];
00122       o3m[iz] /= np[iz];
00123       psm[iz] /= np[iz];
00124     } else {
00125       timem[iz] = GSL_NAN;
00126       lonm[iz] = GSL_NAN;
00127       latm[iz] = GSL_NAN;
00128       tm[iz] = GSL_NAN;
00129       um[iz] = GSL_NAN;
00130       vm[iz] = GSL_NAN;
00131       wm[iz] = GSL_NAN;
00132       h2om[iz] = GSL_NAN;
00133       o3m[iz] = GSL_NAN;
00134       psm[iz] = GSL_NAN;
00135     }
00136   }
00137
00138   /* Create output file... */
00139   printf("Write meteorological data file: %s\n", argv[2]);
00140   if (!(out = fopen(argv[2], "w")))
00141     ERRMSG("Cannot create file!");
00142
00143   /* Write header... */
00144   fprintf(out,
00145           "# $1  = time [s]\n"
00146           "# $2  = altitude [km]\n"
```

```
00147           "# $3  = longitude [deg]\n"
00148           "# $4  = latitude [deg]\n"
00149           "# $5  = pressure [hPa]\n"
00150           "# $6  = temperature [K]\n"
00151           "# $7  = zonal wind [m/s]\n"
00152           "# $8  = meridional wind [m/s]\n"
00153           "# $9  = vertical wind [hPa/s]\n"
00154           "# $10 = H2O volume mixing ratio [1]\n"
00155           "# $11 = O3 volume mixing ratio [1]\n"
00156           "# $12 = surface pressure [hPa]\n\n");
00157
00158   /* Write data... */
00159   for (z = z0; z <= z1; z += dz) {
00160     iz = (int) ((z - z0) / dz);
00161     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00162             timem[iz], z, lonm[iz], latm[iz], P(z), tm[iz],
00163             um[iz], vm[iz], wm[iz], h2om[iz], o3m[iz], psm[iz]);
00164   }
00165
00166   /* Close file... */
00167   fclose(out);
00168
00169   /* Free... */
00170   free(met);
00171
00172   return EXIT_SUCCESS;
00173 }
```

Here is the call graph for this function:



5.20 met_prof.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 /* ------------------------------------------------------------
00028    Dimensions...
00029    ------------------------------------------------------------ */
00030
00031 /* Maximum number of altitudes. */
00032 #define NZ 1000
00033
00034 /* ------------------------------------------------------------
00035    Main...
00036    ------------------------------------------------------------ */
00037
00038 int main(
00039   int argc,
00040   char *argv[]) {
00041
00042   ctl_t ctl;
00043
00044   met_t *met;
00045
00046   FILE *in, *out;
00047
00048   static double timem[NZ], z, z0, z1, dz, lon, lon0, lon1, dlon, lonm[NZ],
00049     lat, lat0, lat1, dlat, latm[NZ], t, tm[NZ], u, um[NZ], v, vm[NZ],
00050     w, wm[NZ], h2o, h2om[NZ], o3, o3m[NZ], ps, psm[NZ];
00051
00052   static int i, iz, np[NZ];
00053
00054   /* Allocate... */
00055   ALLOC(met, met_t, 1);
00056
00057   /* Check arguments... */
00058   if (argc < 4)
00059     ERRMSG("Give parameters: <ctl> <prof.tab> <met0> [ <met1> ... ]");
00060
00061   /* Read control parameters... */
00062   read_ctl(argv[1], argc, argv, &ctl);
00063   z0 = scan_ctl(argv[1], argc, argv, "Z0", -1, "0", NULL);
00064   z1 = scan_ctl(argv[1], argc, argv, "Z1", -1, "60", NULL);
00065   dz = scan_ctl(argv[1], argc, argv, "DZ", -1, "1", NULL);
00066   lon0 = scan_ctl(argv[1], argc, argv, "LON0", -1, "0", NULL);
00067   lon1 = scan_ctl(argv[1], argc, argv, "LON1", -1, "0", NULL);
00068   dlon = scan_ctl(argv[1], argc, argv, "DLON", -1, "1", NULL);
00069   lat0 = scan_ctl(argv[1], argc, argv, "LAT0", -1, "0", NULL);
00070   lat1 = scan_ctl(argv[1], argc, argv, "LAT1", -1, "0", NULL);
00071   dlat = scan_ctl(argv[1], argc, argv, "DLAT", -1, "1", NULL);
00072
00073   /* Loop over input files... */
00074   for (i = 3; i < argc; i++) {
00075
00076     /* Read meteorological data... */
00077     if (!(in = fopen(argv[i], "r")))
00078       continue;
00079     else
00080       fclose(in);
00081     read_met(&ctl, argv[i], met);
00082
00083     /* Average... */
00084     for (z = z0; z <= z1; z += dz) {
00085       iz = (int) ((z - z0) / dz);
00086       if (iz < 0 || iz > NZ)
00087         ERRMSG("Too many altitudes!");
00088       for (lon = lon0; lon <= lon1; lon += dlon)
00089         for (lat = lat0; lat <= lat1; lat += dlat) {
00090           intpol_met_space(met, P(z), lon, lat, &ps,
00091                            &t, &u, &v, &w, &h2o, &o3);
00092           if (gsl_finite(t) && gsl_finite(u)
00093               && gsl_finite(v) && gsl_finite(w)) {
00094             timem[iz] += met->time;
00095             lonm[iz] += lon;
00096             latm[iz] += lat;
00097             tm[iz] += t;
00098             um[iz] += u;
00099             vm[iz] += v;
00100             wm[iz] += w;
```

```
00101             h2om[iz] += h2o;
00102             o3m[iz] += o3;
00103             psm[iz] += ps;
00104             np[iz]++;
00105           }
00106         }
00107     }
00108   }
00109
00110   /* Normalize... */
00111   for (z = z0; z <= z1; z += dz) {
00112     iz = (int) ((z - z0) / dz);
00113     if (np[iz] > 0) {
00114       timem[iz] /= np[iz];
00115       lonm[iz] /= np[iz];
00116       latm[iz] /= np[iz];
00117       tm[iz] /= np[iz];
00118       um[iz] /= np[iz];
00119       vm[iz] /= np[iz];
00120       wm[iz] /= np[iz];
00121       h2om[iz] /= np[iz];
00122       o3m[iz] /= np[iz];
00123       psm[iz] /= np[iz];
00124     } else {
00125       timem[iz] = GSL_NAN;
00126       lonm[iz] = GSL_NAN;
00127       latm[iz] = GSL_NAN;
00128       tm[iz] = GSL_NAN;
00129       um[iz] = GSL_NAN;
00130       vm[iz] = GSL_NAN;
00131       wm[iz] = GSL_NAN;
00132       h2om[iz] = GSL_NAN;
00133       o3m[iz] = GSL_NAN;
00134       psm[iz] = GSL_NAN;
00135     }
00136   }
00137
00138   /* Create output file... */
00139   printf("Write meteorological data file: %s\n", argv[2]);
00140   if (!(out = fopen(argv[2], "w")))
00141     ERRMSG("Cannot create file!");
00142
00143   /* Write header... */
00144   fprintf(out,
00145           "# $1  = time [s]\n"
00146           "# $2  = altitude [km]\n"
00147           "# $3  = longitude [deg]\n"
00148           "# $4  = latitude [deg]\n"
00149           "# $5  = pressure [hPa]\n"
00150           "# $6  = temperature [K]\n"
00151           "# $7  = zonal wind [m/s]\n"
00152           "# $8  = meridional wind [m/s]\n"
00153           "# $9  = vertical wind [hPa/s]\n"
00154           "# $10 = H2O volume mixing ratio [1]\n"
00155           "# $11 = O3 volume mixing ratio [1]\n"
00156           "# $12 = surface pressure [hPa]\n\n");
00157
00158   /* Write data... */
00159   for (z = z0; z <= z1; z += dz) {
00160     iz = (int) ((z - z0) / dz);
00161     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g %g\n",
00162             timem[iz], z, lonm[iz], latm[iz], P(z), tm[iz],
00163             um[iz], vm[iz], wm[iz], h2om[iz], o3m[iz], psm[iz]);
00164   }
00165
00166   /* Close file... */
00167   fclose(out);
00168
00169   /* Free... */
00170   free(met);
00171
00172   return EXIT_SUCCESS;
00173 }
```

5.21 met_sample.c File Reference

Sample meteorological data at given geolocations.

Functions

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

5.21.1 Detailed Description

Sample meteorological data at given geolocations.

Definition in file met_sample.c.

5.21.2 Function Documentation

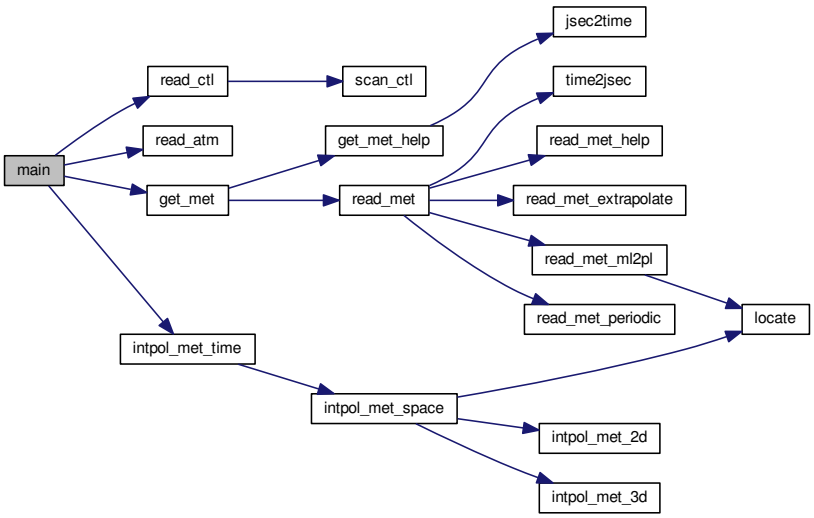
5.21.2.1 int main (int *argc*, char * *argv[]*)

Definition at line 31 of file met_sample.c.

```
00033                {
00034
00035   ctl_t ctl;
00036
00037   atm_t *atm;
00038
00039   met_t *met0, *met1;
00040
00041   FILE *out;
00042
00043   double t, u, v, w, h2o, o3;
00044
00045   int ip;
00046
00047   /* Check arguments... */
00048   if (argc < 4)
00049     ERRMSG("Give parameters: <ctl> <metbase> <atm_in> <sample.tab>");
00050
00051   /* Allocate... */
00052   ALLOC(atm, atm_t, 1);
00053   ALLOC(met0, met_t, 1);
00054   ALLOC(met1, met_t, 1);
00055
00056   /* Read control parameters... */
00057   read_ctl(argv[1], argc, argv, &ctl);
00058
00059   /* Read atmospheric data... */
00060   read_atm(argv[3], &ctl, atm);
00061
00062   /* Create output file... */
00063   printf("Write meteorological data file: %s\n", argv[4]);
00064   if (!(out = fopen(argv[4], "w")))
00065     ERRMSG("Cannot create file!");
00066
00067   /* Write header... */
00068   fprintf(out,
00069           "# $1  = time [s]\n"
00070           "# $2  = altitude [km]\n"
00071           "# $3  = longitude [deg]\n"
00072           "# $4  = latitude [deg]\n"
00073           "# $5  = pressure [hPa]\n"
00074           "# $6  = temperature [K]\n"
00075           "# $7  = zonal wind [m/s]\n"
00076           "# $8  = meridional wind [m/s]\n"
00077           "# $9  = vertical wind [hPa/s]\n"
00078           "# $10 = H2O volume mixing ratio [1]\n"
00079           "# $11 = O3 volume mixing ratio [1]\n\n");
00080
00081   /* Loop over air parcels... */
00082   for (ip = 0; ip < atm->np; ip++) {
00083
00084     /* Get meteorological data... */
00085     get_met(&ctl, argv[2], atm->time[ip], met0, met1);
00086
00087     /* Interpolate meteorological data... */
00088     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00089                     atm->lat[ip], NULL, &t, &u, &v, &w, &h2o, &o3);
00090
00091     /* Write data... */
00092     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g\n",
00093             atm->time[ip], Z(atm->p[ip]), atm->lon[ip], atm->lat[ip],
00094             atm->p[ip], t, u, v, w, h2o, o3);
00095   }
00096
```

```
00097   /* Close file... */
00098   fclose(out);
00099
00100   /* Free... */
00101   free(atm);
00102   free(met0);
00103   free(met1);
00104
00105   return EXIT_SUCCESS;
00106 }
```

Here is the call graph for this function:



5.22 met_sample.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 /* ------------------------------------------------------------
00028    Main...
00029    ------------------------------------------------------------ */
00030
00031 int main(
00032   int argc,
00033   char *argv[]) {
00034
00035   ctl_t ctl;
00036
00037   atm_t *atm;
00038
```

```
00039   met_t *met0, *met1;
00040
00041   FILE *out;
00042
00043   double t, u, v, w, h2o, o3;
00044
00045   int ip;
00046
00047   /* Check arguments... */
00048   if (argc < 4)
00049     ERRMSG("Give parameters: <ctl> <metbase> <atm_in> <sample.tab>");
00050
00051   /* Allocate... */
00052   ALLOC(atm, atm_t, 1);
00053   ALLOC(met0, met_t, 1);
00054   ALLOC(met1, met_t, 1);
00055
00056   /* Read control parameters... */
00057   read_ctl(argv[1], argc, argv, &ctl);
00058
00059   /* Read atmospheric data... */
00060   read_atm(argv[3], &ctl, atm);
00061
00062   /* Create output file... */
00063   printf("Write meteorological data file: %s\n", argv[4]);
00064   if (!(out = fopen(argv[4], "w")))
00065     ERRMSG("Cannot create file!");
00066
00067   /* Write header... */
00068   fprintf(out,
00069           "# $1  = time [s]\n"
00070           "# $2  = altitude [km]\n"
00071           "# $3  = longitude [deg]\n"
00072           "# $4  = latitude [deg]\n"
00073           "# $5  = pressure [hPa]\n"
00074           "# $6  = temperature [K]\n"
00075           "# $7  = zonal wind [m/s]\n"
00076           "# $8  = meridional wind [m/s]\n"
00077           "# $9  = vertical wind [hPa/s]\n"
00078           "# $10 = H2O volume mixing ratio [1]\n"
00079           "# $11 = O3 volume mixing ratio [1]\n\n");
00080
00081   /* Loop over air parcels... */
00082   for (ip = 0; ip < atm->np; ip++) {
00083
00084     /* Get meteorological data... */
00085     get_met(&ctl, argv[2], atm->time[ip], met0, met1);
00086
00087     /* Interpolate meteorological data... */
00088     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00089                     atm->lat[ip], NULL, &t, &u, &v, &w, &h2o, &o3);
00090
00091     /* Write data... */
00092     fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g\n",
00093             atm->time[ip], Z(atm->p[ip]), atm->lon[ip], atm->lat[ip],
00094             atm->p[ip], t, u, v, w, h2o, o3);
00095   }
00096
00097   /* Close file... */
00098   fclose(out);
00099
00100   /* Free... */
00101   free(atm);
00102   free(met0);
00103   free(met1);
00104
00105   return EXIT_SUCCESS;
00106 }
```

5.23 met_zm.c File Reference

Extract zonal mean from meteorological data.

Functions

- int main (int argc, char ∗argv[ ])

5.23.1 Detailed Description

Extract zonal mean from meteorological data.

Definition in file met_zm.c.

5.23.2 Function Documentation

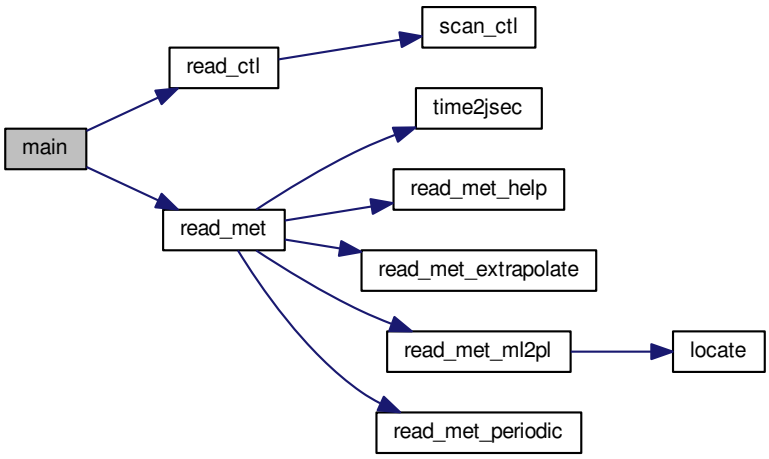
5.23.2.1 int main (int *argc*, char * *argv[]*)

Definition at line 27 of file met_zm.c.

```
00029                {
00030
00031   ctl_t ctl;
00032
00033   met_t *met;
00034
00035   FILE *in, *out;
00036
00037   static double timem[EP][EY], psm[EP][EY], tm[EP][EY], um[EP][EY],
00038     vm[EP][EY], vhm[EP][EY], wm[EP][EY], h2om[EP][EY], o3m[EP][EY],
00039     psm2[EP][EY], tm2[EP][EY], um2[EP][EY], vm2[EP][EY], vhm2[EP][EY],
00040     wm2[EP][EY], h2om2[EP][EY], o3m2[EP][EY];
00041
00042   static int i, ip, ix, iy, np[EP][EY];
00043
00044   /* Allocate... */
00045   ALLOC(met, met_t, 1);
00046
00047   /* Check arguments... */
00048   if (argc < 4)
00049     ERRMSG("Give parameters: <ctl> <zm.tab> <met0> [ <met1> ... ]");
00050
00051   /* Read control parameters... */
00052   read_ctl(argv[1], argc, argv, &ctl);
00053
00054   /* Loop over files... */
00055   for (i = 3; i < argc; i++) {
00056
00057     /* Read meteorological data... */
00058     if (!(in = fopen(argv[i], "r")))
00059       continue;
00060     else
00061       fclose(in);
00062     read_met(&ctl, argv[i], met);
00063
00064     /* Average data... */
00065     for (ix = 0; ix < met->nx; ix++)
00066       for (iy = 0; iy < met->ny; iy++)
00067         for (ip = 0; ip < met->np; ip++) {
00068           timem[ip][iy] += met->time;
00069           tm[ip][iy] += met->t[ix][iy][ip];
00070           um[ip][iy] += met->u[ix][iy][ip];
00071           vm[ip][iy] += met->v[ix][iy][ip];
00072           vhm[ip][iy] += sqrt(gsl_pow_2(met->u[ix][iy][ip])
00073                               + gsl_pow_2(met->v[ix][iy][ip]));
00074           wm[ip][iy] += met->w[ix][iy][ip];
00075           h2om[ip][iy] += met->h2o[ix][iy][ip];
00076           o3m[ip][iy] += met->o3[ix][iy][ip];
00077           psm[ip][iy] += met->ps[ix][iy];
00078           tm2[ip][iy] += gsl_pow_2(met->t[ix][iy][ip]);
00079           um2[ip][iy] += gsl_pow_2(met->u[ix][iy][ip]);
00080           vm2[ip][iy] += gsl_pow_2(met->v[ix][iy][ip]);
00081           vhm2[ip][iy] += gsl_pow_2(met->u[ix][iy][ip])
00082             + gsl_pow_2(met->v[ix][iy][ip]);
00083           wm2[ip][iy] += gsl_pow_2(met->w[ix][iy][ip]);
00084           h2om2[ip][iy] += gsl_pow_2(met->h2o[ix][iy][ip]);
00085           o3m2[ip][iy] += gsl_pow_2(met->o3[ix][iy][ip]);
00086           psm2[ip][iy] += gsl_pow_2(met->ps[ix][iy]);
00087           np[ip][iy]++;
00088         }
00089   }
00090
00091   /* Create output file... */
00092   printf("Write meteorological data file: %s\n", argv[2]);
00093   if (!(out = fopen(argv[2], "w")))
```

```
00094     ERRMSG("Cannot create file!");
00095
00096   /* Write header... */
00097   fprintf(out,
00098           "# $1  = time [s]\n"
00099           "# $2  = altitude [km]\n"
00100           "# $3  = latitude [deg]\n"
00101           "# $4  = temperature mean [K]\n"
00102           "# $5  = temperature standard deviation [K]\n"
00103           "# $6  = zonal wind mean [m/s]\n"
00104           "# $7  = zonal wind standard deviation [m/s]\n"
00105           "# $8  = meridional wind mean [m/s]\n"
00106           "# $9  = meridional wind standard deviation [m/s]\n"
00107           "# $10 = horizontal wind mean [m/s]\n"
00108           "# $11 = horizontal wind standard deviation [m/s]\n"
00109           "# $12 = vertical wind mean [hPa/s]\n"
00110           "# $13 = vertical wind standard deviation [hPa/s]\n"
00111           "# $14 = H2O vmr mean [1]\n"
00112           "# $15 = H2O vmr standard deviation [1]\n"
00113           "# $16 = O3 vmr mean [1]\n"
00114           "# $17 = O3 vmr standard deviation [1]\n"
00115           "# $18 = surface pressure mean [hPa]\n"
00116           "# $19 = surface pressure standard deviation [hPa]\n");
00117
00118   /* Write data... */
00119   for (iy = 0; iy < met->ny; iy++) {
00120     fprintf(out, "\n");
00121     for (ip = 0; ip < met->np; ip++)
00122       fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g"
00123               " %g %g %g %g %g %g %g %g\n",
00124               timem[ip][iy] / np[ip][iy], Z(met->p[ip]), met->lat[iy],
00125               tm[ip][iy] / np[ip][iy],
00126               sqrt(tm2[ip][iy] / np[ip][iy] -
00127                    gsl_pow_2(tm[ip][iy] / np[ip][iy])),
00128               um[ip][iy] / np[ip][iy],
00129               sqrt(um2[ip][iy] / np[ip][iy] -
00130                    gsl_pow_2(um[ip][iy] / np[ip][iy])),
00131               vm[ip][iy] / np[ip][iy],
00132               sqrt(vm2[ip][iy] / np[ip][iy] -
00133                    gsl_pow_2(vm[ip][iy] / np[ip][iy])),
00134               vhm[ip][iy] / np[ip][iy],
00135               sqrt(vhm2[ip][iy] / np[ip][iy] -
00136                    gsl_pow_2(vhm[ip][iy] / np[ip][iy])),
00137               wm[ip][iy] / np[ip][iy],
00138               sqrt(wm2[ip][iy] / np[ip][iy] -
00139                    gsl_pow_2(wm[ip][iy] / np[ip][iy])),
00140               h2om[ip][iy] / np[ip][iy],
00141               sqrt(h2om2[ip][iy] / np[ip][iy] -
00142                    gsl_pow_2(h2om[ip][iy] / np[ip][iy])),
00143               o3m[ip][iy] / np[ip][iy],
00144               sqrt(o3m2[ip][iy] / np[ip][iy] -
00145                    gsl_pow_2(o3m[ip][iy] / np[ip][iy])),
00146               psm[ip][iy] / np[ip][iy],
00147               sqrt(psm2[ip][iy] / np[ip][iy] -
00148                    gsl_pow_2(psm[ip][iy] / np[ip][iy])));
00149   }
00150
00151   /* Close file... */
00152   fclose(out);
00153
00154   /* Free... */
00155   free(met);
00156
00157   return EXIT_SUCCESS;
00158 }
```

Here is the call graph for this function:



5.24 met_zm.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 int main(
00028   int argc,
00029   char *argv[]) {
00030
00031   ctl_t ctl;
00032
00033   met_t *met;
00034
00035   FILE *in, *out;
00036
00037   static double timem[EP][EY], psm[EP][EY], tm[EP][EY], um[EP][EY],
00038     vm[EP][EY], vhm[EP][EY], wm[EP][EY], h2om[EP][EY], o3m[EP][EY],
00039     psm2[EP][EY], tm2[EP][EY], um2[EP][EY], vm2[EP][EY], vhm2[EP][EY],
00040     wm2[EP][EY], h2om2[EP][EY], o3m2[EP][EY];
00041
00042   static int i, ip, ix, iy, np[EP][EY];
00043
00044   /* Allocate... */
00045   ALLOC(met, met_t, 1);
00046
00047   /* Check arguments... */
00048   if (argc < 4)
00049     ERRMSG("Give parameters: <ctl> <zm.tab> <met0> [ <met1> ... ]");
00050
00051   /* Read control parameters... */
00052   read_ctl(argv[1], argc, argv, &ctl);
```

```
00053
00054   /* Loop over files... */
00055   for (i = 3; i < argc; i++) {
00056
00057     /* Read meteorological data... */
00058     if (!(in = fopen(argv[i], "r")))
00059       continue;
00060     else
00061       fclose(in);
00062     read_met(&ctl, argv[i], met);
00063
00064     /* Average data... */
00065     for (ix = 0; ix < met->nx; ix++)
00066       for (iy = 0; iy < met->ny; iy++)
00067         for (ip = 0; ip < met->np; ip++) {
00068           timem[ip][iy] += met->time;
00069           tm[ip][iy] += met->t[ix][iy][ip];
00070           um[ip][iy] += met->u[ix][iy][ip];
00071           vm[ip][iy] += met->v[ix][iy][ip];
00072           vhm[ip][iy] += sqrt(gsl_pow_2(met->u[ix][iy][ip])
00073                               + gsl_pow_2(met->v[ix][iy][ip]));
00074           wm[ip][iy] += met->w[ix][iy][ip];
00075           h2om[ip][iy] += met->h2o[ix][iy][ip];
00076           o3m[ip][iy] += met->o3[ix][iy][ip];
00077           psm[ip][iy] += met->ps[ix][iy];
00078           tm2[ip][iy] += gsl_pow_2(met->t[ix][iy][ip]);
00079           um2[ip][iy] += gsl_pow_2(met->u[ix][iy][ip]);
00080           vm2[ip][iy] += gsl_pow_2(met->v[ix][iy][ip]);
00081           vhm2[ip][iy] += gsl_pow_2(met->u[ix][iy][ip])
00082             + gsl_pow_2(met->v[ix][iy][ip]);
00083           wm2[ip][iy] += gsl_pow_2(met->w[ix][iy][ip]);
00084           h2om2[ip][iy] += gsl_pow_2(met->h2o[ix][iy][ip]);
00085           o3m2[ip][iy] += gsl_pow_2(met->o3[ix][iy][ip]);
00086           psm2[ip][iy] += gsl_pow_2(met->ps[ix][iy]);
00087           np[ip][iy]++;
00088         }
00089   }
00090
00091   /* Create output file... */
00092   printf("Write meteorological data file: %s\n", argv[2]);
00093   if (!(out = fopen(argv[2], "w")))
00094     ERRMSG("Cannot create file!");
00095
00096   /* Write header... */
00097   fprintf(out,
00098           "# $1  = time [s]\n"
00099           "# $2  = altitude [km]\n"
00100           "# $3  = latitude [deg]\n"
00101           "# $4  = temperature mean [K]\n"
00102           "# $5  = temperature standard deviation [K]\n"
00103           "# $6  = zonal wind mean [m/s]\n"
00104           "# $7  = zonal wind standard deviation [m/s]\n"
00105           "# $8  = meridional wind mean [m/s]\n"
00106           "# $9  = meridional wind standard deviation [m/s]\n"
00107           "# $10 = horizontal wind mean [m/s]\n"
00108           "# $11 = horizontal wind standard deviation [m/s]\n"
00109           "# $12 = vertical wind mean [hPa/s]\n"
00110           "# $13 = vertical wind standard deviation [hPa/s]\n"
00111           "# $14 = H2O vmr mean [1]\n"
00112           "# $15 = H2O vmr standard deviation [1]\n"
00113           "# $16 = O3 vmr mean [1]\n"
00114           "# $17 = O3 vmr standard deviation [1]\n"
00115           "# $18 = surface pressure mean [hPa]\n"
00116           "# $19 = surface pressure standard deviation [hPa]\n");
00117
00118   /* Write data... */
00119   for (iy = 0; iy < met->ny; iy++) {
00120     fprintf(out, "\n");
00121     for (ip = 0; ip < met->np; ip++)
00122       fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g"
00123               " %g %g %g %g %g %g %g %g\n",
00124               timem[ip][iy] / np[ip][iy], Z(met->p[ip]), met->lat[iy],
00125               tm[ip][iy] / np[ip][iy],
00126               sqrt(tm2[ip][iy] / np[ip][iy] -
00127                    gsl_pow_2(tm[ip][iy] / np[ip][iy])),
00128               um[ip][iy] / np[ip][iy],
00129               sqrt(um2[ip][iy] / np[ip][iy] -
00130                    gsl_pow_2(um[ip][iy] / np[ip][iy])),
00131               vm[ip][iy] / np[ip][iy],
00132               sqrt(vm2[ip][iy] / np[ip][iy] -
00133                    gsl_pow_2(vm[ip][iy] / np[ip][iy])),
00134               vhm[ip][iy] / np[ip][iy],
00135               sqrt(vhm2[ip][iy] / np[ip][iy] -
00136                    gsl_pow_2(vhm[ip][iy] / np[ip][iy])),
00137               wm[ip][iy] / np[ip][iy],
00138               sqrt(wm2[ip][iy] / np[ip][iy] -
00139                    gsl_pow_2(wm[ip][iy] / np[ip][iy])),
```

```
00140               h2om[ip][iy] / np[ip][iy],
00141               sqrt(h2om2[ip][iy] / np[ip][iy] -
00142                    gsl_pow_2(h2om[ip][iy] / np[ip][iy])),
00143               o3m[ip][iy] / np[ip][iy],
00144               sqrt(o3m2[ip][iy] / np[ip][iy] -
00145                    gsl_pow_2(o3m[ip][iy] / np[ip][iy])),
00146               psm[ip][iy] / np[ip][iy],
00147               sqrt(psm2[ip][iy] / np[ip][iy] -
00148                    gsl_pow_2(psm[ip][iy] / np[ip][iy])));
00149   }
00150
00151   /* Close file... */
00152   fclose(out);
00153
00154   /* Free... */
00155   free(met);
00156
00157   return EXIT_SUCCESS;
00158 }
```

5.25 smago.c File Reference

Estimate horizontal diffusivity based on Smagorinsky theory.

Functions

- int main (int argc, char ∗argv[ ])

5.25.1 Detailed Description

Estimate horizontal diffusivity based on Smagorinsky theory.

Definition in file smago.c.

5.25.2 Function Documentation

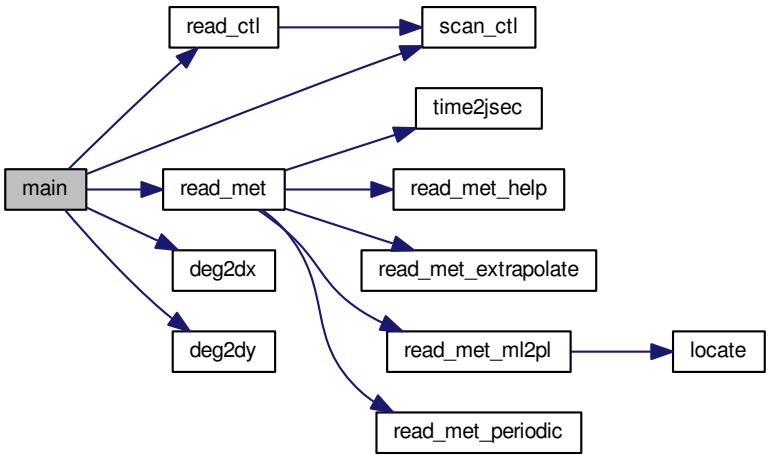
5.25.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 8 of file smago.c.

```
00010                {
00011
00012   ctl_t ctl;
00013
00014   met_t *met;
00015
00016   FILE *out;
00017
00018   static double dz, dzmin = 1e10, z, t, s, ls2, k[EX][EY], c = 0.15;
00019
00020   static int ip, ip2, ix, iy;
00021
00022   /* Allocate... */
00023   ALLOC(met, met_t, 1);
00024
00025   /* Check arguments... */
00026   if (argc < 4)
00027     ERRMSG("Give parameters: <ctl> <map.tab> <met>");
00028
00029   /* Read control parameters... */
00030   read_ctl(argv[1], argc, argv, &ctl);
00031   z = scan_ctl(argv[1], argc, argv, "Z", -1, "", NULL);
00032
00033   /* Read meteorological data... */
00034   read_met(&ctl, argv[3], met);
```

```
00035
00036   /* Find nearest pressure level... */
00037   for (ip2 = 0; ip2 < met->np; ip2++) {
00038     dz = fabs(Z(met->p[ip2]) - z);
00039     if (dz < dzmin) {
00040       dzmin = dz;
00041       ip = ip2;
00042     }
00043   }
00044
00045   /* Write info... */
00046   printf("Analyze %g hPa...\n", met->p[ip]);
00047
00048   /* Calculate horizontal diffusion coefficients... */
00049   for (ix = 1; ix < met->nx - 1; ix++)
00050     for (iy = 1; iy < met->ny - 1; iy++) {
00051       t = 0.5 * ((met->u[ix + 1][iy][ip] - met->u[ix - 1][iy][ip])
00052                  / (1000. *
00053                     deg2dx(met->lon[ix + 1] - met->lon[ix - 1], met->
      lat[iy]))
00054                  - (met->v[ix][iy + 1][ip] - met->v[ix][iy - 1][ip])
00055                  / (1000. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1])));
00056       s = 0.5 * ((met->u[ix][iy + 1][ip] - met->u[ix][iy - 1][ip])
00057                  / (1000. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1]))
00058                  + (met->v[ix + 1][iy][ip] - met->v[ix - 1][iy][ip])
00059                  / (1000. *
00060                     deg2dx(met->lon[ix + 1] - met->lon[ix - 1],
00061                            met->lat[iy])));
00062       ls2 = gsl_pow_2(c * 500. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1]));
00063       if (fabs(met->lat[iy]) > 80)
00064         ls2 *= (90. - fabs(met->lat[iy])) / 10.;
00065       k[ix][iy] = ls2 * sqrt(2.0 * (gsl_pow_2(t) + gsl_pow_2(s)));
00066     }
00067
00068   /* Create output file... */
00069   printf("Write data file: %s\n", argv[2]);
00070   if (!(out = fopen(argv[2], "w")))
00071     ERRMSG("Cannot create file!");
00072
00073   /* Write header... */
00074   fprintf(out,
00075           "# $1 = longitude [deg]\n"
00076           "# $2 = latitude [deg]\n"
00077           "# $3 = zonal wind [m/s]\n"
00078           "# $4 = meridional wind [m/s]\n"
00079           "# $5 = horizontal diffusivity [m^2/s]\n");
00080
00081   /* Write data... */
00082   for (iy = 0; iy < met->ny; iy++) {
00083     fprintf(out, "\n");
00084     for (ix = 0; ix < met->nx; ix++)
00085       if (met->lon[ix] >= 180)
00086         fprintf(out, "%g %g %g %g %g\n",
00087                 met->lon[ix] - 360.0, met->lat[iy],
00088                 met->u[ix][iy][ip], met->v[ix][iy][ip], k[ix][iy]);
00089     for (ix = 0; ix < met->nx; ix++)
00090       if (met->lon[ix] <= 180)
00091         fprintf(out, "%g %g %g %g %g\n",
00092                 met->lon[ix], met->lat[iy],
00093                 met->u[ix][iy][ip], met->v[ix][iy][ip], k[ix][iy]);
00094   }
00095
00096   /* Close file... */
00097   fclose(out);
00098
00099   /* Free... */
00100   free(met);
00101
00102   return EXIT_SUCCESS;
00103 }
```

Here is the call graph for this function:



5.26 smago.c

```
00001
00006 #include "libtrac.h"
00007
00008 int main(
00009   int argc,
00010   char *argv[]) {
00011
00012   ctl_t ctl;
00013
00014   met_t *met;
00015
00016   FILE *out;
00017
00018   static double dz, dzmin = 1e10, z, t, s, ls2, k[EX][EY], c = 0.15;
00019
00020   static int ip, ip2, ix, iy;
00021
00022   /* Allocate... */
00023   ALLOC(met, met_t, 1);
00024
00025   /* Check arguments... */
00026   if (argc < 4)
00027     ERRMSG("Give parameters: <ctl> <map.tab> <met>");
00028
00029   /* Read control parameters... */
00030   read_ctl(argv[1], argc, argv, &ctl);
00031   z = scan_ctl(argv[1], argc, argv, "Z", -1, "", NULL);
00032
00033   /* Read meteorological data... */
00034   read_met(&ctl, argv[3], met);
00035
00036   /* Find nearest pressure level... */
00037   for (ip2 = 0; ip2 < met->np; ip2++) {
00038     dz = fabs(Z(met->p[ip2]) - z);
00039     if (dz < dzmin) {
00040       dzmin = dz;
00041       ip = ip2;
00042     }
00043   }
00044
00045   /* Write info... */
00046   printf("Analyze %g hPa...\n", met->p[ip]);
00047
00048   /* Calculate horizontal diffusion coefficients... */
00049   for (ix = 1; ix < met->nx - 1; ix++)
00050     for (iy = 1; iy < met->ny - 1; iy++) {
00051       t = 0.5 * ((met->u[ix + 1][iy][ip] - met->u[ix - 1][iy][ip])
```

```
00052                   / (1000. *
00053                      deg2dx(met->lon[ix + 1] - met->lon[ix - 1], met->
    lat[iy]))
00054                   - (met->v[ix][iy + 1][ip] - met->v[ix][iy - 1][ip])
00055                   / (1000. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1])));
00056       s = 0.5 * ((met->u[ix][iy + 1][ip] - met->u[ix][iy - 1][ip])
00057                  / (1000. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1]))
00058                  + (met->v[ix + 1][iy][ip] - met->v[ix - 1][iy][ip])
00059                  / (1000. *
00060                     deg2dx(met->lon[ix + 1] - met->lon[ix - 1],
00061                            met->lat[iy])));
00062       ls2 = gsl_pow_2(c * 500. * deg2dy(met->lat[iy + 1] - met->lat[iy - 1]));
00063       if (fabs(met->lat[iy]) > 80)
00064         ls2 *= (90. - fabs(met->lat[iy])) / 10.;
00065       k[ix][iy] = ls2 * sqrt(2.0 * (gsl_pow_2(t) + gsl_pow_2(s)));
00066     }
00067
00068   /* Create output file... */
00069   printf("Write data file: %s\n", argv[2]);
00070   if (!(out = fopen(argv[2], "w")))
00071     ERRMSG("Cannot create file!");
00072
00073   /* Write header... */
00074   fprintf(out,
00075           "# $1 = longitude [deg]\n"
00076           "# $2 = latitude [deg]\n"
00077           "# $3 = zonal wind [m/s]\n"
00078           "# $4 = meridional wind [m/s]\n"
00079           "# $5 = horizontal diffusivity [m^2/s]\n");
00080
00081   /* Write data... */
00082   for (iy = 0; iy < met->ny; iy++) {
00083     fprintf(out, "\n");
00084     for (ix = 0; ix < met->nx; ix++)
00085       if (met->lon[ix] >= 180)
00086         fprintf(out, "%g %g %g %g %g\n",
00087                 met->lon[ix] - 360.0, met->lat[iy],
00088                 met->u[ix][iy][ip], met->v[ix][iy][ip], k[ix][iy]);
00089     for (ix = 0; ix < met->nx; ix++)
00090       if (met->lon[ix] <= 180)
00091         fprintf(out, "%g %g %g %g %g\n",
00092                 met->lon[ix], met->lat[iy],
00093                 met->u[ix][iy][ip], met->v[ix][iy][ip], k[ix][iy]);
00094   }
00095
00096   /* Close file... */
00097   fclose(out);
00098
00099   /* Free... */
00100   free(met);
00101
00102   return EXIT_SUCCESS;
00103 }
```

5.27 split.c File Reference

Split air parcels into a larger number of parcels.

Functions

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

5.27.1 Detailed Description

Split air parcels into a larger number of parcels.

Definition in file split.c.

5.27.2 Function Documentation

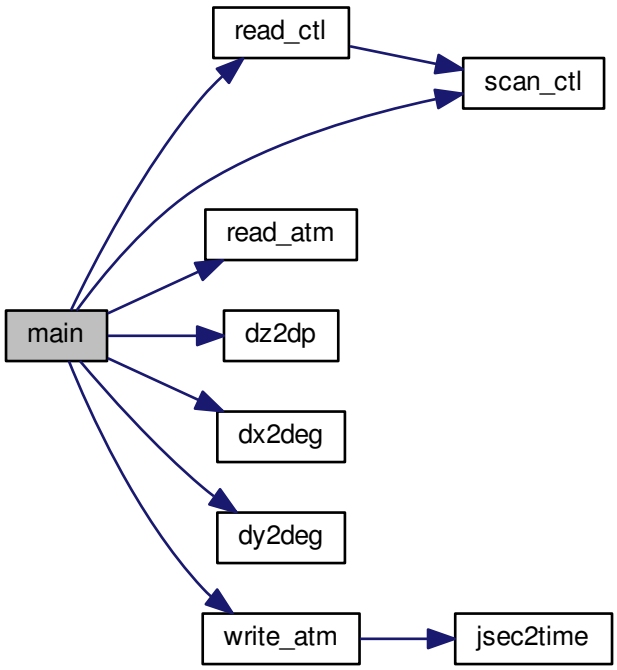
5.27.2.1 int main (int *argc*, char * *argv[]*)

Definition at line 27 of file split.c.

```
00029                {
00030
00031   atm_t *atm, *atm2;
00032
00033   ctl_t ctl;
00034
00035   gsl_rng *rng;
00036
00037   double m, mtot = 0, dt, dx, dz, mmax = 0,
00038     t0, t1, z0, z1, lon0, lon1, lat0, lat1;
00039
00040   int i, ip, iq, n;
00041
00042   /* Allocate... */
00043   ALLOC(atm, atm_t, 1);
00044   ALLOC(atm2, atm_t, 1);
00045
00046   /* Check arguments... */
00047   if (argc < 4)
00048     ERRMSG("Give parameters: <ctl> <atm_in> <atm_out>");
00049
00050   /* Read control parameters... */
00051   read_ctl(argv[1], argc, argv, &ctl);
00052   n = (int) scan_ctl(argv[1], argc, argv, "SPLIT_N", -1, "", NULL);
00053   m = scan_ctl(argv[1], argc, argv, "SPLIT_M", -1, "-999", NULL);
00054   dt = scan_ctl(argv[1], argc, argv, "SPLIT_DT", -1, "0", NULL);
00055   t0 = scan_ctl(argv[1], argc, argv, "SPLIT_T0", -1, "0", NULL);
00056   t1 = scan_ctl(argv[1], argc, argv, "SPLIT_T1", -1, "0", NULL);
00057   dz = scan_ctl(argv[1], argc, argv, "SPLIT_DZ", -1, "0", NULL);
00058   z0 = scan_ctl(argv[1], argc, argv, "SPLIT_Z0", -1, "0", NULL);
00059   z1 = scan_ctl(argv[1], argc, argv, "SPLIT_Z1", -1, "0", NULL);
00060   dx = scan_ctl(argv[1], argc, argv, "SPLIT_DX", -1, "0", NULL);
00061   lon0 = scan_ctl(argv[1], argc, argv, "SPLIT_LON0", -1, "0", NULL);
00062   lon1 = scan_ctl(argv[1], argc, argv, "SPLIT_LON1", -1, "0", NULL);
00063   lat0 = scan_ctl(argv[1], argc, argv, "SPLIT_LAT0", -1, "0", NULL);
00064   lat1 = scan_ctl(argv[1], argc, argv, "SPLIT_LAT1", -1, "0", NULL);
00065
00066   /* Init random number generator... */
00067   gsl_rng_env_setup();
00068   rng = gsl_rng_alloc(gsl_rng_default);
00069
00070   /* Read atmospheric data... */
00071   read_atm(argv[2], &ctl, atm);
00072
00073   /* Get total and maximum mass... */
00074   if (ctl.qnt_m >= 0)
00075     for (ip = 0; ip < atm->np; ip++) {
00076       mtot += atm->q[ctl.qnt_m][ip];
00077       mmax = GSL_MAX(mmax, atm->q[ctl.qnt_m][ip]);
00078     }
00079   if (m > 0)
00080     mtot = m;
00081
00082   /* Loop over air parcels... */
00083   for (i = 0; i < n; i++) {
00084
00085     /* Select air parcel... */
00086     if (ctl.qnt_m >= 0)
00087       do {
00088         ip = (int) gsl_rng_uniform_int(rng, (long unsigned int) atm->np);
00089       } while (gsl_rng_uniform(rng) > atm->q[ctl.qnt_m][ip] / mmax);
00090     else
00091       ip = (int) gsl_rng_uniform_int(rng, (long unsigned int) atm->np);
00092
00093     /* Set time... */
00094     if (t1 > t0)
00095       atm2->time[atm2->np] = t0 + (t1 - t0) * gsl_rng_uniform_pos(rng);
00096     else
00097       atm2->time[atm2->np] = atm->time[ip]
00098         + gsl_ran_gaussian_ziggurat(rng, dt / 2.3548);
00099
00100    /* Set vertical position... */
00101    if (z1 > z0)
00102      atm2->p[atm2->np] = P(z0 + (z1 - z0) * gsl_rng_uniform_pos(rng));
00103    else
00104      atm2->p[atm2->np] = atm->p[ip]
```

```
00105         + dz2dp(gsl_ran_gaussian_ziggurat(rng, dz / 2.3548), atm->p[ip]);
00106
00107     /* Set horizontal position... */
00108     if (lon1 > lon0 && lat1 > lat0) {
00109       atm2->lon[atm2->np] = lon0 + (lon1 - lon0) * gsl_rng_uniform_pos(rng);
00110       atm2->lat[atm2->np] = lat0 + (lat1 - lat0) * gsl_rng_uniform_pos(rng);
00111     } else {
00112       atm2->lon[atm2->np] = atm->lon[ip]
00113         + gsl_ran_gaussian_ziggurat(rng, dx2deg(dx, atm->lat[ip]) / 2.3548);
00114       atm2->lat[atm2->np] = atm->lat[ip]
00115         + gsl_ran_gaussian_ziggurat(rng, dy2deg(dx) / 2.3548);
00116     }
00117
00118     /* Copy quantities... */
00119     for (iq = 0; iq < ctl.nq; iq++)
00120       atm2->q[iq][atm2->np] = atm->q[iq][ip];
00121
00122     /* Adjust mass... */
00123     if (ctl.qnt_m >= 0)
00124       atm2->q[ctl.qnt_m][atm2->np] = mtot / n;
00125
00126     /* Increment particle counter... */
00127     if ((++atm2->np) >= NP)
00128       ERRMSG("Too many air parcels!");
00129   }
00130
00131   /* Save data and close file... */
00132   write_atm(argv[3], &ctl, atm2, atm->time[0]);
00133
00134   /* Free... */
00135   free(atm);
00136   free(atm2);
00137
00138   return EXIT_SUCCESS;
00139 }
```

Here is the call graph for this function:



5.28 split.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 int main(
00028   int argc,
00029   char *argv[]) {
00030
00031   atm_t *atm, *atm2;
00032
00033   ctl_t ctl;
00034
00035   gsl_rng *rng;
00036
00037   double m, mtot = 0, dt, dx, dz, mmax = 0,
00038     t0, t1, z0, z1, lon0, lon1, lat0, lat1;
00039
00040   int i, ip, iq, n;
00041
00042   /* Allocate... */
00043   ALLOC(atm, atm_t, 1);
00044   ALLOC(atm2, atm_t, 1);
00045
00046   /* Check arguments... */
00047   if (argc < 4)
00048     ERRMSG("Give parameters: <ctl> <atm_in> <atm_out>");
00049
00050   /* Read control parameters... */
00051   read_ctl(argv[1], argc, argv, &ctl);
00052   n = (int) scan_ctl(argv[1], argc, argv, "SPLIT_N", -1, "", NULL);
00053   m = scan_ctl(argv[1], argc, argv, "SPLIT_M", -1, "-999", NULL);
00054   dt = scan_ctl(argv[1], argc, argv, "SPLIT_DT", -1, "0", NULL);
00055   t0 = scan_ctl(argv[1], argc, argv, "SPLIT_T0", -1, "0", NULL);
00056   t1 = scan_ctl(argv[1], argc, argv, "SPLIT_T1", -1, "0", NULL);
00057   dz = scan_ctl(argv[1], argc, argv, "SPLIT_DZ", -1, "0", NULL);
00058   z0 = scan_ctl(argv[1], argc, argv, "SPLIT_Z0", -1, "0", NULL);
00059   z1 = scan_ctl(argv[1], argc, argv, "SPLIT_Z1", -1, "0", NULL);
00060   dx = scan_ctl(argv[1], argc, argv, "SPLIT_DX", -1, "0", NULL);
00061   lon0 = scan_ctl(argv[1], argc, argv, "SPLIT_LON0", -1, "0", NULL);
00062   lon1 = scan_ctl(argv[1], argc, argv, "SPLIT_LON1", -1, "0", NULL);
00063   lat0 = scan_ctl(argv[1], argc, argv, "SPLIT_LAT0", -1, "0", NULL);
00064   lat1 = scan_ctl(argv[1], argc, argv, "SPLIT_LAT1", -1, "0", NULL);
00065
00066   /* Init random number generator... */
00067   gsl_rng_env_setup();
00068   rng = gsl_rng_alloc(gsl_rng_default);
00069
00070   /* Read atmospheric data... */
00071   read_atm(argv[2], &ctl, atm);
00072
00073   /* Get total and maximum mass... */
00074   if (ctl.qnt_m >= 0)
00075     for (ip = 0; ip < atm->np; ip++) {
00076       mtot += atm->q[ctl.qnt_m][ip];
00077       mmax = GSL_MAX(mmax, atm->q[ctl.qnt_m][ip]);
00078     }
00079   if (m > 0)
00080     mtot = m;
00081
00082   /* Loop over air parcels... */
00083   for (i = 0; i < n; i++) {
00084
00085     /* Select air parcel... */
00086     if (ctl.qnt_m >= 0)
00087       do {
00088         ip = (int) gsl_rng_uniform_int(rng, (long unsigned int) atm->np);
00089       } while (gsl_rng_uniform(rng) > atm->q[ctl.qnt_m][ip] / mmax);
```

```
00090     else
00091       ip = (int) gsl_rng_uniform_int(rng, (long unsigned int) atm->np);
00092
00093     /* Set time... */
00094     if (t1 > t0)
00095       atm2->time[atm2->np] = t0 + (t1 - t0) * gsl_rng_uniform_pos(rng);
00096     else
00097       atm2->time[atm2->np] = atm->time[ip]
00098         + gsl_ran_gaussian_ziggurat(rng, dt / 2.3548);
00099
00100    /* Set vertical position... */
00101    if (z1 > z0)
00102      atm2->p[atm2->np] = P(z0 + (z1 - z0) * gsl_rng_uniform_pos(rng));
00103    else
00104      atm2->p[atm2->np] = atm->p[ip]
00105        + dz2dp(gsl_ran_gaussian_ziggurat(rng, dz / 2.3548), atm->p[ip]);
00106
00107    /* Set horizontal position... */
00108    if (lon1 > lon0 && lat1 > lat0) {
00109      atm2->lon[atm2->np] = lon0 + (lon1 - lon0) * gsl_rng_uniform_pos(rng);
00110      atm2->lat[atm2->np] = lat0 + (lat1 - lat0) * gsl_rng_uniform_pos(rng);
00111    } else {
00112      atm2->lon[atm2->np] = atm->lon[ip]
00113        + gsl_ran_gaussian_ziggurat(rng, dx2deg(dx, atm->lat[ip]) / 2.3548);
00114      atm2->lat[atm2->np] = atm->lat[ip]
00115        + gsl_ran_gaussian_ziggurat(rng, dy2deg(dx) / 2.3548);
00116    }
00117
00118    /* Copy quantities... */
00119    for (iq = 0; iq < ctl.nq; iq++)
00120      atm2->q[iq][atm2->np] = atm->q[iq][ip];
00121
00122    /* Adjust mass... */
00123    if (ctl.qnt_m >= 0)
00124      atm2->q[ctl.qnt_m][atm2->np] = mtot / n;
00125
00126    /* Increment particle counter... */
00127    if ((++atm2->np) >= NP)
00128      ERRMSG("Too many air parcels!");
00129  }
00130
00131  /* Save data and close file... */
00132  write_atm(argv[3], &ctl, atm2, atm->time[0]);
00133
00134  /* Free... */
00135  free(atm);
00136  free(atm2);
00137
00138  return EXIT_SUCCESS;
00139 }
```

5.29 time2jsec.c File Reference

Convert date to Julian seconds.

Functions

- int main (int argc, char ∗argv[ ])

5.29.1 Detailed Description

Convert date to Julian seconds.

Definition in file time2jsec.c.

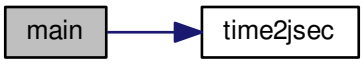
5.29.2 Function Documentation

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

Definition at line 27 of file time2jsec.c.

```
00029                {
00030
00031   double jsec, remain;
00032
00033   int day, hour, min, mon, sec, year;
00034
00035   /* Check arguments... */
00036   if (argc < 8)
00037     ERRMSG("Give parameters: <year> <mon> <day> <hour> <min> <sec> <remain>");
00038
00039   /* Read arguments... */
00040   year = atoi(argv[1]);
00041   mon = atoi(argv[2]);
00042   day = atoi(argv[3]);
00043   hour = atoi(argv[4]);
00044   min = atoi(argv[5]);
00045   sec = atoi(argv[6]);
00046   remain = atof(argv[7]);
00047
00048   /* Convert... */
00049   time2jsec(year, mon, day, hour, min, sec, remain, &jsec);
00050   printf("%.2f\n", jsec);
00051
00052   return EXIT_SUCCESS;
00053 }
```

Here is the call graph for this function:



5.30 time2jsec.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 int main(
00028   int argc,
00029   char *argv[]) {
```

```
00030
00031   double jsec, remain;
00032
00033   int day, hour, min, mon, sec, year;
00034
00035   /* Check arguments... */
00036   if (argc < 8)
00037     ERRMSG("Give parameters: <year> <mon> <day> <hour> <min> <sec> <remain>");
00038
00039   /* Read arguments... */
00040   year = atoi(argv[1]);
00041   mon = atoi(argv[2]);
00042   day = atoi(argv[3]);
00043   hour = atoi(argv[4]);
00044   min = atoi(argv[5]);
00045   sec = atoi(argv[6]);
00046   remain = atof(argv[7]);
00047
00048   /* Convert... */
00049   time2jsec(year, mon, day, hour, min, sec, remain, &jsec);
00050   printf("%.2f\n", jsec);
00051
00052   return EXIT_SUCCESS;
00053 }
```

5.31 topo.c File Reference

Terrain height variances.

5.31.1 Detailed Description

Terrain height variances.

Definition in file topo.c.

5.32 topo.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2018 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #define TOPO_NLAT 181
00026 #define TOPO_NLON 361
00027
00028 static double topo_lat[TOPO_NLAT] =
00029   { -90, -89, -88, -87, -86, -85, -84, -83, -82, -81, -80, -79, -78, -77, -76,
00030   -75, -74, -73, -72, -71, -70, -69, -68, -67, -66, -65, -64, -63, -62, -61,
00031   -60, -59, -58, -57,
00032   -56, -55, -54, -53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42,
00033   -41, -40, -39, -38,
00034   -37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23,
00035   -22, -21, -20, -19,
00036   -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2,
00037   -1, 0, 1, 2, 3, 4,
00038   5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
00039   25, 26, 27, 28, 29,
00040   30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
```

```
00041   49, 50, 51, 52, 53,
00042   54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
00043   73, 74, 75, 76, 77,
00044   78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90
00045 };
00046
00047 static double topo_lon[TOPO_NLON] =
00048   { -180, -179, -178, -177, -176, -175, -174, -173, -172, -171, -170, -169,
00049   -168, -167, -166, -165, -164, -163, -162, -161, -160, -159, -158, -157,
00050   -156, -155, -154, -153,
00051   -152, -151, -150, -149, -148, -147, -146, -145, -144, -143, -142, -141,
00052   -140, -139, -138, -137,
00053   -136, -135, -134, -133, -132, -131, -130, -129, -128, -127, -126, -125,
00054   -124, -123, -122, -121,
00055   -120, -119, -118, -117, -116, -115, -114, -113, -112, -111, -110, -109,
00056   -108, -107, -106, -105,
00057   -104, -103, -102, -101, -100, -99, -98, -97, -96, -95, -94, -93, -92, -91,
00058   -90, -89, -88, -87,
00059   -86, -85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72,
00060   -71, -70, -69, -68,
00061   -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54, -53,
00062   -52, -51, -50, -49,
00063   -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38, -37, -36, -35, -34,
00064   -33, -32, -31, -30,
00065   -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17, -16, -15,
00066   -14, -13, -12, -11,
00067   -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
00068   11, 12, 13, 14, 15,
00069   16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
00070   35, 36, 37, 38, 39,
00071   40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
00072   59, 60, 61, 62, 63,
00073   64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
00074   83, 84, 85, 86, 87,
00075   88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104,
00076   105, 106, 107, 108, 109,
00077   110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124,
00078   125, 126, 127, 128,
00079   129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
00080   144, 145, 146, 147,
00081   148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162,
00082   163, 164, 165, 166,
00083   167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180
00084 };
00085
00086 static double topo_zvar[TOPO_NLON][TOPO_NLAT] = {
00087   {1.95e+03, 4.62e+03, 834, 2.3e+03, 1.88e+04, 3.27e+05, 1.77e+04, 0, 0, 0, 0,
00088    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00089    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00090    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 190, 0, 4.95e+03, 0.00222, 0, 0, 0, 0, 0, 0,
00091    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00092    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00093    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00094    415, 3.96e+04, 2.33e+04, 6.03e+04, 2.24e+04, 0, 1.65e+04, 13.9, 0, 0, 0, 0,
00095    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00096   {1.87e+03, 4.67e+03, 975, 2.3e+03, 2.5e+04, 2.65e+05, 5.9e+04, 0, 0, 0, 0,
00097    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00098    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00099    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 913, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00100    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00101    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00102    0, 0, 0, 0, 0, 0, 0.00111, 587, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.145,
00103    2.33e+04, 3.82e+04, 4.12e+04, 1.02e+03, 0, 2.28e+04, 16.9, 0, 0, 0, 0, 0,
00104    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00105   {1.78e+03, 4.72e+03, 1.2e+03, 2.24e+03, 2.56e+04, 1.88e+05, 1.34e+04, 0, 0,
00106    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00107    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00108    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00109    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00110    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00111    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.32e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00112    1.85, 1.54e+04, 4.36e+04, 1.84e+04, 0.422, 0, 3.53e+03, 1.99, 0, 0, 0, 0,
00113    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00114   {1.68e+03, 4.76e+03, 1.49e+03, 2.21e+03, 2.5e+04, 1.98e+05, 3.35e+04, 0, 0,
00115    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00116    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 295, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00117    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00118    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00119    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00120    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.14e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00121    0.0588, 1.94e+04, 1.82e+04, 1.29e+03, 0, 0, 0.0555, 0, 0, 0, 0, 0, 0, 0, 0,
00122    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00123   {1.59e+03, 4.8e+03, 1.82e+03, 2.17e+03, 1.94e+04, 2.95e+05, 1.76e+03, 0, 0,
00124    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00125    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 50.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00126    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00127    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
00128    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00129    0, 0, 0, 0, 0, 0, 0, 0, 0, 5.04e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00130    1.06e+04, 1.5e+04, 1.87e+04, 500, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00131    0, 0, 0, 0, 0, 0, 0, 0},
00132   {1.51e+03, 4.85e+03, 2.18e+03, 2.35e+03, 1.41e+04, 3.94e+05, 946, 0, 0, 0,
00133    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00134    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00135    0, 0, 0, 0, 0, 0, 0, 0, 0, 5.12, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00136    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00137    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00138    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00139    2.85e+04, 8.37e+03, 5.5e+03, 427, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00140    0, 0, 0, 0, 0, 0, 0, 0},
00141   {1.43e+03, 4.88e+03, 2.59e+03, 3.56e+03, 6.54e+03, 8.3e+05, 603, 0, 0, 0, 0,
00142    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00143    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00144    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00145    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00146    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00147    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.05e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00148    3.52e+03, 2.88e+04, 1.07e+04, 1.72e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00149    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00150   {1.35e+03, 4.91e+03, 3.07e+03, 5.93e+03, 6.11e+03, 8.28e+05, 96.4, 0, 0, 0,
00151    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00152    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00153    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.23e+04, 0, 0, 0, 0, 0, 0,
00154    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00155    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00156    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.72e+03, 0, 0, 0, 0, 0, 0, 0, 281, 99.6, 0, 0,
00157    6.94e+03, 3.46e+04, 1.64e+04, 1.59e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00158    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00159   {1.29e+03, 4.94e+03, 3.6e+03, 1e+04, 7.29e+03, 1.36e+06, 0.00111, 0, 0, 0,
00160    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00161    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00162    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.13e+04, 107, 0, 0, 0, 0,
00163    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00164    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00165    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1e+03, 0, 0, 0, 0, 0, 0, 0, 181, 0, 0,
00166    1.43e+03, 429, 1.41e+03, 3.4e+04, 1.78e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00167    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00168   {1.22e+03, 4.97e+03, 4.17e+03, 1.23e+04, 1.36e+04, 1.28e+06, 0, 0, 0, 0, 0,
00169    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00170    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00171    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 93.5, 0, 0, 0, 0, 0, 0, 0,
00172    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00173    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00174    0, 0, 0, 0, 0, 0, 0, 0, 0, 698, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 2.96e+03,
00175    0.805, 2.78e+04, 2.08e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00176    0, 0, 0, 0, 0, 0, 0, 0},
00177   {1.17e+03, 5.01e+03, 4.76e+03, 1.44e+04, 2.86e+04, 6.95e+05, 0, 0, 0, 0, 0,
00178    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00179    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00180    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.92, 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, 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, 0, 0,
00183    0, 0, 0, 0, 0, 0, 0, 0, 0, 4.16e+03, 0, 0, 0, 188, 0, 0, 0, 0, 0, 715,
00184    3.89e+03, 0, 4.13e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00185    0, 0, 0, 0, 0, 0, 0, 0},
00186   {1.12e+03, 5.03e+03, 5.35e+03, 1.7e+04, 7.2e+04, 8.09e+05, 0, 0, 0, 0, 0, 0,
00187    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00188    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00189    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00190    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00191    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00192    0, 0, 0, 0, 0, 0, 1.86e+04, 0, 0, 0, 0, 0, 0, 0, 0, 1.37e+03, 0, 0, 627,
00193    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00194   {1.07e+03, 5.07e+03, 5.88e+03, 2.16e+04, 1.18e+05, 1.26e+06, 0, 0, 0, 0, 0,
00195    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00196    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00197    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00198    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00199    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00200    0, 0, 0, 0, 0, 0, 0, 1.19e+04, 3.26e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 282,
00201    2.7e+03, 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},
00203   {1.02e+03, 5.13e+03, 6.34e+03, 2.98e+04, 1.4e+05, 1.08e+06, 0, 0, 0, 0, 0,
00204    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00205    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00206    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00207    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00208    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00209    0, 0, 0, 0, 0, 0, 0, 885, 4.49e+04, 0, 0, 0, 0, 0, 2.23e+03, 0, 0, 0, 0,
00210    1.97e+03, 9.47e+03, 0, 0.29, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00211    0, 0, 0, 0, 0, 0, 0, 0},
00212   {982, 5.18e+03, 6.7e+03, 3.74e+04, 1.75e+05, 6.06e+05, 0, 0, 0, 0, 0, 0, 0,
00213    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00214    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
00215    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00216    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00217    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00218    0, 0, 0, 0, 0, 0, 9.14e+03, 0, 0, 0, 0, 0, 4.13e+03, 0.373, 3.14e+03, 0, 0,
00219    2.63e+04, 1.26e+04, 0, 3.48e+03, 1.49e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00220    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00221   {938, 5.22e+03, 6.98e+03, 4.09e+04, 2.97e+05, 3.44e+05, 0, 0, 0, 0, 0, 0, 0,
00222    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00223    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00224    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00225    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00226    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00227    0, 0, 0, 0, 0, 0, 1.07e+03, 3.07e+04, 0, 0, 0, 0, 0.582, 2.81e+03,
00228    1.41e+03, 0.504, 20.1, 3.09e+04, 2.41e+04, 1.76, 8.38e+03, 7.96e+03, 0, 0,
00229    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00230   {894, 5.24e+03, 7.24e+03, 5.06e+04, 5.06e+05, 1.7e+05, 0, 0, 0, 0, 0, 100,
00231    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00232    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00233    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00234    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00235    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00236    0, 0, 0, 0, 0, 0, 0, 0.98, 5.86e+04, 0, 0, 0, 0, 24.5, 231, 531, 8.69, 0,
00237    2.97e+04, 1.25e+04, 527, 2.28e+04, 2.86e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00238    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00239   {860, 5.23e+03, 7.47e+03, 5.87e+04, 2.44e+05, 6.08e+04, 0, 0, 0, 0, 212,
00240    1.18e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00241    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00242    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00243    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00244    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00245    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.88, 1.66e+04, 0, 0, 0, 0, 18.7, 58,
00246    2.71e+03, 8.16e+03, 1.18e+03, 3.97e+04, 2.25e+04, 3.43e+03, 4.95e+04,
00247    3.98e+04, 78.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00248    0},
00249   {825, 5.19e+03, 7.59e+03, 5.94e+04, 3.13e+05, 2.55e+04, 0.135, 0, 0, 0,
00250    5.86e+03, 3.04e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00251    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00252    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00253    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00254    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00255    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.74e+04, 6.91e+03, 0, 0,
00256    5.09e+03, 2.22e+03, 58.7, 5.5e+03, 1.92e+04, 49.1, 1.73e+04, 1.32e+04,
00257    1.57e+04, 4.32e+04, 4.15e+04, 620, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00258    0, 0, 0, 0, 0, 0, 0, 0},
00259   {789, 5.12e+03, 7.62e+03, 6.12e+04, 2.61e+05, 1.27e+04, 3.18, 0, 0, 0,
00260    1.83e+04, 2.22e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00261    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00262    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00263    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00264    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00265    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.08e+03, 2.07e+04, 0, 0,
00266    1.58e+04, 3.19e+04, 3.24e+03, 9.64e+03, 2.51e+04, 1.79e+04, 1.13e+04,
00267    1.69e+04, 1.59e+04, 4.27e+04, 5.15e+04, 1.95e+03, 0, 0, 0, 0, 0, 0, 0, 0,
00268    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00269   {759, 5.02e+03, 7.51e+03, 5.96e+04, 3.62e+05, 9.69e+03, 23.9, 0.011, 0, 0,
00270    1.65e+04, 4.47e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00271    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00272    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.61, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00273    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00274    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.96e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00275    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 437, 7.46e+04, 22.4,
00276    0, 2.09e+04, 5.01e+04, 7.92e+04, 2.85e+03, 2.31e+04, 2.52e+04, 1.93e+04,
00277    2.23e+04, 3.55e+04, 2.97e+04, 5.53e+04, 2.22e+03, 37.2, 0, 0, 0, 0, 0, 0,
00278    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00279   {725, 4.88e+03, 7.29e+03, 4.45e+04, 9.99e+05, 6.84e+03, 63.6, 4.57, 0, 0,
00280    5.12e+03, 148, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00281    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00282    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00283    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00284    0, 0, 0, 0, 0, 0, 2.9e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00285    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.403, 1.24e+05, 1.7e+04,
00286    2.39, 1.69e+04, 8.33e+04, 4.42e+04, 1.35e+04, 3.63e+03, 2.63e+04, 2.21e+04,
00287    1.87e+04, 1.82e+04, 3.01e+04, 4.17e+04, 1.93e+03, 79.8, 0, 0, 0, 0, 0, 0,
00288    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00289   {692, 4.73e+03, 7e+03, 3.33e+04, 1.06e+06, 6.62e+03, 114, 41.3, 0, 0, 17.9,
00290    0, 6.4e+03, 4.16e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00291    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00292    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00293    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00294    0, 0, 0, 0, 3.88e+03, 6.09e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00295    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.44e+03,
00296    3.54e+04, 32.7, 2.66e+03, 1.26e+04, 2.21e+04, 1.3e+04, 1.06e+04, 1.58e+04,
00297    1.95e+04, 1.71e+04, 2.78e+04, 3.19e+04, 1.97e+04, 1.7e+03, 116, 0, 0, 0, 0,
00298    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00299   {662, 4.56e+03, 6.67e+03, 2.34e+04, 8.46e+05, 1.28e+04, 198, 157, 0, 24.5,
00300    0, 0, 6e+04, 4.45e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00301    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
00302    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00303    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00304    0, 0, 0, 0, 0, 2.06e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00305    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.98e+04, 2.15e+03,
00306    876, 5.78e+03, 9.51e+03, 1.19e+04, 2.47e+04, 1.59e+04, 6.2e+03, 3.61e+04,
00307    5.24e+04, 3.94e+04, 1.82e+04, 2.59e+03, 67.8, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00308    0, 0, 0, 0, 0, 0, 0, 0, 0},
00309   {632, 4.37e+03, 6.36e+03, 2.28e+04, 6.07e+05, 5.39e+03, 349, 431, 0,
00310    2.73e+03, 0, 0, 1.03e+05, 7.8e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00311    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00312    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00313    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00314    0, 0, 0, 0, 0, 0, 0, 5.2e+05, 6.78e+05, 1.21e+05, 0, 0, 0, 0, 0, 0, 0, 0,
00315    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00316    0, 0.00552, 1.71e+03, 5.65e+04, 1.82e+04, 1.5e+04, 1.41e+04, 8.74e+03,
00317    2.11e+04, 1.07e+04, 1.75e+04, 4.12e+04, 1.39e+05, 7.02e+04, 1.48e+04,
00318    1.2e+03, 30.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00319   {603, 4.16e+03, 6.2e+03, 3.64e+04, 4.22e+05, 2.99e+03, 627, 935, 0,
00320    1.15e+04, 0, 1.44e+03, 1.21e+05, 9.21e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00321    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00322    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00323    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00324    0, 0, 0, 0, 0, 0, 0, 0, 0, 7.81e+04, 4.12e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00325    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00326    0, 1.8, 744, 1.8e+05, 8.88e+04, 5.12e+04, 2.64e+04, 2.48e+04, 8.92e+03,
00327    1.63e+04, 3.53e+04, 1.32e+04, 1.5e+05, 8.88e+04, 2.65e+04, 1.14e+03, 17.6,
00328    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00329   {573, 3.94e+03, 6.44e+03, 5.44e+04, 2.44e+05, 379, 1.08e+03, 1.71e+03,
00330    1.01e+03, 2.17e+04, 0, 1.6e+04, 9.06e+04, 3.67e+04, 0, 0, 0, 0, 0, 0, 0, 0,
00331    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00332    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00333    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00334    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00335    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00336    5.1e+04, 5.36e+04, 1.02e+05, 1.7e+05, 1.98e+05, 1.88e+05, 1.18e+04,
00337    3.12e+04, 7.85e+04, 2.33e+04, 1.08e+05, 1.19e+05, 3.43e+04, 958, 3.13, 0,
00338    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00339   {548, 3.75e+03, 7.59e+03, 4.36e+04, 2.62e+05, 865, 1.62e+03, 2.56e+03,
00340    1.2e+04, 2.6e+04, 0, 3.42e+04, 2.01e+04, 4.5e+03, 0, 0, 0, 0, 0, 0, 0, 0,
00341    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00342    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00343    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00344    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00345    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00346    3.16e+04, 5.63e+04, 5e+03, 2e+05, 2.13e+05, 1.74e+05, 1.49e+05, 1.37e+04,
00347    2.01e+04, 1.19e+04, 8.14e+04, 7.6e+04, 2.66e+04, 2.5e+03, 8.52, 0, 0, 0, 0,
00348    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00349   {516, 3.6e+03, 8.88e+03, 5.23e+04, 2.31e+05, 1.98e+03, 1.72e+03, 2.5e+03,
00350    2.45e+04, 2.99e+04, 0, 4.59e+04, 1.91e+04, 5.55e+03, 0, 0, 0, 0, 0, 0, 0,
00351    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00352    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00353    0, 0, 2.67, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00354    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00355    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00356    0, 0, 13.1, 3.05e+03, 1.3e+04, 5.97e+03, 3.3e+05, 2.66e+05, 3.4e+05,
00357    1.38e+04, 2.73e+04, 4.69e+04, 6.91e+04, 8.94e+04, 3.33e+04, 1.68e+03, 6.17,
00358    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00359   {489, 3.47e+03, 1.01e+04, 7.65e+04, 1.28e+05, 2.24e+03, 1.5e+03, 1.77e+03,
00360    2.89e+04, 3.55e+04, 52.2, 3.14e+04, 6.24e+04, 5.35e+03, 0, 0, 0, 0, 0, 0,
00361    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00362    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00363    0, 0, 0, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00364    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00365    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00366    0, 0, 0, 0, 0, 0, 3.35e+04, 1.36e+05, 8.84e+03, 3.27e+04, 8.18e+05, 5.98e+04,
00367    2.08e+04, 5.56e+04, 6.37e+04, 1.31e+05, 4.22e+04, 2.04e+03, 0.0099, 0, 0,
00368    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00369   {466, 3.37e+03, 1.16e+04, 8.94e+04, 1.39e+05, 2.84e+03, 1.52e+03, 1.6e+03,
00370    2.96e+04, 3.9e+04, 3.41e+03, 1.37e+04, 1.25e+05, 6.61e+03, 159, 0, 0, 0, 0,
00371    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00372    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00373    0, 0, 0, 9.72e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00374    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00375    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00376    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.73e+05, 1.38e+05, 8.91e+04, 2.86e+05, 1.48e+05,
00377    5.23e+04, 3.09e+04, 6.11e+04, 1.39e+05, 5.49e+04, 3.59e+03, 0.444, 0, 0, 0,
00378    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00379   {444, 3.28e+03, 1.17e+04, 8.25e+04, 1.6e+05, 5.86e+03, 2.39e+03, 3.6e+03,
00380    2.86e+04, 3.42e+04, 1.17e+04, 8.18e+03, 1.22e+05, 7.01e+03, 7.11e+03, 0, 0,
00381    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00382    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00383    134, 0, 0, 0, 0, 8.42e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00384    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00385    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00386    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.22e+05, 2.4e+05, 3.53e+05, 9.47e+04,
00387    1.73e+05, 1.7e+04, 1.93e+04, 7.48e+04, 1.17e+05, 7.78e+04, 4.2e+03, 1.05,
00388    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

```
00389   {420, 3.18e+03, 1.13e+04, 6.88e+04, 2.03e+05, 9.63e+03, 2.6e+03, 5.85e+03,
00390    2.85e+04, 2.43e+04, 1.94e+04, 6.97e+03, 7.67e+04, 1.19e+04, 1.59e+04, 0, 0,
00391    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00392    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00393    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00394    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00395    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00396    0, 0, 0, 0, 0, 0, 0, 0, 2.17e+04, 5.76e+05, 2.08e+05, 5.59e+04, 2.95e+05,
00397    2.98e+04, 4.03e+04, 5.62e+04, 1.38e+05, 1.39e+05, 6.54e+03, 0, 0, 0, 0, 0,
00398    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00399   {396, 3.05e+03, 1.06e+04, 9.89e+04, 3.51e+05, 8.16e+03, 2.14e+03, 7.3e+03,
00400    2.83e+04, 1.27e+04, 1.93e+04, 5.21e+03, 3.59e+04, 1.42e+04, 1.12e+04, 0, 0,
00401    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00402    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00403    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00404    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00405    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00406    0, 0, 0, 0, 0, 0, 0, 0, 8.62e+03, 6e+05, 2.27e+05, 8.85e+04, 4.73e+05,
00407    5e+04, 1.11e+05, 6.02e+04, 8.32e+04, 1.58e+05, 1.79e+04, 0, 0, 0, 0, 0, 0,
00408    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00409   {376, 2.9e+03, 9.85e+03, 7.11e+04, 4.09e+05, 1.11e+04, 1.29e+03, 8.4e+03,
00410    2.25e+04, 4.38e+03, 1.3e+04, 2.78e+03, 5.24e+04, 3.8e+04, 4.81e+04, 0, 0,
00411    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00412    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00413    0, 0, 0, 0, 0, 0, 0, 0, 0.00222, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00414    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00415    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00416    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.26e+03, 3.52e+05, 8.84e+04, 1.08e+05,
00417    2.57e+05, 4.38e+04, 8.8e+04, 2.55e+04, 5.42e+04, 1.45e+05, 7.13e+04, 0, 0,
00418    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00419   {357, 2.72e+03, 8.89e+03, 4.9e+04, 3.42e+05, 1.22e+04, 595, 8.75e+03,
00420    1.6e+04, 717, 6.06e+03, 4e+03, 4.48e+04, 3.22e+04, 1.47e+05, 0, 0, 0, 0, 0,
00421    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00422    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00423    0, 0, 0, 0.00111, 0.00222, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00424    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00425    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00426    0, 0, 0, 0, 0, 0, 0, 0, 0, 3.97e+03, 2.88e+05, 1.52e+05, 2.43e+05,
00427    2.17e+05, 5.38e+04, 4.74e+04, 1.36e+04, 4.25e+04, 1.37e+05, 1.08e+05, 0, 0,
00428    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00429   {338, 2.55e+03, 7.94e+03, 2.56e+04, 3.16e+05, 1.14e+04, 359, 7.45e+03,
00430    7.91e+03, 182, 2.2e+03, 7.38e+03, 1.34e+04, 1.54e+04, 1.37e+05, 0, 0, 0, 0,
00431    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00432    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00433    0, 0, 0, 0, 0.00442, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00434    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00435    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00436    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.29e+04, 2.63e+05, 6.23e+05, 1.43e+05,
00437    9.33e+04, 1.01e+05, 1.09e+04, 6.55e+03, 4.95e+04, 1.83e+05, 3.31e+04, 0, 0,
00438    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00439   {318, 2.39e+03, 7.22e+03, 1.67e+04, 3.02e+05, 1.12e+04, 555, 4.9e+03,
00440    3.12e+03, 182, 1.12e+03, 1.1e+04, 3.15e+03, 2.78e+04, 1.15e+05, 315, 0, 0,
00441    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00442    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00443    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00444    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00445    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00446    0, 0, 0, 0, 0, 0, 0, 1e+05, 2.34e+05, 4.02e+05, 1.57e+05, 3.68e+04,
00447    1.3e+05, 3.07e+04, 9.23e+03, 3.3e+04, 1.12e+05, 2.3e+04, 0, 0, 0, 0, 0, 0,
00448    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00449   {298, 2.25e+03, 6.66e+03, 1.53e+04, 2.97e+05, 1.24e+04, 986, 4.03e+03,
00450    1.38e+03, 463, 901, 1.26e+04, 6.51e+03, 1.5e+04, 1.59e+05, 1.64e+03, 0, 0,
00451    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00452    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00453    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00454    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00455    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00456    0, 0, 0, 0, 0, 0, 0, 0, 4.74e+05, 4.4e+05, 2.53e+05, 2.45e+04, 2.99e+04,
00457    8.15e+04, 2.05e+04, 1.41e+04, 3.17e+04, 8.02e+04, 1.37e+04, 0, 0, 0, 0, 0,
00458    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00459   {281, 2.13e+03, 6.28e+03, 1.16e+04, 3.85e+05, 9.33e+03, 719, 7.23e+03, 838,
00460    2e+03, 2.46e+03, 1.28e+04, 1.33e+04, 7.28e+03, 1.68e+05, 47, 0, 0, 0, 0, 0,
00461    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00462    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00463    0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00464    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00465    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00466    0, 0, 0, 0, 0, 0, 7.45e+05, 4.6e+05, 2.3e+05, 2.7e+04, 5.11e+04, 6.95e+04,
00467    3.79e+04, 2.53e+04, 2.89e+04, 6.93e+04, 4.97e+03, 0, 0, 0, 0, 0, 0, 0, 0,
00468    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00469   {266, 2.01e+03, 5.96e+03, 9.66e+03, 4.28e+05, 9.15e+03, 413, 1.01e+04,
00470    1.01e+03, 2.33e+03, 4.09e+03, 1.4e+04, 2.04e+04, 2.6e+03, 8.7e+04, 1.6e+04,
00471    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00472    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00473    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.02e+03, 0, 0, 0, 0, 0, 0,
00474    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00475    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
00476    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00665, 7.81e+05, 4.26e+05, 1.1e+05, 5e+04,
00477    3.86e+04, 6.02e+04, 3.47e+04, 3.34e+04, 1.26e+04, 5.93e+04, 2.84e+03, 0, 0,
00478    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00479   {252, 1.93e+03, 5.66e+03, 8.38e+03, 3.9e+05, 9.54e+03, 551, 8e+03, 1.77e+03,
00480    1.35e+03, 4.01e+03, 1.85e+04, 2.93e+04, 3.66e+03, 7.93e+04, 5.98e+04, 0, 0,
00481    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00482    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00483    0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.02e+03, 0, 0, 0, 0, 0, 0, 0, 0,
00484    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00485    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00486    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.16e+04, 6e+05, 3.78e+05, 8e+04, 4.92e+04,
00487    8.2e+04, 8.06e+04, 5.53e+04, 1.52e+04, 7.83e+03, 5.76e+04, 159, 0, 0, 0, 0,
00488    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00489   {241, 1.85e+03, 5.37e+03, 7.53e+03, 3.16e+05, 1.22e+04, 1.74e+03, 4.71e+03,
00490    602, 1.12e+03, 2.68e+03, 2.44e+04, 2.49e+04, 6.05e+03, 8.46e+04, 1.49e+05,
00491    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00492    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00493    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00494    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00495    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00496    0, 0, 0, 0, 0, 0, 0, 0, 5.99e+05, 2.02e+05, 1.67e+05, 5.41e+04, 5.98e+04,
00497    1.14e+05, 7.03e+04, 2.09e+04, 1.3e+04, 2.24e+04, 4.08e+04, 0, 0, 0, 0, 0,
00498    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00499   {226, 1.78e+03, 5.04e+03, 7.52e+03, 3.11e+05, 8.56e+03, 4.73e+03, 3.48e+03,
00500    289, 1.35e+03, 1.36e+03, 2.61e+04, 1.95e+04, 6.82e+03, 9.05e+04, 1.38e+05,
00501    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00502    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00503    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00504    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00505    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00506    0, 0, 0, 0, 0, 0, 0, 0, 7e+03, 4.33e+05, 1.52e+05, 8.69e+04, 3.87e+04,
00507    2.83e+04, 9.17e+04, 7.38e+04, 2.35e+04, 1.6e+04, 5.67e+04, 2.46e+04, 0, 0,
00508    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00509   {217, 1.71e+03, 4.67e+03, 9.29e+03, 3.54e+05, 7.91e+03, 6.33e+03, 3.8e+03,
00510    141, 1.41e+03, 533, 2.61e+04, 1.69e+04, 1.29e+04, 1.85e+05, 1.14e+05, 0, 0,
00511    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00512    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00513    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00514    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00515    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00516    0, 0, 0, 0, 3.94e+03, 5.4e+04, 2.56e+05, 1.94e+05, 7.98e+04, 3.14e+04,
00517    7.3e+04, 8.55e+04, 1.06e+05, 2.89e+04, 4.34e+04, 9.37e+04, 4.19e+03, 6, 0,
00518    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00519   {206, 1.66e+03, 4.24e+03, 1.56e+04, 4.74e+05, 8.8e+03, 6.96e+03, 3.85e+03,
00520    80.9, 1.53e+03, 914, 2.12e+04, 1.58e+04, 1.79e+04, 2.98e+05, 1.25e+05, 0,
00521    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00522    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00523    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00524    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00525    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00526    0, 0, 0, 0, 4.63e+03, 9.26e+04, 7.22e+04, 3.89e+05, 1.47e+05, 7.1e+04,
00527    5.85e+04, 8.17e+04, 8.47e+04, 1.61e+05, 2.26e+04, 2.03e+04, 3.26e+03, 110,
00528    4.45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00529   {199, 1.62e+03, 3.73e+03, 2.14e+04, 5.68e+05, 7.26e+03, 8.13e+03, 3.01e+03,
00530    337, 2.16e+03, 1.73e+03, 1.59e+04, 1.62e+04, 1.42e+04, 1.49e+05, 1.67e+05,
00531    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00532    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00533    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00534    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00535    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00536    0, 0, 0, 0, 32, 5.81e+03, 2.52e+04, 1.54e+05, 2.39e+05, 7.49e+04, 9.45e+04,
00537    9.69e+04, 7.51e+04, 8.43e+04, 1.39e+05, 4.26e+04, 2.62e+03, 1.42e+03,
00538    2.34e+03, 40.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00539    0},
00540   {189, 1.58e+03, 3.16e+03, 2.8e+04, 4.36e+05, 8.26e+03, 9.03e+03, 2.57e+03,
00541    730, 2.99e+03, 1.83e+03, 1.35e+04, 1.85e+04, 1.05e+04, 1.05e+05, 2.59e+05,
00542    1.66e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00543    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00544    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00545    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00546    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00547    0, 0, 0, 0, 4.87e+03, 1.77e+04, 1.95e+04, 3.53e+04, 2.52e+05, 3.06e+05,
00548    1.32e+05, 7.97e+04, 7.47e+04, 9.7e+04, 7.1e+04, 8.2e+04, 1.07e+05,
00549    7.56e+04, 1.66e+03, 6.26e+03, 1.24e+03, 51.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00550    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00551   {182, 1.55e+03, 2.58e+03, 3.17e+04, 2.62e+05, 1.51e+04, 9.29e+03, 2.47e+03,
00552    917, 3.88e+03, 3.45e+03, 9.69e+03, 2.43e+04, 1.09e+04, 8.23e+04, 2.96e+05,
00553    7.51e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00554    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00555    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00556    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00557    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00558    0, 0, 0, 576, 3.23e+04, 3.51e+03, 2.12e+04, 1.01e+05, 3.96e+05, 2.26e+05,
00559    7.69e+04, 6.88e+04, 4.44e+04, 7.95e+04, 5.77e+04, 8.56e+04, 1.03e+05,
00560    6.61e+04, 4.21e+03, 2.55e+03, 4.25e+03, 121, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00561    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00562   {176, 1.53e+03, 2.1e+03, 3.41e+04, 1.41e+05, 1.21e+04, 8.39e+03, 4.41e+03,
```

```
00563    340, 3.79e+03, 3.28e+03, 6.47e+03, 3.43e+04, 6.83e+03, 8.54e+04, 2.28e+05,
00564    2.29e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00565    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00566    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00567    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00568    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00569    0, 0, 0, 2.45e+03, 0.02, 826, 4e+04, 2.13e+05, 2.32e+05, 1.81e+05,
00570    4.07e+04, 4.57e+04, 4.82e+04, 3.53e+04, 5.4e+04, 7.17e+04, 2.02e+05,
00571    4.59e+04, 9.69e+03, 3.58e+03, 4.86e+03, 136, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00572    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00573   {168, 1.52e+03, 1.7e+03, 3.87e+04, 1.15e+05, 8.18e+03, 7.39e+03, 5.19e+03,
00574    420, 3.98e+03, 3.41e+03, 5.06e+03, 3.18e+04, 1.64e+03, 5.22e+04, 3.1e+05,
00575    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00576    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00577    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00578    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00579    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00580    0, 0, 4.4e+03, 1.26e+05, 2.1e+05, 2.76e+05, 1.6e+05, 1.05e+05, 7.09e+04,
00581    8.87e+04, 7.45e+04, 4.52e+04, 5.69e+04, 7.48e+04, 1.78e+05, 1.9e+04,
00582    1.03e+04, 4.46e+03, 4.92e+03, 145, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00583    0, 0, 0, 0, 0, 0, 0},
00584   {163, 1.51e+03, 1.43e+03, 4.27e+04, 2.51e+05, 7.72e+03, 5.88e+03, 5.03e+03,
00585    462, 3.86e+03, 4.83e+03, 4.86e+03, 2.04e+04, 2.39e+03, 4.04e+04, 3.13e+05,
00586    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00587    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00588    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00589    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00590    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00591    0, 159, 5.89e+04, 2.14e+05, 2.45e+05, 1.66e+05, 9.88e+04, 8.75e+04,
00592    9.18e+04, 3.31e+04, 9.58e+04, 7.69e+04, 6.23e+04, 1e+05, 2.19e+05,
00593    3.06e+03, 7.57e+03, 2.76e+03, 4.28e+03, 295, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00594    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00595   {159, 1.49e+03, 1.25e+03, 4.46e+04, 3.53e+05, 5.63e+03, 5.31e+03, 4.64e+03,
00596    727, 4.07e+03, 4.74e+03, 3.77e+03, 1.3e+04, 8.47e+03, 3.99e+04, 2.97e+05,
00597    3.7e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00598    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00599    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00600    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00601    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00602    0, 1.96e+04, 1.58e+04, 5.66e+04, 2.59e+05, 1.62e+05, 1.85e+05, 1.58e+05,
00603    1.04e+05, 7.11e+04, 1.31e+05, 4.03e+04, 1.02e+05, 1.83e+05, 9.01e+04,
00604    1.28e+05, 2.9e+05, 7.83e+03, 5.02e+03, 4.72e+03, 6.2e+03, 1.27e+03, 13.7,
00605    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00606   {154, 1.47e+03, 1.13e+03, 4.59e+04, 2.93e+05, 1.01e+04, 5.46e+03, 3.89e+03,
00607    1.61e+03, 4.83e+03, 3.89e+03, 2.91e+03, 1.08e+04, 3.55e+04, 5.47e+04,
00608    2.97e+05, 7.84e+04, 2.08e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00609    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00610    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00611    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00612    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00613    0, 0, 0, 0, 0, 0, 5.39, 1.35e+05, 1.85e+05, 2.92e+05, 2.11e+05, 6.78e+04,
00614    1.1e+05, 1.09e+05, 6.01e+04, 9.64e+04, 1.04e+05, 3.86e+04, 5.33e+04,
00615    1.92e+05, 1.38e+05, 1.33e+05, 1.27e+05, 1.92e+04, 4.55e+03, 2.64e+03,
00616    3.13e+03, 6.09e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00617    0, 0},
00618   {152, 1.42e+03, 1.11e+03, 4.66e+04, 2.78e+05, 2.42e+04, 4.6e+03, 2.64e+03,
00619    2.32e+03, 5.82e+03, 3.15e+03, 2.41e+03, 9.61e+03, 7.62e+04, 1.05e+05,
00620    3.53e+05, 9e+04, 1.79e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00621    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00622    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00623    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00624    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00625    0, 0, 0, 0, 0, 5.39e+04, 1.66e+05, 3.88e+05, 2.92e+05, 8.72e+04, 3.48e+04,
00626    5.59e+04, 9.97e+04, 1.15e+05, 1.18e+05, 1.78e+05, 4.38e+04, 2.51e+04,
00627    1.59e+05, 1.34e+05, 5.41e+04, 4.04e+04, 5.45e+03, 2e+03, 1.8e+03, 5.55e+03,
00628    1.28e+03, 0, 28, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00629   {147, 1.35e+03, 1.16e+03, 4.35e+04, 1.67e+05, 3.98e+04, 2.97e+03, 724,
00630    1.82e+03, 6.15e+03, 2.88e+03, 1.45e+03, 9.02e+03, 2.19e+04, 9.43e+04,
00631    3.27e+05, 1.18e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00632    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00633    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00634    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00635    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.97,
00636    0, 0, 0, 0, 2.26e+03, 1.1e+05, 1.63e+05, 5.1e+05, 1.62e+05, 6.37e+04,
00637    3.21e+04, 4.83e+04, 8.29e+04, 1.39e+05, 1.54e+05, 1.92e+05, 6.08e+04,
00638    7.06e+04, 6.62e+04, 8.12e+04, 3.91e+04, 1.21e+04, 6.8e+03, 3.26e+03,
00639    1.72e+03, 9e+03, 267, 0, 3.9e+03, 237, 0.387, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00640    0, 0, 0, 0, 0, 0},
00641   {144, 1.27e+03, 1.27e+03, 4.05e+04, 8.69e+04, 8.44e+04, 2.15e+03, 211,
00642    1.41e+03, 5.84e+03, 2.84e+03, 825, 1.13e+04, 1.88e+04, 7.51e+04, 3.01e+05,
00643    8.68e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00644    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00645    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00646    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00647    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.16e+04, 1.11e+05, 1.77e+05,
00648    1.95e+05, 9.37e+04, 2.72e+04, 4.39e+04, 3.41e+04, 1.95e+04, 1.88e+05,
00649    1.01e+05, 4.02e+05, 2.74e+05, 5.66e+04, 3.63e+04, 4.85e+04, 2.89e+04,
```

```
00650    1.5e+05, 1.1e+05, 1.74e+05, 6.54e+04, 5.68e+04, 1.41e+05, 8.46e+04,
00651    3.73e+04, 4.79e+04, 8.31e+03, 3.03e+03, 673, 3.72e+03, 1.57e+04, 33.1, 246,
00652    7.08e+03, 4.37e+03, 806, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00653   {139, 1.18e+03, 1.47e+03, 3.74e+04, 4.19e+04, 8.39e+04, 1.59e+03, 106,
00654    1.75e+03, 5.59e+03, 2.66e+03, 566, 1.29e+04, 1.47e+04, 4e+04, 2.76e+05,
00655    4.37e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00656    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00657    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00658    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00659    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.24e+03, 9.87e+04, 2.23e+05,
00660    2.04e+05, 1.61e+05, 1.07e+05, 8.86e+04, 3.85e+04, 3.56e+04, 2.36e+04,
00661    2.52e+05, 4.65e+04, 3.62e+05, 2.05e+05, 4.54e+04, 6.13e+04, 5.96e+04,
00662    5.12e+04, 8.3e+04, 7.05e+04, 5.89e+04, 1.15e+04, 9.86e+03, 2.79e+04,
00663    3.52e+04, 3.34e+04, 2.21e+04, 5.23e+03, 6.65e+03, 8.06e+03, 7.02e+03,
00664    1.62e+04, 3.97e+03, 3.56e+04, 5.79e+03, 2.8e+03, 3.13e+03, 0, 0.184, 0, 0,
00665    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00666   {134, 1.09e+03, 1.72e+03, 2.57e+04, 2.66e+04, 6.96e+04, 894, 269, 2.86e+03,
00667    4.94e+03, 1.98e+03, 423, 1.11e+04, 9.61e+03, 1.74e+04, 2.52e+05, 2.91e+04,
00668    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00669    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00670    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00671    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00672    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.98e+04, 5.85e+04, 3.92e+04, 2.97e+04,
00673    2.37e+05, 2.73e+05, 8.34e+04, 6.33e+04, 1.51e+05, 1.47e+05, 1.71e+05,
00674    2.96e+05, 1.56e+05, 2.93e+05, 3.33e+05, 2.11e+05, 3.37e+04, 6.55e+04,
00675    7.19e+04, 6.8e+04, 4.88e+04, 1.01e+04, 3.03e+04, 1.13e+04, 7.86e+03,
00676    2.98e+04, 1.85e+04, 8.8e+03, 8.97e+03, 2.02e+03, 1.24e+04, 1.54e+04,
00677    3.24e+03, 1.28e+04, 3.36e+04, 6.3e+03, 8.74e+03, 3.26e+03, 3.12e+03, 142,
00678    102, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00679   {133, 997, 2e+03, 1.27e+04, 2.98e+04, 6.89e+04, 906, 687, 4.41e+03,
00680    4.07e+03, 1.57e+03, 313, 5.38e+03, 6.75e+03, 1.04e+04, 2.44e+05, 3.58e+04,
00681    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00682    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00683    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00684    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00685    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.43e+04, 8.73e+04, 7.45e+04, 4.42e+04, 3.03e+05,
00686    1.55e+05, 6.69e+04, 2.8e+04, 5.35e+04, 6.83e+04, 6.18e+04, 1.53e+05,
00687    1.58e+05, 2.2e+05, 2.06e+05, 1.09e+05, 9.04e+04, 7.85e+04, 1.42e+05,
00688    1.48e+05, 7.3e+04, 2.03e+04, 3.71e+03, 1.56e+04, 8.5e+03, 7.06e+03,
00689    1.53e+04, 1.37e+04, 8.32e+03, 2.63e+03, 1.74e+04, 4.81e+03, 1.06e+04,
00690    4.76e+03, 9.88e+03, 1.18e+04, 58.5, 6.3e+03, 4e+03, 6.1e+03, 415, 816,
00691    86.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00692   {129, 896, 2.26e+03, 9.03e+03, 2.68e+04, 7.62e+04, 785, 1.76e+03, 6.61e+03,
00693    3.98e+03, 1.37e+03, 296, 1.75e+03, 5.69e+03, 9.63e+03, 2.4e+05, 7.31e+03,
00694    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00695    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00696    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00697    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00698    0, 0, 0, 0, 0, 0, 0.00111, 3.21e+03, 1.22e+05, 4.34e+04, 1.15e+05,
00699    6.85e+04, 1.4e+05, 8.91e+04, 6.97e+04, 4.55e+04, 1.88e+04, 2.85e+04,
00700    8.11e+04, 4.58e+04, 9.08e+04, 1.53e+05, 2.93e+05, 1.27e+05, 1.43e+05,
00701    1.53e+05, 1.95e+05, 8.59e+04, 3.71e+04, 9.8e+03, 8.34e+03, 2.73e+04,
00702    6.4e+03, 6.28e+03, 2.27e+04, 5.5e+04, 5.78e+03, 3.29e+03, 1.61e+04,
00703    1.11e+03, 7.16e+03, 1.33e+04, 4.66e+04, 0, 0, 5.69e+03, 4.04e+03, 6.54e+03,
00704    0, 4.31e+03, 643, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00705   {125, 798, 2.44e+03, 7.01e+03, 1.2e+04, 8.3e+04, 779, 3.08e+03, 1.03e+04,
00706    6.09e+03, 1.44e+03, 391, 309, 4.67e+03, 8.05e+03, 1.67e+05, 4.88e+03, 0, 0,
00707    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00708    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00709    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00710    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00711    0, 0, 0, 0, 96.4, 3.68e+04, 3.4e+05, 6.16e+05, 1.16e+06, 2.04e+05,
00712    1.22e+05, 4.96e+04, 6.6e+04, 6.34e+04, 5.91e+04, 5.58e+04, 8.58e+04,
00713    4.11e+04, 1.28e+04, 5.85e+04, 1.07e+05, 1.82e+05, 2.61e+05, 1.99e+05,
00714    1.96e+05, 1.38e+05, 2.09e+04, 1.12e+04, 1.39e+04, 1.29e+04, 1.16e+04,
00715    8.43e+03, 4.95e+03, 3.83e+04, 1.39e+03, 7e+03, 1.3e+04, 4.62e+03, 4.68e+03,
00716    1.15e+04, 4.76e+04, 0, 0, 4.6e+03, 6.23e+03, 1.57e+04, 0, 5.17e+03,
00717    1.06e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00718   {121, 694, 2.5e+03, 5.6e+03, 1.85e+03, 8.04e+04, 454, 2.93e+03, 7.69e+03,
00719    6.28e+03, 1.58e+03, 381, 58.2, 1.02e+05, 1.01e+04, 7.69e+04, 9.42e+03, 0,
00720    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00721    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00722    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00723    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00724    0, 1.79e+03, 0, 0, 0, 1.82e+03, 3.45e+05, 6.93e+04, 4.4e+05, 6.17e+05,
00725    2.23e+05, 9.99e+04, 1.23e+05, 6.5e+04, 8.25e+04, 2.71e+04, 6.57e+04,
00726    1.2e+05, 1.48e+05, 1.37e+04, 4.13e+04, 1.37e+05, 2.47e+05, 2.6e+05,
00727    2.79e+05, 1.85e+05, 4.37e+04, 9.27e+03, 8.35e+03, 7.75e+03, 2.05e+04,
00728    3.66e+03, 1.87e+04, 3.49e+03, 9.04e+03, 523, 1.59e+03, 2.61e+03, 5.33e+03,
00729    1.14e+04, 1.32e+04, 1.01e+04, 0, 9.74e+03, 1.03e+04, 8.12e+03, 1.56e+04,
00730    233, 2.67e+03, 2.33e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00731   {117, 597, 2.48e+03, 4.65e+03, 795, 8.2e+04, 1.15e+03, 2.18e+03, 4.18e+03,
00732    2.75e+03, 2.33e+03, 122, 579, 1.64e+04, 2.81e+04, 9.67e+04, 6e+03, 0, 0, 0,
00733    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00734    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00735    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00736    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
00737    0, 46.2, 4.67e+04, 2.17e+05, 3.51e+05, 4.47e+04, 2.79e+05, 2.82e+05,
00738    5.04e+04, 1.22e+05, 7.19e+04, 4.94e+04, 1.6e+04, 8.83e+04, 2.77e+04,
00739    2.13e+05, 1.28e+05, 2.26e+04, 4.79e+04, 1.79e+05, 3.17e+05, 2.58e+05,
00740    2.46e+05, 2.23e+05, 1.52e+04, 6.75e+03, 4.75e+03, 9e+03, 5.11e+03,
00741    5.88e+03, 2.31e+04, 2.34e+03, 551, 2.16e+03, 2.07e+03, 1.09e+03, 2.04e+03,
00742    1.08e+04, 1.18e+04, 6.28e+03, 7.14e+03, 2.2e+04, 5.4e+03, 5.43e+03,
00743    1.18e+04, 1.14e+04, 4.97e+03, 1.9e+03, 0.23, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00744    0, 0},
00745   {114, 504, 2.39e+03, 3.96e+03, 920, 6.67e+04, 2.9e+03, 1.3e+03, 4.35e+03,
00746    1.11e+03, 3.62e+03, 45.1, 1.65e+03, 1.43e+03, 1.44e+05, 8.75e+04, 431, 0,
00747    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00748    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00749    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00750    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00751    0, 0, 2.47e+04, 3.06e+05, 2.57e+05, 1.96e+05, 1.45e+05, 1.21e+05, 2.94e+05,
00752    1.21e+05, 7.77e+04, 7.83e+04, 5.69e+04, 4.64e+04, 5.61e+04, 9.59e+04,
00753    1.51e+05, 1.7e+05, 1.86e+05, 9.28e+04, 1.18e+05, 1.33e+05, 2.74e+05,
00754    2.43e+05, 1.7e+05, 3.42e+04, 8.78e+03, 3.78e+04, 2.91e+03, 7.2e+03,
00755    4.16e+04, 4.63e+04, 1.56e+04, 320, 703, 1.02e+03, 881, 1.74e+03, 2.54e+03,
00756    8.25e+03, 7.49e+03, 2.34e+03, 1.03e+04, 2.31e+04, 7.78e+03, 6.92e+03,
00757    4.44e+03, 2.28e+04, 1.51e+04, 371, 0.206, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00758    0},
00759   {111, 417, 2.29e+03, 3.35e+03, 1.1e+03, 7.32e+04, 3.32e+03, 1.01e+03,
00760    4.11e+03, 731, 3.45e+03, 147, 2.93e+03, 1.52e+03, 4.2e+04, 4.41e+04,
00761    3.61e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00762    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00763    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00764    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00765    0, 0, 0, 58.9, 1.3e+04, 2.14e+04, 8.68e+04, 3.16e+05, 2.39e+04, 2.54e+04,
00766    4.34e+04, 1.38e+05, 9.5e+04, 1.8e+05, 4.89e+04, 1.05e+05, 5.33e+04,
00767    7.34e+04, 9.1e+04, 9.09e+04, 1.18e+05, 1.31e+05, 1.4e+05, 9.22e+04,
00768    7.29e+04, 1.6e+05, 1.27e+05, 1.72e+05, 5.84e+04, 7.7e+03, 4.56e+03,
00769    2.11e+04, 4.99e+03, 5.22e+03, 5.41e+03, 4.92e+04, 3.64e+04, 1.82e+03, 214,
00770    2.83e+03, 606, 5.75e+03, 6.07e+03, 7.82e+03, 1.1e+04, 1.87e+03, 1.26e+04,
00771    1.82e+04, 8.83e+03, 9.23e+03, 2.78, 4.78e+04, 2.41e+04, 4.45, 1.17, 0, 0,
00772    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00773   {107, 341, 2.17e+03, 2.67e+03, 1.39e+03, 6.31e+04, 3.59e+03, 745, 3.49e+03,
00774    766, 1.71e+03, 671, 3.2e+03, 1.64e+03, 1.69e+04, 5.49e+04, 4.38e+04, 0, 0,
00775    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00776    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00777    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00778    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00779    1.56e+04, 3.97e+04, 1.07e+05, 2.08e+04, 7.08, 1.84e+04, 3.62e+04, 3.37e+04,
00780    1.44e+05, 1.96e+05, 1.88e+05, 5.78e+04, 1.43e+05, 1.31e+05, 7.91e+04,
00781    7.33e+04, 2.91e+04, 1.48e+05, 1.8e+05, 1.3e+05, 1.59e+05, 1.72e+05,
00782    1.31e+05, 6.44e+04, 1.31e+04, 4.12e+03, 7.54e+03, 1.48e+03, 1.61e+03,
00783    9.81e+03, 9.68e+03, 2.1e+04, 4.35e+04, 2.51e+03, 1.51e+03, 689, 1.23e+03,
00784    2.09e+03, 2.55e+03, 5.19e+03, 2.28e+03, 1.34e+04, 1.15e+03, 1.6e+04,
00785    1.45e+04, 7.19e+03, 4e+03, 200, 1.84e+04, 2.75e+04, 15, 22.8, 0, 0, 0, 0,
00786    0, 0, 0, 0, 0, 0, 0, 0},
00787   {104, 273, 2.08e+03, 1.9e+03, 1.93e+03, 5.74e+04, 4.08e+03, 1.01e+03,
00788    3.79e+03, 895, 637, 2.61e+03, 3.17e+03, 1.74e+03, 3.58e+04, 7.35e+04, 330,
00789    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00790    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00791    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00792    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00793    453, 7.38e+04, 8.47e+04, 1.99e+04, 1.32e+04, 2.32e+04, 3.66e+04, 1.85e+04,
00794    9.52e+04, 6.32e+04, 9.61e+04, 1.08e+05, 1.73e+05, 6.28e+04, 5.96e+04,
00795    3.3e+04, 7.03e+04, 2.33e+04, 1.93e+05, 1.08e+05, 1.14e+05, 9.53e+04,
00796    1.09e+05, 3.89e+04, 3.65e+03, 4.08e+03, 1.87e+03, 1.66e+03, 1.18e+03,
00797    1.88e+03, 3.3e+03, 9.66e+03, 4.06e+04, 809, 973, 221, 1.9e+03, 3.2e+03,
00798    2.05e+03, 982, 5.31e+03, 4.17e+03, 1.61e+04, 2.77e+03, 1.34e+04, 1.1e+04,
00799    7.9e+03, 6.2e+03, 2.18e+03, 1.21e+04, 1.86e+04, 0.211, 245, 0.0099, 0, 0,
00800    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00801   {103, 217, 2.02e+03, 1.24e+03, 2.83e+03, 4.14e+04, 6.66e+03, 942, 4.88e+03,
00802    511, 395, 5.86e+03, 3.69e+03, 2.27e+03, 3.38e+05, 3.95e+04, 0, 0, 0, 0, 0,
00803    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00804    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00805    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00806    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 175, 1.04e+03,
00807    5.42e+04, 9.28e+04, 2.81e+03, 1.1e+04, 4.88e+04, 5.54e+04, 4.08e+04,
00808    1.7e+04, 2.05e+05, 1.61e+05, 1.14e+05, 1.06e+05, 1.47e+05, 2.35e+05,
00809    1.24e+05, 2.11e+05, 1.49e+05, 6.24e+04, 5.1e+04, 1.05e+05, 1.45e+05,
00810    7.96e+04, 8.96e+03, 1.12e+04, 3.37e+03, 3.34e+03, 3.27e+03, 472, 1.13e+03,
00811    2.04e+03, 6.21e+03, 9.58e+03, 5.29e+04, 771, 683, 818, 3.96e+03, 6.75e+03,
00812    1.26e+03, 442, 2.97e+03, 5e+03, 1.03e+04, 2.38e+03, 9.44e+03, 6.35e+03,
00813    9.99e+03, 6.86e+03, 291, 7.08e+03, 1.14e+04, 37.8, 1.22e+03, 0.165, 0, 0,
00814    0, 0, 0, 0, 0, 0, 0, 0, 0},
00815   {101, 171, 1.98e+03, 782, 4.13e+03, 1.7e+04, 9.44e+03, 992, 5.39e+03, 284,
00816    401, 9.08e+03, 4.81e+03, 2.04e+03, 2.51e+04, 1.86e+05, 1.73e+04, 0, 0, 0,
00817    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00818    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00819    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00820    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00552, 0, 0, 0, 0, 9.09e+03,
00821    3.52e+04, 2.34e+04, 0, 1.02e+04, 2.29e+04, 5.12e+04, 9.55e+04, 1.16e+05,
00822    8.91e+04, 2.04e+05, 6.27e+04, 3.04e+04, 9.24e+04, 1.56e+05, 1.54e+05,
00823    1.07e+05, 1.95e+05, 5.86e+04, 1.12e+05, 1.19e+05, 1.02e+05, 7.74e+04,
```

```
00824    1.11e+05, 1.13e+04, 1.39e+04, 1.11e+04, 1.97e+03, 1.29e+03, 1.11e+03,
00825    1.84e+03, 2.02e+03, 8.28e+03, 6.79e+03, 2.22e+03, 1.3e+03, 522, 501,
00826    3.12e+03, 2.15e+04, 2.14e+03, 494, 2.95e+03, 3.76e+03, 1.12e+04, 2.12e+03,
00827    2.87e+03, 3.38e+03, 7.68e+03, 1.07e+03, 0, 2.26e+03, 23.1, 1.98, 445, 96.1,
00828    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00829   {99.4, 139, 1.97e+03, 538, 5.39e+03, 1.23e+04, 1e+04, 694, 5.74e+03, 250,
00830    934, 1.09e+04, 6.03e+03, 1.33e+03, 3.03e+04, 9.22e+04, 1.84e+03, 0, 0, 0,
00831    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00832    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00833    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00834    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.57e+04, 9.11e+04,
00835    0, 0, 1.02e+03, 2.61e+04, 5.56e+04, 8.43e+04, 4.53e+04, 3.34e+04, 1.35e+05,
00836    1.28e+05, 9.78e+03, 2.3e+04, 4.38e+04, 1.03e+05, 1.43e+05, 1.1e+05,
00837    2.52e+05, 1.93e+04, 1.84e+05, 1.1e+05, 1.72e+05, 1.33e+05, 8.46e+04,
00838    2.84e+04, 6.36e+03, 3.82e+04, 2.06e+03, 780, 1.08e+03, 2.14e+03, 3.03e+03,
00839    3.62e+03, 2.5e+03, 1.06e+04, 2.24e+03, 1.93e+03, 816, 1.13e+03, 1.66e+04,
00840    2.04e+03, 604, 5.01e+03, 2.44e+03, 9.44e+03, 3.17e+03, 625, 1.76e+03,
00841    7.35e+03, 6.31e+03, 0, 5.03e+03, 300, 35.1, 313, 464, 0, 0, 0, 0, 0, 0, 0,
00842    0, 0, 0, 0},
00843   {98.3, 118, 2.07e+03, 429, 5.85e+03, 1.44e+04, 8.31e+03, 580, 5.51e+03, 457,
00844    583, 1.02e+04, 7.47e+03, 2.33e+03, 4.67e+04, 2.46e+04, 0, 0, 0, 0, 0, 0, 0,
00845    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00846    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 56.6, 0, 0, 0, 0, 0, 0, 0,
00847    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00848    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 92, 0.00111, 1.55,
00849    7.43e+03, 1.34e+05, 2.38e+05, 1.61e+05, 2.23e+05, 9.63e+04, 6.5e+04,
00850    1.59e+05, 6.42e+04, 2.43e+04, 6.85e+04, 5.79e+04, 6.79e+04, 1.92e+05,
00851    8.84e+04, 5.02e+04, 1.14e+04, 3.3e+05, 2.84e+05, 1.75e+05, 1.18e+04,
00852    7.08e+04, 2.58e+04, 7.32e+03, 2.27e+04, 1.41e+03, 775, 2.88e+03, 5.2e+03,
00853    5.66e+03, 2.23e+03, 1.81e+03, 3.89e+03, 6.04e+03, 6.5e+03, 1.96e+03,
00854    2.51e+03, 5.08e+03, 1e+03, 1.43e+03, 4.41e+03, 9.97e+03, 1.37e+03,
00855    4.37e+03, 1.48e+03, 850, 5.78e+03, 7.4e+03, 0, 2.8e+03, 4.04e+03, 1.73e+03,
00856    0.189, 0.0658, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00857   {97.4, 106, 2.34e+03, 442, 5.17e+03, 1.35e+04, 9.37e+03, 838, 5.83e+03, 726,
00858    310, 9.89e+03, 8.47e+03, 3.68e+03, 6.35e+04, 1.14e+04, 0, 0, 0, 0, 0, 0, 0,
00859    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00860    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00861    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00862    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.23e+04, 2.66e+05,
00863    3.98e+05, 1.82e+05, 6.25e+04, 1.1e+05, 3.48e+04, 1.39e+04, 9.41e+04,
00864    2.79e+04, 3.5e+04, 2.22e+04, 1.02e+05, 2.52e+05, 2.33e+05, 9.67e+04,
00865    1.28e+04, 9e+03, 4.39e+04, 5.15e+04, 2.83e+05, 9.19e+03, 5.38e+03,
00866    6.14e+03, 4.36e+03, 5.5e+03, 3.44e+03, 3.13e+03, 5.71e+03, 6.44e+03, 842,
00867    444, 990, 2.85e+03, 9e+03, 3.03e+03, 501, 1.95e+03, 805, 527, 2.45e+03,
00868    3.62e+03, 1.21e+04, 2.15e+03, 5.45e+03, 1.74e+03, 1.86e+03, 1.1e+03,
00869    6.88e+03, 16.4, 3.65e+03, 5.15e+03, 70.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00870    0, 0},
00871   {98.1, 100, 2.88e+03, 551, 4.02e+03, 9.23e+03, 9.99e+03, 1.59e+03, 4.7e+03,
00872    1.45e+03, 190, 9.51e+03, 9.83e+03, 4.8e+03, 6.37e+04, 5.44e+03, 0, 0, 0, 0,
00873    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00874    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00875    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00876    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 289, 4.34, 4.51e+04,
00877    4.03e+05, 3.78e+05, 8.46e+04, 5.43e+04, 2.79e+04, 8.73e+04, 4.74e+03,
00878    1.38e+04, 3.31e+04, 9.27e+04, 5.41e+04, 8.46e+04, 1.72e+05, 1.32e+05,
00879    2.02e+05, 1.56e+05, 1.01e+05, 3.02e+04, 4.08e+04, 2.84e+05, 2.24e+05,
00880    1.51e+04, 3.59e+03, 4.65e+03, 5.43e+03, 1.78e+03, 4.89e+03, 1.09e+03,
00881    6.38e+03, 2.1e+03, 1.16e+03, 1.08e+03, 1.22e+03, 2.63e+03, 1.07e+04,
00882    1.16e+03, 643, 226, 141, 133, 522, 5.87e+03, 4.52e+03, 4.24e+03, 2.07e+03,
00883    1.69e+03, 2.42e+03, 5.28e+03, 7.37e+03, 648, 5.17e+03, 4.97e+03, 0, 0, 0,
00884    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00885   {96.6, 101, 3.79e+03, 646, 2.95e+03, 2.3e+03, 9.86e+03, 2.01e+03, 1.71e+03,
00886    622, 172, 9.27e+03, 1.16e+04, 4.87e+03, 3.75e+04, 1.04e+04, 0, 0, 0, 0, 0,
00887    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00888    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00889    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00890    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4e+03, 329, 4.14, 1.36e+05,
00891    7.26e+05, 3.02e+05, 1.13e+05, 9.75e+04, 4.1e+04, 7.64e+04, 1.72e+04,
00892    2.39e+04, 2.24e+04, 2.68e+05, 4.04e+04, 2.69e+04, 2.01e+05, 1.01e+05,
00893    1.68e+05, 1.53e+05, 1.62e+05, 1.11e+05, 2.43e+04, 6.44e+04, 1.11e+04,
00894    7.47e+03, 1.12e+04, 4.08e+03, 4.31e+03, 4.07e+03, 2.72e+03, 887, 2.06e+03,
00895    1.16e+03, 2.1e+03, 6.18e+03, 2.46e+03, 1.1e+03, 5.53e+03, 5.08e+03,
00896    1.02e+03, 799, 1.14e+03, 521, 768, 208, 262, 2.07e+03, 1.61e+03, 2.68e+03,
00897    1.89e+03, 1.17e+03, 3.31e+03, 3.87e+03, 4.16e+03, 2.42e+03, 5.5e+03, 0.595,
00898    50.7, 0.188, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00899   {97.7, 101, 4.86e+03, 593, 2.28e+03, 1.8e+03, 9.98e+03, 4.27e+03, 716, 368,
00900    403, 9.05e+03, 1.27e+04, 4.65e+03, 2.86e+04, 2.16e+04, 0, 1.44e+03, 0, 0,
00901    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00902    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00903    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00904    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.29e+04, 4.12e+05, 3.89e+05,
00905    3.07e+05, 4.98e+05, 8.35e+04, 9.88e+04, 2.96e+04, 1.44e+04, 1.77e+04,
00906    2.84e+04, 6.77e+04, 3.1e+04, 5.13e+04, 8.8e+04, 6.16e+04, 2.96e+04,
00907    1.24e+05, 2.21e+05, 2.51e+05, 1.69e+05, 1.28e+05, 8.71e+04, 7.02e+04,
00908    1.37e+04, 5.98e+03, 1.28e+04, 5.78e+03, 5.13e+03, 4.88e+03, 2.85e+03,
00909    5.41e+03, 1.26e+03, 528, 2.42e+03, 3.24e+03, 4.46e+03, 1.59e+03, 701,
00910    1.89e+03, 3.67e+03, 1.5e+03, 1.14e+03, 716, 1.56e+03, 2.85e+03, 680, 820,
```

```
00911    1.65e+03, 1.31e+03, 570, 1.3e+03, 1.31e+03, 952, 3.53e+03, 3.09e+03, 341,
00912    0.639, 79.3, 158, 252, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00913   {98.2, 101, 5.85e+03, 368, 2.14e+03, 1.96e+03, 9.07e+03, 6.87e+03, 1e+03,
00914    187, 639, 8.78e+03, 1.26e+04, 3.96e+03, 3.57e+04, 2.42e+04, 0.00111, 285,
00915    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00916    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00917    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00918    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 397, 1.77e+05, 2.15e+05,
00919    1.44e+05, 1.99e+05, 1.78e+05, 4.27e+04, 8.34e+04, 8.11e+04, 4.23e+04,
00920    1.77e+04, 8.42e+04, 5.01e+04, 9.26e+04, 1.2e+04, 1.13e+04, 6.13e+03,
00921    1.03e+04, 4e+04, 5.3e+04, 1.85e+04, 2.34e+04, 1.42e+04, 9.1e+03, 8.75e+03,
00922    2.25e+04, 7.12e+04, 1.82e+04, 3.87e+03, 6.75e+03, 2.5e+03, 2.41e+03,
00923    1.14e+03, 2.35e+03, 1.55e+03, 4.41e+03, 4.55e+03, 6.12e+03, 1.87e+03,
00924    2.39e+03, 1.46e+03, 2.85e+03, 2.72e+03, 549, 1.31e+03, 838, 3.78e+03,
00925    1.38e+03, 511, 837, 1.51e+03, 873, 503, 492, 0.0719, 0, 0, 181, 2.07e+03,
00926    84.8, 2.66e+03, 2.68e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00927   {99.6, 97.5, 6.66e+03, 371, 2.48e+03, 3.38e+03, 1.08e+04, 3.79e+03, 1e+03,
00928    73.1, 495, 8.49e+03, 1.2e+04, 3.6e+03, 5e+04, 2.14e+04, 5.12e+03, 1.49e+04,
00929    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00930    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00931    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00932    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.53e+05, 2.72e+05, 9.81e+04,
00933    6.45e+04, 8.31e+04, 2.41e+04, 2.8e+04, 6.03e+04, 3.25e+04, 7.64e+04,
00934    5.96e+04, 1.12e+05, 5.51e+04, 2.46e+04, 7.24e+03, 5.33e+03, 3.82e+03,
00935    4.93e+03, 1.73e+04, 2.04e+04, 8.36e+03, 1.06e+04, 9.46e+03, 5.99e+03,
00936    4.58e+03, 1.55e+04, 2.78e+04, 2.36e+03, 2.15e+03, 3.1e+03, 3.03e+03,
00937    3.61e+03, 2.34e+03, 2.5e+03, 2.32e+03, 8.62e+03, 940, 539, 1.91e+03, 404,
00938    2.15e+03, 2.72e+03, 2.58e+03, 230, 251, 521, 5.59e+03, 1e+03, 315,
00939    1.18e+03, 615, 458, 352, 36.2, 0, 4.7, 0, 0, 3.28e+03, 0.00772, 823,
00940    6.14e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00941   {102, 88.8, 7.3e+03, 787, 2.4e+03, 5.4e+03, 9.98e+03, 1.93e+03, 346, 270,
00942    355, 8.63e+03, 1.14e+04, 4.28e+03, 7.04e+04, 1.04e+04, 1.08e+04, 1.21e+04,
00943    2.39e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00944    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00945    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00946    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.29e+05, 5.06e+05,
00947    1.09e+05, 3.82e+04, 2.57e+04, 8.06e+03, 2.54e+04, 9.36e+04, 6.2e+04,
00948    1.43e+05, 1.43e+05, 1.11e+05, 2.24e+04, 6.12e+03, 1.81e+03, 4.58e+03,
00949    2.94e+03, 4.79e+03, 6.31e+03, 7.58e+03, 5.62e+03, 7.49e+03, 5.43e+03,
00950    3.24e+03, 1.55e+03, 6.17e+03, 4.15e+03, 3.76e+03, 2.29e+03, 2.53e+03,
00951    1.93e+03, 2.76e+03, 3.32e+03, 912, 5.11e+03, 2.66e+04, 3.6e+03, 452,
00952    2.06e+03, 303, 2.32e+03, 2.02e+03, 2.73e+03, 371, 1.09e+03, 640, 4.83e+03,
00953    988, 99.3, 1.31e+03, 263, 144, 167, 0, 41.1, 553, 0.00552, 0, 5.56e+03,
00954    46.1, 2.14e+03, 6.67e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00955   {104, 74.9, 7.69e+03, 1.53e+03, 5.1e+03, 6.08e+03, 8.46e+03, 1.2e+03, 471,
00956    419, 517, 9.04e+03, 1.19e+04, 6.39e+03, 7.3e+04, 5.16e+03, 1.96e+04,
00957    1.34e+04, 5.43e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00958    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00959    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00960    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.89e+04,
00961    3.79e+05, 2.87e+05, 8.08e+04, 4.65e+04, 5.2e+04, 8.01e+04, 5.25e+04,
00962    8.27e+04, 1.53e+05, 9.2e+04, 1e+04, 1.59e+04, 1.13e+04, 3e+03, 4.65e+03,
00963    4.11e+03, 2.03e+04, 2.08e+04, 4.21e+03, 2.86e+03, 2.3e+03, 4.41e+03,
00964    4.79e+03, 2.72e+03, 4.05e+03, 5.56e+03, 4.43e+03, 4.63e+03, 3.16e+03,
00965    2.23e+03, 4.36e+03, 965, 1.73e+03, 4.82e+03, 2.36e+04, 1.67e+04, 293, 83.7,
00966    703, 694, 1.16e+03, 2.59e+03, 704, 550, 269, 865, 2.09e+03, 763, 510, 922,
00967    325, 14.7, 62.8, 2.63, 18.3, 243, 200, 0.917, 5.25e+03, 1.61e+03, 4.45e+03,
00968    6.07e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00969   {108, 57.2, 7.83e+03, 2.95e+03, 2.93e+03, 7.49e+03, 7.21e+03, 723, 1.3e+03,
00970    1.05e+03, 1.24e+03, 8.75e+03, 1.18e+04, 9.18e+03, 5.92e+04, 1.27e+04,
00971    2.89e+04, 1.74e+04, 4.62e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00972    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00973    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00974    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00975    3.08e+05, 3.43e+05, 5.13e+05, 8.54e+04, 1.41e+05, 9.24e+04, 1.17e+05,
00976    2.44e+05, 7.01e+05, 1.01e+05, 2.74e+04, 1.37e+03, 4.87e+03, 6.93e+03,
00977    4.8e+03, 5.14e+03, 2.5e+03, 3.4e+03, 7.74e+03, 4.18e+03, 6.08e+03,
00978    3.22e+03, 3.58e+03, 2.75e+03, 3.15e+03, 2.08e+03, 4.62e+03, 2.71e+03,
00979    1.21e+03, 1.93e+03, 952, 1.16e+03, 5.37e+03, 6.3e+03, 2.59e+04, 609, 71.4,
00980    121, 354, 279, 427, 1.14e+03, 636, 250, 213, 555, 256, 1.68e+03, 1.64e+03,
00981    465, 1.75e+03, 311, 35.2, 0.123, 3.46, 499, 2.4e+03, 4.03e+03, 1.71e+03,
00982    5.01e+03, 3.73e+03, 2.87e+03, 606, 16.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00983   {111, 39.1, 7.8e+03, 5.58e+03, 2.59e+03, 1.29e+04, 6.97e+03, 299, 1.87e+03,
00984    1.22e+03, 3.51e+03, 6.87e+03, 1.03e+04, 1.19e+04, 3.96e+04, 2.21e+04,
00985    2.76e+04, 2.22e+04, 5.96e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00986    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00987    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00988    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 56.8,
00989    4.88e+05, 1.23e+05, 6.56e+05, 8.02e+04, 4.25e+05, 1.15e+05, 1.96e+05,
00990    1.23e+05, 4.03e+04, 1.9e+03, 1.91e+03, 1.67e+03, 3.1e+03, 1.09e+04,
00991    5.03e+03, 2.43e+03, 2.79e+03, 1.14e+03, 3.21e+03, 3.79e+03, 5.11e+03,
00992    1.88e+03, 2.41e+03, 3.2e+03, 2.13e+03, 2.29e+03, 5.56e+03, 2.66e+03,
00993    3.43e+03, 5.96e+03, 3.18e+03, 286, 461, 3.06e+03, 414, 208, 454, 457, 517,
00994    643, 444, 431, 209, 101, 344, 2.2e+03, 3.24e+03, 1.06e+03, 1.35e+03, 93.4,
00995    1.09e+03, 198, 121, 1.31, 1.82, 1.18e+03, 3.08e+03, 4.07e+03, 6.41e+03,
00996    1.77e+03, 762, 135, 0, 2.94e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00997   {115, 26.3, 7.67e+03, 8.15e+03, 2.34e+03, 1.61e+04, 7.97e+03, 168, 2.21e+03,
```

```
00998    3.79e+03, 7.98e+03, 4.94e+03, 8.2e+03, 1.3e+04, 2.54e+04, 1.07e+04,
00999    3.16e+04, 2.25e+04, 5.95e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01000    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01001    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01002    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.39e+04,
01003    6.35e+05, 1.51e+05, 1.59e+05, 7.73e+05, 1.04e+05, 664, 5.07e+04, 9.23e+03,
01004    2.99e+03, 898, 878, 1.7e+03, 1.2e+03, 9.19e+03, 5.46e+03, 5.3e+03,
01005    1.96e+03, 683, 1.71e+03, 2.11e+03, 2.34e+03, 1.37e+03, 2.22e+03, 1.76e+03,
01006    1.24e+03, 1.69e+03, 3.3e+03, 1.4e+03, 4.93e+03, 2.01e+03, 1.3e+03,
01007    4.09e+03, 9.49e+03, 443, 251, 408, 0.317, 46.5, 189, 1.3e+03, 499, 286,
01008    516, 1.29e+03, 1.1e+03, 1.81e+03, 1.3e+03, 661, 1.06e+03, 363, 1.31e+03,
01009    234, 650, 434, 0.0663, 2.81e+03, 2.79e+03, 5.13e+03, 3.32e+03, 4.66e+03,
01010    545, 697, 2.98e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
01011   {119, 18, 7.53e+03, 1.08e+04, 1.49e+03, 1.22e+04, 1.12e+04, 105, 2.32e+03,
01012    9.74e+03, 1.28e+04, 3.4e+03, 7.07e+03, 1.36e+04, 1.98e+04, 1.07e+04,
01013    4.25e+04, 1.86e+04, 4.53e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01014    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01015    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01016    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.48e+05,
01017    1.81e+05, 5.48e+05, 1.03e+06, 7.77e+05, 1.02e+03, 0.00332, 0, 0, 0.0476,
01018    2.45, 1.14, 39.5, 795, 757, 1.19e+03, 1.27e+03, 1.43e+03, 1.73e+03,
01019    1.25e+03, 905, 1.73e+03, 719, 1.07e+03, 881, 1.8e+03, 2.1e+03, 1.85e+03,
01020    1.32e+03, 6.53e+03, 7.7e+03, 403, 458, 773, 253, 242, 168, 254, 207, 187,
01021    280, 696, 715, 1.85e+03, 1.35e+03, 388, 258, 549, 1.61e+03, 1.32e+03, 945,
01022    1.38e+03, 289, 469, 75.1, 101, 3.52e+03, 6.25e+03, 2.03e+03, 155, 39.4,
01023    1.48e+03, 1.35e+03, 2.26e+03, 892, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
01024   {125, 15.2, 7.47e+03, 1.41e+04, 1.2e+03, 7.58e+03, 1.29e+04, 63.8, 2.16e+03,
01025    6.77e+03, 1.25e+04, 2.89e+03, 6.34e+03, 1.33e+04, 1.76e+04, 6.66e+03,
01026    4.61e+04, 1.12e+04, 1.27e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01027    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01028    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01029    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.71e+05,
01030    4.42e+05, 1.59e+05, 707, 531, 0, 0, 0, 0, 0, 0, 0, 0.0144, 145, 579, 630,
01031    710, 566, 905, 1.55e+03, 1.04e+03, 1.62e+03, 1.61e+03, 1.87e+03, 1.89e+03,
01032    1.05e+03, 1.59e+03, 736, 2.58e+03, 2.5e+03, 1.74e+03, 2.93e+03, 864, 419,
01033    781, 890, 382, 205, 713, 269, 529, 315, 975, 993, 1.12e+03, 965, 1.24e+03,
01034    613, 2.24e+03, 588, 889, 1.12e+03, 115, 336, 4.09e+03, 1.25e+04, 902,
01035    2.96e+03, 218, 2.73e+03, 1.02e+03, 1.2e+04, 1.93e+03, 22.7, 1.15e+04,
01036    1.47e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
01037   {129, 19.6, 7.5e+03, 1.78e+04, 5.55e+03, 3.28e+03, 1.3e+04, 84.6, 1.71e+03,
01038    3.53e+03, 7.56e+03, 3.35e+03, 5.76e+03, 1.23e+04, 1.72e+04, 2.38e+03,
01039    2.17e+04, 2.81e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01040    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01041    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01042    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.02e+04,
01043    7.04e+04, 3.3e+04, 2.04e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.05, 317, 565,
01044    771, 502, 4.99e+03, 1.14e+04, 4.65e+03, 805, 722, 791, 674, 884, 909, 746,
01045    2.7e+03, 605, 647, 975, 1.1e+03, 405, 406, 634, 422, 167, 1.04e+03, 151,
01046    588, 648, 2.08e+03, 516, 492, 724, 178, 864, 2.45e+03, 790, 3.12e+03,
01047    1.18e+04, 3.16e+03, 23.3, 1.27e+04, 1.28e+04, 1.15e+04, 1.68e+04, 1.07e+04,
01048    4.05e+03, 3.93e+03, 1.68e+04, 4.38e+03, 3.13e+03, 3.92e+04, 1.63e+04, 0, 0,
01049    0, 0, 0, 0, 0, 0, 0, 0},
01050   {135, 30.4, 7.54e+03, 2.06e+04, 1.34e+04, 1.54e+03, 9.69e+03, 191, 1.44e+03,
01051    2.46e+03, 3.46e+03, 4.9e+03, 5.3e+03, 1.08e+04, 1.68e+04, 1.22e+03,
01052    2.96e+04, 1.06e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01053    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01054    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01055    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.39e+05,
01056    1.51e+05, 1.9e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 44.3, 745, 523, 327,
01057    8.64e+03, 1.31e+04, 1.4e+04, 2.23e+03, 803, 733, 833, 1.26e+03, 799, 244,
01058    871, 541, 1.01e+03, 398, 719, 438, 1.12e+03, 608, 533, 239, 957, 158,
01059    1e+03, 1.26e+03, 660, 33.4, 0, 4.33, 307, 1.19e+03, 824, 2.28e+03, 656,
01060    3.94e+03, 4.74e+03, 200, 1.89e+04, 1.81e+04, 7.7e+03, 1.13e+04, 1.47e+04,
01061    5.96e+03, 5.14e+03, 1.12e+04, 1.43e+03, 1e+04, 7.84e+04, 3.37e+04, 0, 0, 0,
01062    0, 0, 0, 0, 0, 0, 0},
01063   {142, 47.6, 7.5e+03, 2.3e+04, 1.81e+04, 6.9e+03, 3.29e+03, 686, 1.9e+03,
01064    1.12e+03, 1.64e+03, 7.07e+03, 4.84e+03, 8.59e+03, 1.52e+04, 1.43e+03,
01065    8.54e+03, 7.63e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01066    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01067    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01068    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.83e+04,
01069    2.24e+05, 3.4e+05, 4.83e+03, 0.18, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 728,
01070    364, 417, 1.82e+03, 5.67e+03, 1.5e+04, 4.34e+03, 2.46e+03, 890, 1.02e+03,
01071    988, 832, 1.02e+03, 922, 812, 1.77e+03, 606, 1.03e+03, 1.07e+03, 611, 528,
01072    664, 188, 1.22e+03, 316, 2.03e+03, 263, 131, 1.29, 0, 0, 30.5, 732,
01073    1.52e+03, 3.03e+03, 772, 4.62e+03, 8.66e+03, 2.29e+03, 1.27e+04, 1.64e+04,
01074    0.0341, 1.65e+04, 1.47e+04, 196, 4.24e+03, 2.7e+03, 4.57e+03, 2.71e+04,
01075    1.08e+05, 8.74e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01076   {149, 70.2, 7.43e+03, 2.48e+04, 1.85e+04, 1.94e+04, 5.5e+03, 899, 2.17e+03,
01077    1.96e+03, 2.01e+03, 8.59e+03, 5.88e+03, 6.21e+03, 1.45e+04, 3.75e+03,
01078    2.47e+04, 3.16e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01079    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01080    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01081    0, 0, 0, 0, 1.03e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 558, 1.13e+06,
01082    4.55e+05, 3.73e+05, 5.7e+03, 0.241, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13.4,
01083    129, 272, 159, 226, 2.27e+03, 5.32e+03, 3.65e+03, 3e+03, 820, 862, 396,
01084    669, 1.05e+03, 3.34e+03, 1.72e+03, 1.4e+03, 1.33e+04, 2.18e+03, 1.13e+03,
```

```
01085    892, 534, 884, 262, 1.25e+03, 648, 822, 405, 0, 0, 0, 0, 0.00552, 2.49e+03,
01086    2.15e+03, 3.91e+03, 2.51e+03, 2.97e+03, 3.49e+03, 3.05e+03, 5.43e+03, 281,
01087    0, 1.54e+04, 2.16e+04, 6.11e+03, 1.29e+04, 4.9e+03, 1.01e+04, 1.01e+05,
01088    1.95e+05, 5.52e+04, 4.5e+03, 0, 0, 0, 0, 0, 0, 0, 0},
01089   {155, 95.7, 7.25e+03, 2.61e+04, 1.62e+04, 3.49e+04, 7.24e+03, 1.06e+03,
01090    1.15e+03, 3.66e+03, 3.08e+03, 8.5e+03, 7.74e+03, 4.93e+03, 1.27e+04,
01091    7.44e+03, 4.71e+04, 2.07e+04, 0, 0, 0, 9.91e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01092    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01093    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01094    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.96e+04, 3.87e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01095    0, 0, 0, 0, 1.3e+05, 3.09e+05, 5.31e+05, 7.25e+04, 841, 1.47e+03, 201, 0, 0,
01096    0, 0, 0, 0, 0, 0, 0.377, 4.02, 1.48e+03, 1.47e+03, 103, 38.4, 123,
01097    1.25e+03, 4.42e+03, 4.53e+03, 1.41e+03, 798, 560, 1.15e+03, 2.11e+03,
01098    2.87e+03, 1.58e+03, 2.44e+03, 6.87e+03, 1.03e+04, 672, 782, 345, 600, 418,
01099    827, 340, 710, 529, 0, 0, 0, 0, 0, 2.23e+03, 2.83e+03, 5.21e+03, 1.01e+04,
01100    2.43e+03, 5.23e+03, 6.22e+03, 451, 0, 0, 4.87e+03, 1.83e+04, 1.3e+04,
01101    1.06e+04, 2.98e+03, 3.66e+04, 1.52e+05, 2.32e+05, 9.78e+03, 1.38e+04, 0, 0,
01102    0, 0, 0, 0, 0, 0},
01103   {163, 122, 6.92e+03, 2.74e+04, 5.19e+03, 1.22e+04, 1.44e+04, 366, 966,
01104    2.72e+03, 4.1e+03, 5.11e+03, 8.37e+03, 6.97e+03, 8.7e+03, 9.35e+03, 4e+04,
01105    2.44e+04, 0, 0, 0, 7.46, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01106    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01107    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01108    1.13e+04, 30.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.43e+05, 3.42e+05,
01109    1.02e+05, 6.65e+03, 4.14e+03, 4.79e+03, 2.28e+03, 115, 0, 0, 0, 0, 0, 0, 0,
01110    0.287, 5.1, 1.29e+03, 543, 992, 957, 661, 195, 1.91e+03, 2.82e+03, 558,
01111    411, 775, 1.22e+03, 1.48e+03, 860, 2.24e+03, 493, 7.92e+03, 2.24e+04, 988,
01112    1.08e+03, 422, 578, 538, 702, 344, 1.01e+03, 280, 0, 0, 0, 0, 0, 0,
01113    1.61e+03, 7.29e+03, 2.08e+04, 3.28e+03, 7.73e+03, 1.44e+03, 5.66, 204,
01114    4.7e+03, 848, 1.08e+03, 1.08e+04, 1.48e+04, 1.22e+04, 1.35e+04, 1.36e+05,
01115    1.61e+05, 6.54e+03, 1.04e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0},
01116   {170, 141, 6.56e+03, 2.84e+04, 813, 1.13e+04, 1.88e+04, 287, 2.22e+03, 837,
01117    3.76e+03, 2.76e+03, 2.16e+04, 1.27e+04, 2.73e+03, 6.42e+03, 3.44e+04,
01118    3.12e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01119    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01120    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01121    249, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.15e+03, 2.18e+05, 2.12e+05,
01122    5.14e+04, 5.27e+04, 5.15e+03, 4.23e+03, 1.57e+03, 99, 0.0299, 0, 0, 0, 0,
01123    0, 0, 0.0711, 10.3, 623, 754, 1.16e+03, 673, 536, 513, 633, 565, 444, 394,
01124    586, 687, 500, 1.02e+03, 7.08e+03, 927, 8.11e+03, 1.36e+03, 9.08e+03,
01125    2.74e+03, 786, 1.1e+03, 1.5e+03, 404, 906, 1.05e+03, 75.9, 0, 0, 0, 0, 0,
01126    0, 3.76e+03, 7.18e+03, 1.88e+04, 2.41e+03, 1.17e+04, 2.7e+03, 2.48,
01127    2.43e+03, 1.71e+03, 9.87e+03, 0, 1.71e+04, 5.83e+03, 1.61e+04, 1.61e+04,
01128    1.66e+05, 1.04e+05, 3.69e+04, 2.14e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01129   {178, 151, 6.29e+03, 2.9e+04, 2.79e+03, 1.72e+04, 1.07e+04, 934, 1.61e+03,
01130    1.56e+03, 3.61e+03, 4.12e+04, 1.13e+05, 1.56e+04, 455, 1.73e+03, 5.46e+04,
01131    2.82e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01132    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01133    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01134    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.64e+04, 3e+05, 2.12e+05, 5.88e+04,
01135    2.04e+03, 29.6, 85.1, 121, 81.5, 1.13, 0, 0, 0, 0, 0, 0, 0, 29.3, 752, 867,
01136    670, 3.54e+03, 2.45e+03, 1.68e+03, 669, 287, 539, 175, 351, 919, 2.27e+03,
01137    2.1e+03, 1.88e+03, 5.88e+03, 1.86e+04, 7.49e-10, 1.3e+04, 2.3e+03, 637,
01138    674, 245, 235, 865, 482, 0.104, 0, 0, 0, 0, 0, 0, 452, 4.2e+03, 5.54e+03,
01139    8.16e+03, 7.23e+03, 1.31e+03, 258, 3.22e+03, 1.23e+03, 4.71e+03, 1.14e+04,
01140    1.04e+04, 5.51e+03, 4.03e+04, 3.09e+03, 6.69e+04, 5.28e+04, 4.28e+04,
01141    5.43e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01142   {186, 153, 5.99e+03, 2.89e+04, 1.16e+04, 1.81e+04, 3.12e+03, 1.46e+03, 882,
01143    4.23e+03, 1.01e+04, 5.3e+04, 1.45e+05, 2.92e+04, 2.82e+03, 601, 7.27e+04,
01144    1.36e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01145    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01146    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01147    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.52e+03, 1.05e+05, 1.37e+05, 1.08e+05,
01148    1.39e+05, 0, 0, 0.00111, 4.48, 30, 0.954, 0, 0, 0, 0, 0, 0, 0, 8.38, 796,
01149    1.04e+03, 2.14e+03, 3.74e+03, 2.15e+03, 1.5e+03, 1.1e+03, 879, 1.05e+03,
01150    1.21e+03, 157, 407, 2.46e-10, 14.9, 114, 3.05e+03, 260, 7.49e-10, 1.06e+04,
01151    1.04e+03, 1.18e+03, 409, 294, 272, 549, 311, 0, 0, 0, 0, 0, 0, 0, 39.9,
01152    144, 4.07e+03, 3.2e+03, 246, 0, 1.19e+03, 3.01e+03, 1.61e+03, 9.96e+03,
01153    1.8e+04, 1.87e+04, 4.11e+03, 4.85e+04, 2.54e+04, 6.68e+04, 1.62e+04,
01154    4.46e+04, 7.25e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01155   {195, 152, 5.63e+03, 2.84e+04, 1.81e+04, 1.84e+04, 1.53e+03, 1.35e+03, 771,
01156    3.07e+03, 3.04e+04, 3.23e+05, 4.06e+05, 2.7e+04, 4.99e+03, 2.54e+03,
01157    6.58e+04, 1.66e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01158    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01159    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01160    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.57e+03, 9.61e+03, 2.83e+04,
01161    9.51e+04, 6.81e+04, 9.87e+04, 5.77e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01162    0, 35.7, 827, 858, 5.18e+03, 5.66e+03, 1.59e+04, 7.75e+03, 1.58e+03,
01163    1.39e+03, 971, 523, 323, 781, 891, 3.75e+03, 1.99e+03, 899, 1.02e+03,
01164    1.42e+04, 2.98e+03, 1.94e+03, 561, 65.1, 194, 664, 724, 23.9, 0, 0, 0, 0,
01165    0, 0, 37.1, 611, 703, 734, 1.03e+04, 2.14e+04, 2.74e+03, 2.26e+04,
01166    6.81e+03, 4.68e+03, 1.69e+04, 2.1e+04, 3.69e+04, 8.96e+03, 1.47e+05,
01167    3.23e+04, 3.2e+04, 5.02e+04, 8.56e+04, 8.77e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01168   {204, 148, 5.34e+03, 2.74e+04, 1.82e+04, 1.21e+04, 1.65e+03, 1.46e+03,
01169    1.06e+03, 2.36e+03, 3.29e+04, 3.94e+05, 3.43e+05, 7.94e+03, 1.67e+03,
01170    8.39e+03, 8.34e+04, 2.13e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01171    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
01172    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01173    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.2e+05, 5.98e+04,
01174    1.98e+04, 2.27e+04, 3.18e+04, 4.69e+04, 3.01e+04, 0, 0, 0, 0, 0, 0.402, 0,
01175    0, 0, 0, 0, 0, 0, 178, 390, 1.04e+03, 2.46e+03, 3.11e+03, 2.18e+04,
01176    1.51e+04, 4.55e+03, 1.35e+03, 1.83e+03, 724, 333, 529, 546, 3.29e+03,
01177    6.65e+03, 1.37e+03, 4.57e+03, 1.26e+04, 2.05e+03, 4.9e+03, 285, 87.5, 435,
01178    820, 2.12e+03, 0, 0, 0, 0, 0, 0, 0, 288, 1.58e+03, 9.97e+03, 4.05e+03,
01179    1.13e+04, 2.46e+03, 1.82e+04, 2.06e+04, 8.79e+03, 9.55e+03, 5.21e+04,
01180    8.2e+03, 4.34e+04, 1.73e+04, 1.22e+05, 5.41e+04, 6.09e+04, 4.65e+04,
01181    9.74e+04, 1.37e+05, 0.0099, 0, 0, 0, 0, 0, 0, 0},
01182   {213, 147, 5.15e+03, 2.56e+04, 1.51e+04, 3.43e+03, 2.04e+03, 744, 1.94e+03,
01183    5.96e+03, 3.5e+04, 1.75e+05, 4.81e+04, 9.09e+03, 431, 1.03e+04, 1.13e+05,
01184    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01185    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01186    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01187    0, 0, 0, 0, 0, 0, 0, 1.6e+05, 6.49e+05, 7.14e+03, 4.56e+03, 1.3e+03,
01188    2.73e+03, 1.94e+03, 44.7, 0, 0, 0, 0, 0, 2.25e+03, 3.15e+03, 0, 0, 0, 0, 0,
01189    0, 205, 474, 750, 2.38e+03, 3.27e+03, 5.52e+04, 5.22e+04, 9.3e+03,
01190    2.16e+03, 1.82e+03, 1.35e+03, 1.39e+03, 1.71e+03, 1.47e+03, 3.05e+03,
01191    3.83e+03, 623, 8.63e+03, 2.51e+03, 1.88e+03, 2.8e+03, 403, 309, 643, 893,
01192    1.7e+03, 0, 0, 0, 0, 0, 0, 12.8, 14.6, 9.82e+03, 2.08e+04, 3.13e+03,
01193    1.24e+04, 3.07e+03, 4.63e+03, 1.61e+04, 2.43e+03, 1.2e+04, 3e+04, 4.47e+03,
01194    2.19e+05, 1e+05, 7.75e+04, 3.96e+04, 4.87e+04, 1.39e+04, 1.32e+05,
01195    1.58e+05, 0, 0, 0, 0, 0, 0, 0, 0},
01196   {220, 144, 5e+03, 2.29e+04, 1.22e+04, 902, 2.72e+03, 646, 3.96e+03,
01197    9.35e+03, 7.26e+04, 7.24e+04, 3.64e+04, 1.84e+04, 1.18e+03, 6.44e+03,
01198    1.49e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01199    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01200    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01201    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.69e+03, 6.45e+05, 2.29e+05, 0, 0, 0, 37.5,
01202    11.7, 0, 0, 0, 0, 0, 0.00111, 120, 6.37e+03, 0, 0, 0, 0.651, 17.9, 80, 180,
01203    207, 381, 893, 1.42e+03, 1.24e+05, 9.82e+04, 2.58e+04, 2.69e+03, 1.99e+03,
01204    1.55e+03, 3.71e+03, 123, 1.52e+03, 643, 171, 2.97e+03, 2.55e+03, 1.65e+03,
01205    1.93e+03, 931, 673, 190, 458, 557, 1.21e+03, 0, 0, 0, 0, 0, 0, 106, 108,
01206    7.19e+03, 2.53e+04, 338, 8.01e+03, 4.79e+03, 6.72e+03, 9.65e+03, 3.69e+03,
01207    1.71e+04, 2.58e+04, 2.88e+04, 3.97e+05, 2.08e+05, 4.46e+04, 2.99e+04,
01208    7.65e+04, 7.15e+04, 2.21e+05, 2.36e+05, 62.5, 0, 0, 0, 0, 0, 0, 0},
01209   {233, 144, 4.86e+03, 2.01e+04, 1.11e+04, 857, 2.74e+03, 1.15e+03, 3.95e+03,
01210    1.03e+04, 1.05e+05, 3.78e+04, 2.53e+04, 2.94e+04, 1.35e+03, 4.64e+03,
01211    1.83e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01212    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01213    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01214    0, 0, 0, 0, 0, 0, 0, 0, 96.3, 2.58e+04, 2.59e+05, 0, 0, 0, 0, 0, 0, 0, 0,
01215    0, 0, 0, 0, 0.321, 2.53e+03, 0, 0.00111, 2.8, 78, 147, 110, 286, 182, 290,
01216    789, 974, 1.69e+04, 9.81e+04, 3.62e+04, 1.74e+04, 1.05e+03, 1.02e+03,
01217    2.62e+03, 260, 570, 1.2e+03, 18.3, 3.23e+03, 1.53e+03, 1e+03, 1.62e+03,
01218    1.12e+03, 456, 534, 21, 13.1, 10.3, 0, 0, 0, 0, 0, 0, 0, 428, 3.3e+03,
01219    6.38e+03, 0, 6.71e+03, 9.19e+03, 537, 5.42e+03, 3.76e+03, 1.82e+04,
01220    2.33e+04, 2.9e+04, 2.09e+05, 2.3e+05, 5.37e+04, 3.49e+04, 4.16e+04,
01221    4.88e+04, 2.67e+05, 1.96e+05, 1.54e+04, 0, 0, 0, 0, 0, 0, 0},
01222   {243, 142, 4.72e+03, 1.79e+04, 1.18e+04, 1.02e+03, 2.25e+03, 2.32e+03,
01223    4.05e+03, 7.85e+03, 7.99e+04, 3.67e+04, 8.05e+04, 1.75e+04, 5.85e+03,
01224    5.85e+03, 1.72e+05, 4.46e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01225    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01226    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01227    0, 2.46e+03, 1.58e+04, 6.12e+04, 27, 2e+04, 1.3e+04, 0, 0, 0, 0, 0, 0, 0,
01228    2.32e+04, 2.9e+04, 1.41e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0577, 0, 0, 118,
01229    1.09e+03, 0, 0.739, 3.05, 49, 72.4, 79.2, 21.3, 4.17, 48.9, 498, 1.26e+03,
01230    1.75e+03, 2.75e+04, 2.45e+04, 3.56e+04, 6.15e+03, 2.36e+03, 1.04e+03,
01231    4.08e+03, 3.51e+03, 4.4e+03, 1.72e+03, 1.53e+03, 4.65e+03, 1.45e+03, 867,
01232    1.54e+03, 1.85e+03, 217, 19.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0155, 1.51e+03,
01233    0, 0, 334, 161, 15.5, 2.48e+03, 5.72e+03, 1.83e+04, 6.62e+04, 2.67e+04,
01234    1.99e+05, 8.97e+04, 9.62e+04, 7.55e+04, 1.44e+05, 8.68e+04, 2.64e+05,
01235    2.5e+05, 2.29e+04, 0, 0, 0, 0, 0, 0, 0},
01236   {251, 145, 4.55e+03, 1.63e+04, 1.54e+04, 1.8e+03, 2.05e+03, 2.93e+03,
01237    5.43e+03, 3.58e+03, 6.73e+04, 4.13e+04, 7.11e+04, 1.25e+04, 1.46e+04,
01238    7.81e+03, 3.55e+04, 2.29e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01239    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01240    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01241    10.9, 1.04e+04, 3.46e+05, 9.66e+05, 4.33e+05, 5.29e+05, 7.72e+03, 5.76e+03,
01242    2.17e+04, 1.1e+04, 0, 0, 0, 0, 0, 266, 5.99e+03, 2.14e+04, 2.18e+03, 0, 0,
01243    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.53e+04, 958, 0, 0.0521, 2.84, 5.61, 0.94, 0,
01244    0, 0, 0, 83.9, 522, 2.42e+03, 2.04e+03, 4.05e+04, 5.68e+04, 6.98e+04,
01245    1.26e+04, 2.73e+03, 1.53e+04, 2.81e+03, 1.38e+04, 2.31e+03, 2.31e+03,
01246    2.12e+03, 2.47e+03, 500, 2.07e+03, 3.95e+03, 0.0099, 0.802, 0, 0.00881,
01247    159, 0.00111, 0.00111, 0, 124, 0, 368, 0, 2.44, 0, 0, 0, 0, 0.494,
01248    1.57e+03, 5.91e+03, 4.62e+04, 1.26e+05, 1.19e+05, 6.21e+04, 8.98e+04,
01249    8.88e+04, 9.87e+04, 1.53e+05, 1.49e+05, 2.08e+05, 2.94e+05, 6.48e+04, 0, 0,
01250    0, 0, 0, 0, 0, 0},
01251   {260, 146, 4.39e+03, 1.57e+04, 2.15e+04, 2.28e+03, 2.29e+03, 3.06e+03,
01252    9.53e+03, 8.39e+03, 6.06e+04, 2.93e+04, 5.34e+04, 2.95e+04, 3.7e+04,
01253    1.42e+04, 8.36e+04, 2.39e+04, 1.61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01254    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01255    36.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01256    0, 2.36e+04, 1.14e+06, 1.33e+06, 5.95e+05, 5.42e+05, 4.67e+05, 5.88e+05,
01257    2.08e+06, 1.92e+06, 9.94e+05, 1.53e+05, 262, 0.00111, 0, 0, 0, 0, 20.3,
01258    4.05e+04, 1.3e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.383, 3.74e+03, 0.131, 0,
```

```
01259    0, 0, 1.35, 0, 0, 0, 0, 0, 1.63, 124, 1.02e+03, 1.95e+03, 2.5e+03,
01260    4.95e+04, 5.4e+04, 2.97e+04, 6.57e+03, 8.01e+03, 1.01e+04, 8.51e+03,
01261    4.83e+03, 6.79e+03, 2.01e+03, 829, 713, 2.19e+03, 2.94e+03, 125, 405, 793,
01262    784, 71.9, 0.042, 0, 0.00442, 0, 38.5, 2.4, 0, 0, 0, 0, 0, 4.11, 62.8,
01263    1.73e+03, 3.11e+04, 6.43e+04, 2.92e+05, 3.6e+04, 0.00775, 1.36e+04,
01264    6.15e+04, 1.32e+05, 1.9e+05, 1.95e+05, 2.19e+05, 1.57e+05, 8.52e+04, 0, 0,
01265    0, 0, 0, 0, 0},
01266   {271, 148, 4.24e+03, 1.66e+04, 2.72e+04, 2.73e+03, 3.04e+03, 4.53e+03,
01267    9.22e+03, 4.95e+03, 4.1e+04, 1.97e+04, 5.86e+04, 8.07e+04, 4.13e+04,
01268    3.04e+04, 8.53e+04, 1.96e+04, 146, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01269    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01270    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.92e+04,
01271    2.18e+06, 2.32e+06, 5.37e+05, 7.08e+05, 4.37e+05, 2.17e+05, 1.09e+05,
01272    1.92e+05, 9.87e+05, 1.7e+06, 9.46e+05, 1.1e+06, 2.63e+05, 718, 0, 0, 0,
01273    1.56e+04, 6e+04, 2.15e+04, 0, 0, 0, 0, 0, 0, 0, 0, 4.02e+04, 383, 655,
01274    1.65e+03, 1.35e+03, 0.0663, 2.21, 3.64, 0, 0.98, 0, 0, 0, 0, 0, 0, 40.5,
01275    249, 1.19e+03, 1.05e+03, 4.17e+03, 2.22e+04, 1.59e+04, 1.68e+04, 7.88e+03,
01276    2.49e+04, 4.62e+03, 6.25e+03, 7.78e+03, 971, 456, 255, 1.54e+03, 2.02e+03,
01277    2.35e+03, 1.41e+03, 2.16e+03, 4.33e+03, 0, 0, 7.75, 264, 3.95, 1.32e+03,
01278    4.59e+03, 4.01e+03, 859, 5.2e+03, 0, 0, 0, 1.76, 6.11e+03, 1.62e+04,
01279    8.16e+04, 2.46e+05, 1.86e+04, 0, 5.2, 3.8e+04, 1.84e+05, 2.05e+05,
01280    1.61e+05, 1.37e+05, 1.13e+05, 1.07e+05, 0, 0, 0, 0, 0, 0, 0},
01281   {280, 150, 4.12e+03, 1.84e+04, 2.98e+04, 3.56e+03, 2.58e+03, 5.21e+03,
01282    1.2e+04, 3.43e+03, 2.33e+04, 9.79e+03, 5.49e+04, 5.97e+04, 1.83e+04,
01283    5.49e+04, 1.47e+05, 153, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01284    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01285    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3e+04, 1.51e+06,
01286    1.95e+06, 1.94e+05, 8.02e+05, 1.57e+06, 1.7e+05, 1.77e+05, 2.07e+04,
01287    8.26e+03, 1.16e+03, 4e+03, 5.7e+03, 4.34e+04, 1.03e+06, 8.04e+05, 5.65e+05,
01288    3.55e+05, 8.55e+04, 5.14e+04, 4.63e+04, 7.56e+04, 3.78e+03, 0, 0, 0, 0, 0,
01289    0, 0, 0, 7.7e+04, 0, 6.07e+04, 1.35e+03, 7.05, 0, 0.0411, 1.64, 7.57, 1.26,
01290    0, 0, 0, 0, 0, 0, 0, 0, 35.9, 56.9, 151, 408, 3.38e+03, 5.92e+03, 1.98e+04,
01291    1.15e+04, 1.48e+04, 898, 5.03e+03, 6.34e+03, 1.19e+03, 1.91e+03, 1.64e+03,
01292    1.03e+03, 1.54e+03, 1.43e+03, 3.6e+03, 3.42e+03, 1.73e+03, 6.57e+03,
01293    1.13e+03, 4.57e+03, 1.45e+03, 1.12e+03, 6.11e+03, 1.41e+04, 3.82e+03,
01294    7.95e+03, 9.13e+03, 0, 62.9, 552, 0.222, 1.86e+04, 1.6e+04, 1.24e+05,
01295    1.42e+05, 1.07e+03, 0, 0, 0, 1.39e+05, 2.34e+05, 2.58e+05, 1.44e+05,
01296    2.17e+05, 2.19e+05, 0, 0, 0, 0, 0, 0, 0},
01297   {289, 154, 3.98e+03, 2.04e+04, 3.05e+04, 4.99e+03, 2.47e+03, 5.98e+03,
01298    1.4e+04, 8.09e+03, 2.89e+03, 245, 3.14e+04, 3.63e+03, 8.51e+03, 5.94e+04,
01299    1.78e+05, 1.43e+04, 0, 0, 2.34e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01300    0, 0, 0, 0, 0, 0, 316, 1.08e+03, 167, 2.04e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01301    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.38e+04,
01302    4.73e+05, 2.59e+06, 4.49e+05, 5.76e+05, 1.2e+06, 4.89e+05, 6.3e+04,
01303    7.56e+04, 1.56e+04, 898, 405, 254, 229, 393, 1.03e+03, 8.1e+04, 6.95e+05,
01304    9.13e+05, 8.87e+05, 5.32e+05, 6.21e+05, 5.82e+05, 9.15e+04, 2.78e+03, 211,
01305    0, 0, 0, 0, 0, 0, 0, 8.04e+03, 0, 3.09e+04, 1.19e+04, 0.16, 0, 0.0411,
01306    0.794, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.376, 1.73, 4.35, 34.3, 84.1, 4.21e+03,
01307    2.19e+04, 7.83e+03, 2.24e+04, 2.22e+04, 874, 5.46e+03, 5.58e+03, 1.64e+03,
01308    1.73e+03, 1.28e+03, 995, 685, 697, 1.99e+03, 1.34e+03, 3.73e+03, 1.62e+03,
01309    294, 684, 954, 3.76e+03, 1.38e+04, 0, 150, 2.47e+03, 0, 37.8, 508,
01310    1.91e+03, 8.62e+03, 1.23e+04, 1.43e+05, 1.96e+04, 0, 0, 0, 0, 6.86e+04,
01311    1.56e+05, 2.33e+05, 1.26e+05, 1.61e+05, 9.23e+04, 0, 0, 0, 0, 0, 0, 0},
01312   {297, 161, 3.87e+03, 2.23e+04, 3.03e+04, 7.18e+03, 2.54e+03, 7.32e+03,
01313    1.04e+04, 1.19e+04, 0, 0, 2.45e+04, 1.64e+03, 4.83e+03, 4.47e+04, 2.02e+05,
01314    2.9e+04, 1.57e+04, 0, 1.05e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01315    0, 0, 842, 3.77e+03, 5.39e+03, 2.95e+04, 4.59e+04, 1.62e+04, 8.86e+03,
01316    3.19e+04, 291, 180, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01317    0, 0, 0, 0, 0, 0, 0, 0, 0, 5.94e+03, 1.03e+06, 1.13e+06, 2.51e+05,
01318    8.52e+05, 1.11e+06, 1.33e+05, 5.45e+04, 7.62e+03, 3.27e+03, 1.16e+03, 295,
01319    191, 69, 86.8, 344, 513, 5.56e+03, 2.93e+05, 4.37e+05, 6e+05, 1.4e+06,
01320    6.56e+05, 3.59e+05, 3.31e+04, 2.16e+03, 9.61e+03, 2.06e+03, 0, 0, 0, 0, 0,
01321    0, 0, 0, 5.08e+04, 1.46e+04, 0, 0.0653, 2.04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01322    0, 0, 0, 9.91, 39.9, 1.69e+03, 1.97e+04, 1.82e+04, 1.89e+04, 2.51e+04,
01323    1.04e+04, 1.41e+04, 6.15e+03, 2.73e+03, 1.57e+03, 2.07e+03, 3.31e+03,
01324    2.07e+03, 1.17e+03, 304, 389, 1.2e+03, 2.24e+03, 2.12e+03, 372, 661,
01325    2.76e+03, 2.05e+04, 0.00222, 8.21, 801, 0, 0.172, 43.8, 7.6e+03, 2.41e+04,
01326    2.61e+04, 1.64e+05, 887, 0, 0, 0, 0, 4.24e+03, 6.21e+04, 3.18e+05,
01327    1.58e+05, 1.79e+05, 9.82e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01328   {305, 165, 3.75e+03, 2.38e+04, 2.93e+04, 9.82e+03, 2.48e+03, 1.01e+04,
01329    6.91e+03, 1.18e+04, 0, 0, 1.65e+04, 5.07e+03, 1.31e+04, 2.85e+04, 2.51e+05,
01330    4.69e+03, 1.98e+04, 3.76e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01331    0, 111, 7.14e+03, 2.08e+04, 1.65e+05, 7.67e+05, 8.91e+05, 1.58e+05,
01332    1.05e+06, 5.05e+04, 2.13e+04, 1.29e+03, 1.22e+04, 1.73e+04, 2.32e+04,
01333    1.37e+04, 0, 947, 657, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01334    0, 0, 0, 1.21e+06, 7.2e+05, 2.38e+05, 1.2e+06, 9.71e+05, 9.51e+04,
01335    5.33e+04, 6.61e+03, 4.95e+03, 4.41e+03, 1.1e+03, 54, 65.1, 109, 110, 174,
01336    538, 1.71e+03, 1.63e+04, 4.48e+05, 1.41e+06, 4.35e+05, 7.42e+05, 9.52e+04,
01337    2.01e+05, 4.8e+04, 6.99e+04, 1.57e+06, 0, 0, 0, 0, 0, 0, 7.28e+04, 1.2e+04,
01338    4.4e+03, 0.0479, 0.616, 0.162, 0.254, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01339    0, 0, 0.073, 270, 9.17e+03, 4.61e+04, 2.98e+04, 3.85e+04, 4.06e+04,
01340    3.48e+04, 2.72e+03, 2.99e+03, 5.93e+03, 1.66e+03, 3.3e+03, 2.45e+03,
01341    4.05e+03, 1.35e+03, 1.04e+03, 1.03e+03, 1.43e+03, 2.06e+03, 1.12e+03,
01342    1.3e+03, 5.54e+03, 2.48e+04, 14.3, 12.8, 3.24e+03, 14.3, 0.00772, 27.7,
01343    1.6e+04, 7.94e+04, 5.27e+04, 6.37e+04, 0, 0, 0, 0, 0, 0, 2.06e+03,
01344    3.02e+05, 2.24e+05, 1.75e+05, 1.77e+05, 0, 0, 0, 0, 0, 0, 0},
01345   {315, 170, 3.65e+03, 2.45e+04, 2.81e+04, 1.3e+04, 2.23e+03, 1.45e+04,
```

```
01346    4.7e+03, 1.15e+04, 0, 0, 2.26e+03, 7.36e+03, 2.78e+04, 6.84e+04, 1.68e+05,
01347    2.01e+03, 4.31e+04, 3.22e+04, 1.71e+03, 1.11e+03, 0, 0, 0, 0, 0, 0, 0,
01348    0, 0, 0, 0, 0, 0, 6.02e+04, 8.37e+04, 8.27e+04, 2.82e+05, 3.06e+05,
01349    6.88e+05, 1.34e+05, 5.99e+05, 1.43e+05, 1.91e+05, 1.8e+05, 1.57e+05,
01350    7.11e+04, 3.21e+04, 2.19e+04, 2.12e+04, 7.15e+04, 2.43e+04, 1.84e+04, 0, 0,
01351    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.78e+04, 1.53e+06,
01352    4.71e+05, 3.95e+05, 1.5e+06, 4.74e+05, 5.59e+04, 7.57e+03, 1.85e+03, 373,
01353    362, 1.62e+03, 747, 11.2, 55.2, 20.6, 1.82e+03, 1.86e+04, 3.79e+04,
01354    9.09e+03, 913, 3.01e+03, 8.4e+05, 4.86e+05, 1.07e+06, 6.85e+05, 2.34e+05,
01355    6.88e+05, 4.36e+05, 37, 0, 0, 0, 0, 0, 3.98e+04, 3.29e+04, 3.89e+04, 0.395,
01356    0.264, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.68e+03,
01357    2.35e+04, 3.52e+04, 4.57e+04, 3.04e+04, 4.41e+03, 1.1e+04, 4.83e+03,
01358    1.31e+04, 6.83e+03, 5.99e+03, 6.33e+03, 4.27e+03, 2.05e+03, 825, 1.65e+03,
01359    370, 3.65e+03, 859, 2.92e+03, 6.24e+03, 5.25e+04, 0, 5.99e+03, 4.17e+03,
01360    1.14e+03, 91.9, 774, 1.19e+04, 4.3e+04, 1.33e+05, 1.63e+04, 0, 0, 0, 0,
01361    28.1, 6.55e+03, 3.08, 2e+05, 1.84e+05, 2.4e+05, 8.78e+04, 0, 0, 0, 0, 0, 0,
01362    0},
01363   {324, 180, 3.52e+03, 2.43e+04, 2.69e+04, 1.76e+04, 3.42e+03, 1.69e+04,
01364    2.66e+03, 1.37e+04, 104, 170, 0, 9.48e+03, 5.19e+04, 1.01e+05, 1e+05,
01365    2.13e+04, 3.22e+04, 2.91e+04, 3.24e+04, 1.48e+05, 0, 0, 0, 0, 0, 0, 0, 0,
01366    0, 0, 0, 0, 0, 1.78e+03, 4.3e+04, 3.84e+04, 3.12e+04, 8.58e+04, 6.87e+04,
01367    1.19e+05, 1.33e+05, 2.47e+05, 1.24e+05, 8.61e+04, 1.58e+05, 2.1e+05,
01368    2.72e+05, 1.9e+05, 2.26e+05, 2.21e+05, 2.13e+05, 2.02e+05, 1.62e+04,
01369    1.93e+04, 1.24e+04, 5.79e+03, 2.28e+03, 2.6e+04, 1.01e+03, 0, 0, 0, 0, 0,
01370    0, 0, 0, 0, 0, 0, 0, 5.48e+05, 1.57e+06, 1e+05, 2.07e+05, 1.02e+06,
01371    1.5e+06, 6.03e+03, 4.17e+03, 1.97e+03, 364, 83.6, 1.72e+03, 882, 2.72,
01372    5.01, 937, 345, 1.82e+03, 2.39e+03, 281, 973, 819, 3.77e+03, 1.13e+06,
01373    1.43e+06, 7.15e+05, 4.87e+04, 1.37e+03, 1.8e+04, 4.48e+03, 0, 0, 0, 0, 0,
01374    2.94e+05, 1.25e+05, 2.22e+04, 0, 0.0353, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01375    0, 0, 0, 0, 0, 0, 0, 342, 5.17e+03, 1.51e+04, 3.59e+04, 1.77e+04, 1.21e+04,
01376    4.7e+04, 2.3e+04, 9.72e+03, 1.46e+04, 5.41e+03, 3.77e+03, 3.08e+03,
01377    4.89e+03, 1e+03, 2.21e+03, 2.15e+03, 4.7e+03, 312, 2.25e+03, 1.25e+04,
01378    9.45e+03, 3.28e+03, 1.39e+04, 926, 1.16e+03, 729, 1.09e+04, 9.82e+03,
01379    4.36e+04, 1.82e+05, 8.78e+03, 0, 0, 0, 0, 1.78e+04, 1.22e+05, 6.06e+03,
01380    1.77e+05, 1.25e+05, 2.36e+05, 1.5e+05, 0, 0, 0, 0, 0, 0, 0},
01381   {332, 188, 3.38e+03, 2.3e+04, 2.66e+04, 2.28e+04, 5.55e+03, 1.53e+04, 767,
01382    1.82e+04, 6.4e+03, 1.36e+04, 0, 5.73e+03, 7.89e+04, 6.08e+04, 7.7e+04,
01383    5.04e+04, 6.36e+04, 9.18e+04, 2.57e+05, 4.39e+05, 0, 0, 0, 0, 0, 0, 0, 0,
01384    0, 0, 0, 0, 0, 2.35e+04, 4.51e+04, 2.27e+04, 3.36e+03, 5.41e+03, 4.25e+04,
01385    4.15e+04, 1.19e+04, 1.27e+05, 2.5e+04, 4.48e+04, 1.68e+04, 7.18e+04,
01386    1.11e+05, 7.71e+04, 8.55e+04, 9.78e+04, 1.41e+05, 1.91e+05, 5.26e+05,
01387    7.88e+05, 2.53e+05, 3.2e+05, 9.36e+05, 1.11e+06, 8.51e+05, 6.73e+05,
01388    1.02e+05, 8.14e+04, 4.62e+04, 4.32e+04, 1.18e+04, 1.14e+03, 0, 0, 0, 0,
01389    1.57e+05, 1.57e+06, 1.21e+05, 1.01e+05, 4.49e+05, 1.07e+06, 2.26e+04,
01390    2.17e+03, 1.14e+03, 472, 23.1, 220, 1.72e+03, 5.19, 13.7, 19.9, 9.53, 675,
01391    5.29e+03, 2.13e+03, 1.85e+03, 769, 1.6e+03, 321, 229, 202, 5.64e+05,
01392    1.84e+06, 1.15e+05, 3.76e+04, 2.41e+03, 0, 0, 0, 0, 0, 6.73e+04, 3.2e+05,
01393    2.27e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.91,
01394    726, 5.09e+03, 5.77e+04, 2.6e+04, 1.08e+04, 8.94e+04, 8.35e+04, 2.79e+04,
01395    1.12e+04, 3.63e+03, 7.74e+03, 1.78e+03, 3.55e+03, 1.84e+03, 4.19e+03,
01396    3.31e+03, 3.87e+03, 840, 2.66e+03, 2.4e+04, 0, 1.51e+04, 8.36e+03, 905,
01397    2.65e+03, 2.43e+03, 8.15e+03, 1.65e+04, 9.12e+04, 1.43e+05, 0, 0, 0, 0, 0,
01398    4.37e+04, 1.81e+05, 1.79e+04, 1.32e+05, 1.18e+05, 1.91e+05, 2.69e+05, 0, 0,
01399    0, 0, 0, 0, 0},
01400   {342, 196, 3.25e+03, 2.13e+04, 2.69e+04, 2.88e+04, 5.71e+03, 1.5e+04, 662,
01401    1.82e+04, 1.12e+03, 2.34e+04, 0, 99.6, 2.04e+05, 6.1e+04, 6.95e+04,
01402    5.76e+04, 5.63e+04, 1.54e+05, 4.04e+05, 1.26e+05, 0, 0, 0, 0, 0, 0, 0, 0,
01403    0, 0, 0, 0, 0, 1.48e+05, 3.49e+04, 1.39e+04, 4.35e+03, 8.48e+03, 2.17e+04,
01404    6.35e+04, 1.08e+04, 5.08e+04, 1.12e+04, 3.79e+04, 3.23e+04, 5.24e+04,
01405    2.96e+04, 3.75e+04, 4.48e+04, 5.38e+04, 7.36e+04, 3.82e+05, 2.66e+05,
01406    5.37e+05, 7.1e+05, 7.33e+05, 6.74e+05, 6.85e+05, 5.26e+05, 7.97e+05,
01407    1.21e+06, 7.65e+05, 3.8e+05, 2.53e+05, 2.28e+05, 2.72e+05, 4.49e+05,
01408    2.02e+05, 3.61e+05, 1.05e+06, 2.42e+06, 3.49e+05, 9.16e+04, 9.59e+04,
01409    2.07e+06, 1e+05, 817, 1.12e+03, 962, 73.3, 34.2, 703, 876, 93.3, 57.1,
01410    49.1, 20.1, 365, 2.76e+03, 1.62e+03, 946, 711, 950, 336, 109, 90.6, 346,
01411    4.67e+05, 2.3e+05, 9.19e+04, 869, 8.86, 0, 0, 0, 0, 1.4e+04, 5.22e+04,
01412    7.27e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.57,
01413    26, 3.72, 3.63e+03, 3.09e+04, 7.41e+03, 2.51e+04, 4.26e+04, 2.08e+04,
01414    7.97e+03, 7.56e+03, 4.77e+03, 1.62e+03, 1.08e+03, 2.8e+03, 6.86e+03,
01415    4.81e+03, 4.27e+03, 5.04e+03, 4.19e+03, 9.46e+03, 0, 2.6e+04, 2.33e+04,
01416    6.02e+03, 8.79e+03, 1.23e+04, 3.34e+03, 6.67e+04, 1.56e+05, 3.49e+04, 0, 0,
01417    0, 0, 943, 8.35e+04, 1.33e+05, 6.17e+04, 5.26e+04, 1.57e+05, 2.11e+05,
01418    1.81e+05, 0, 0, 0, 0, 0, 0, 0, 0},
01419   {351, 207, 3.13e+03, 1.96e+04, 2.65e+04, 3.61e+04, 6.51e+03, 1.62e+04, 750,
01420    1.45e+04, 0, 1.77e+04, 0, 0, 2.65e+05, 5.92e+04, 8.69e+04, 1.07e+05,
01421    6.66e+04, 1.16e+05, 2.3e+05, 4.26e+03, 7.42e+04, 4.58e+04, 0, 0, 0, 0, 0,
01422    0, 0, 0, 0, 0, 36, 1.3e+05, 2.19e+04, 6.74e+03, 1.79e+03, 5.97e+03,
01423    1.07e+04, 1.96e+04, 3.24e+04, 7e+04, 2.12e+04, 3.64e+04, 4.25e+04,
01424    7.74e+04, 3.28e+04, 1.31e+04, 6.49e+04, 2.1e+04, 9.15e+04, 1.18e+05,
01425    1.43e+05, 5.15e+04, 3.65e+05, 7.33e+05, 8.69e+05, 5.56e+05, 7.75e+05,
01426    6.34e+05, 3.65e+05, 3.64e+05, 3.51e+05, 4.58e+05, 3.44e+05, 2.96e+05,
01427    8.2e+05, 1.63e+06, 9.26e+05, 2.05e+05, 1.15e+05, 4.27e+04, 1.07e+05,
01428    1.37e+06, 6.78e+05, 1.35e+04, 1.58e+03, 2.15e+03, 476, 562, 1.82e+03, 520,
01429    19, 25.4, 94.8, 84.1, 120, 1.46e+03, 1.42e+03, 166, 2.07e+03, 6.71e+03,
01430    321, 251, 239, 66.5, 112, 1e+03, 1.17e+05, 3.94e+04, 88.7, 0, 0, 0, 0, 0,
01431    263, 8.25e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01432    0, 0, 0, 0, 1.68e+03, 5.21e+03, 1.99e+04, 6.59e+03, 2e+04, 1.82e+04,
```

```
01433    9.54e+03, 1.08e+04, 1.21e+04, 1.15e+03, 747, 6.63e+03, 8.74e+03, 7.15e+03,
01434    3.07e+03, 2.33e+03, 0, 7.86, 1.63e+03, 4.04e+04, 2.92e+04, 5.1e+03,
01435    1.15e+04, 1.15e+04, 9.7e+03, 1.37e+05, 1.44e+05, 5.59e+03, 0, 0, 0, 0,
01436    1.83e+04, 1.26e+05, 1.18e+05, 1.04e+05, 2.42e+03, 1.36e+05, 1.63e+05,
01437    9.51e+04, 0, 0, 0, 0, 0, 0, 0},
01438   {358, 217, 3.01e+03, 1.78e+04, 2.78e+04, 4.38e+04, 9.88e+03, 1.5e+04,
01439    1.18e+03, 7.69e+03, 0, 8.46e+03, 4.74, 0, 3.27e+05, 6.1e+04, 7.38e+04,
01440    1.69e+05, 7.49e+03, 2.55e+04, 8.62e+04, 2.65e+04, 1.55e+04, 1.06e+05, 0, 0,
01441    0, 0, 0, 0, 0, 0, 0, 0, 525, 4.6e+04, 1.85e+04, 174, 29.9, 0, 1.22e+03,
01442    7.53e+03, 8.37e+03, 5.39e+03, 5.2e+04, 1.71e+04, 1.32e+04, 1.41e+05,
01443    3.42e+04, 1.95e+04, 5.17e+04, 2.21e+04, 9.69e+03, 2.37e+04, 1.41e+05,
01444    5.77e+04, 1.58e+04, 3.37e+03, 5.04e+04, 2.91e+05, 2.01e+05, 8.4e+05,
01445    1.05e+06, 9.45e+05, 3.25e+05, 2.44e+05, 6.58e+05, 1.07e+06, 3.32e+05,
01446    1.17e+05, 4.08e+04, 1.29e+04, 3.7e+04, 2.65e+05, 1.66e+06, 2.43e+05,
01447    1.23e+05, 1.15e+03, 122, 1.61e+03, 226, 1.65e+03, 3.7, 20.2, 57.9, 31.8,
01448    53.3, 125, 173, 970, 200, 701, 2.69e+03, 8.12e+03, 1.73e+04, 6.11e+04,
01449    4.56e+03, 279, 169, 791, 9.44e+04, 1.63e+03, 5.64, 0, 0, 0, 0, 0, 3.97,
01450    16.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01451    0, 169, 3.15e+03, 2.96e+03, 3.63e+03, 8.66e+03, 1.06e+04, 1.83e+04,
01452    1.32e+04, 2.06e+04, 1.53e+03, 197, 7.58e+03, 9.31e+03, 3.06e+03, 5.14e+03,
01453    8.82, 2.6e+03, 647, 1.41e+04, 7.89e+04, 5.43e+04, 1.65e+04, 8.86e+03,
01454    2.88e+04, 4.72e+04, 5.37e+04, 2.1e+04, 3.81, 0, 0, 0, 0, 8.7e+04, 1.58e+05,
01455    1.64e+05, 8.55e+04, 0.25, 1.78e+05, 1.26e+05, 1.05e+05, 0, 0, 0, 0, 0, 0,
01456    0},
01457   {364, 230, 2.91e+03, 1.61e+04, 3.11e+04, 5.27e+04, 1.12e+04, 1.24e+04,
01458    3.69e+03, 2.08e+03, 0, 21.9, 0.243, 0, 2.53e+05, 1.11e+05, 2.55e+04,
01459    1.52e+05, 1.06e+05, 6.65e+04, 5.48e+04, 1.47e+05, 2.19e+05, 1.45e+05, 0, 0,
01460    0, 0, 0, 0, 0, 0, 0, 0, 292, 4.42e+04, 1.11e+04, 0, 0, 0, 0, 914, 2.13e+03,
01461    8.34e+03, 3.02e+03, 3.13e+04, 3.4e+03, 2.5e+04, 5.39e+04, 9.99e+04,
01462    1.11e+04, 3.2e+03, 1.56e+03, 3.82e+03, 2.16e+03, 1.71e+03, 3.07e+03,
01463    1.06e+04, 9.96e+03, 1.2e+05, 1.42e+05, 7.09e+05, 3.95e+05, 6.74e+05,
01464    2.87e+05, 1.58e+05, 1.44e+05, 9.57e+04, 1.29e+05, 1.75e+04, 9.37e+04,
01465    8.34e+04, 7.63e+04, 7.87e+05, 3.43e+05, 1.74e+05, 723, 70.4, 368, 478,
01466    1.16e+03, 1.61e+03, 6.87, 61.2, 21.9, 23.9, 120, 163, 160, 1.27e+03,
01467    1.71e+03, 618, 3.83e+03, 1.79e+03, 2.23e+05, 1.08e+05, 2.42e+05, 9.68e+04,
01468    413, 1.19e+03, 1.42e+05, 1.14e+05, 0.00111, 0, 0, 0, 0, 0, 3.54e+04, 35.7,
01469    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01470    3.07e+03, 6.31e+03, 1.21e+04, 1.01e+04, 4.33e+04, 2.63e+04, 1.55e+04,
01471    1.3e+04, 3.03e+03, 467, 2.36e+03, 6.21e+03, 3.58e+03, 5.69e+03, 192, 0, 0,
01472    4.29e+04, 7.18e+04, 8.72e+03, 5.89e+04, 3.71e+03, 1.43e+05, 1.73e+05,
01473    3.06e+03, 164, 0, 0, 0, 0, 0, 7.13e+04, 1.12e+05, 2.96e+05, 1.45e+05,
01474    1.05e+04, 8.51e+04, 1.58e+05, 1.21e+05, 0, 0, 0, 0, 0, 0, 0},
01475   {371, 240, 2.8e+03, 1.46e+04, 3.25e+04, 7.7e+04, 1.03e+04, 1.17e+04,
01476    4.41e+03, 53.6, 0, 0, 0, 0, 5.57e+04, 1.6e+05, 2.83e+04, 4.18e+04,
01477    2.69e+04, 2.91e+04, 5.94e+04, 1.62e+05, 2.29e+05, 4.27e+05, 2.09e+04, 0, 0,
01478    0, 0, 0, 0, 0, 0, 0, 0, 2.5e+04, 4.77, 0, 0, 0, 0, 0, 1.93e+03, 2.89e+03,
01479    0, 1.71e+04, 4.08e+03, 3.12e+03, 1.95e+04, 3.01e+04, 4.27e+03, 1.47e+03,
01480    842, 1.26e+03, 389, 2.64e+03, 8.08e+03, 9.66e+04, 4.45e+04, 5.59e+04,
01481    1.6e+04, 2.56e+05, 5.67e+05, 1.26e+06, 7.98e+05, 1.04e+06, 3e+05, 9.01e+04,
01482    1.17e+05, 1.86e+05, 2.37e+05, 2.57e+05, 3.02e+05, 1.73e+06, 1.41e+05,
01483    5.06e+04, 905, 1.13e+03, 744, 860, 1.1e+03, 1.84e+03, 19.5, 50.7, 76.2,
01484    58.4, 208, 59.5, 5.58, 303, 3.94e+03, 1.42e+05, 3.36e+04, 1.44e+05,
01485    2.31e+05, 1.27e+05, 1.41e+05, 1.42e+05, 2.81e+03, 1.47e+03, 4.1e+04,
01486    1.2e+04, 0, 0, 0, 0, 0, 0, 2.89e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01487    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.55e+03, 3.68e+03, 2.92e+03, 1.72e+04,
01488    2.39e+04, 6.73e+04, 6.72e+03, 2.56e+04, 4.24e+03, 805, 2.24e+03, 2.3e+03,
01489    5.14e+03, 1.37e+03, 6.41e+03, 7.71e+03, 0.0544, 0, 1.34e+04, 4.59e+04,
01490    2.44e+04, 4.9e+04, 8.18e+04, 2.3e+05, 1.86e+05, 0, 0, 0, 0, 0, 0, 0,
01491    5.19e+04, 9.93e+04, 2.54e+05, 2.09e+05, 2.79e+04, 4.97e+04, 1.05e+05,
01492    1.05e+05, 0, 0, 0, 0, 0, 0, 0, 0},
01493   {378, 252, 2.73e+03, 1.35e+04, 3.21e+04, 1.13e+05, 1.54e+04, 1.48e+04, 748,
01494    0, 0, 0, 0, 0, 1.75e+04, 1.54e+05, 4.42e+04, 2.91e+04, 1.54e+04, 3.28e+04,
01495    3.77e+04, 4.58e+05, 4.25e+04, 5.06e+05, 4.29e+05, 0, 0, 0, 0, 0, 0, 0, 0,
01496    0, 0, 3.1e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 32.4, 3.79e+03, 2.87e+03,
01497    6.7e+03, 4.73e+03, 2.16e+03, 881, 2.58e+03, 3.44e+03, 1.85e+03, 2.42e+03,
01498    7.13e+03, 6.46e+04, 2.56e+05, 1.75e+05, 6.73e+03, 1.65e+04, 2.56e+04,
01499    5.51e+04, 2.08e+05, 1.56e+05, 7.74e+05, 1.65e+06, 9.77e+05, 3.14e+05,
01500    3.75e+05, 2.68e+05, 2.99e+05, 4.25e+05, 238, 144, 22.5, 69.1, 699,
01501    1.88e+03, 1.37e+03, 4.11e+03, 24, 75.8, 94.4, 168, 230, 7.54, 16.9, 20.8,
01502    3.5e+03, 4.01e+04, 2.57e+04, 7.48e+04, 1.79e+05, 2.17e+05, 3.65e+05,
01503    8.89e+04, 1.16e+04, 3.75e+03, 7.41e+03, 0.139, 0, 0, 0, 0, 0, 0, 54.8, 0,
01504    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.285, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01505    2.26e+03, 8.07e+03, 9.61e+03, 1.1e+03, 1.11e+04, 3.2e+04, 3.33e+03,
01506    3.11e+04, 3.59e+03, 2.89e+03, 2.09e+03, 3.52e+03, 9.48e+03, 1.36e+04,
01507    2.24e+04, 5.13e+04, 4.47e+04, 233, 6.15e+03, 2.53e+04, 4.61e+04, 1.57e+04,
01508    1.13e+05, 1.93e+05, 1.05e+05, 0, 0, 0, 0, 0, 0, 0, 4.41e+04, 4.51e+04,
01509    1.22e+05, 1.81e+05, 4.02e+04, 6.76e+04, 6.72e+04, 5.9e+04, 0, 0, 0, 0, 0,
01510    0, 0},
01511   {388, 265, 2.67e+03, 1.27e+04, 3.03e+04, 1.03e+05, 1.36e+04, 2.46e+04, 47.3,
01512    0, 0, 0, 0, 0, 2.66e+03, 2.59e+05, 2.01e+05, 4.09e+04, 3.79e+04, 3.18e+04,
01513    4.94e+04, 2.32e+05, 1.39e+03, 9.25e+04, 3.8e+05, 1.21e+05, 9.63e+03, 0, 0,
01514    0, 0, 0, 0, 0, 0, 0, 0.247, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 509, 330,
01515    1.53e+03, 588, 1.24e+03, 4.22e+03, 966, 747, 376, 1.86e+03, 1.36e+04,
01516    1.37e+04, 9.31e+04, 4.97e+04, 1.13e+04, 1.01e+03, 1.2e+03, 9.08e+03,
01517    1.65e+04, 1.22e+05, 1.98e+04, 1.64e+05, 2.08e+05, 2.37e+05, 3.59e+05,
01518    2.73e+05, 1.01e+04, 250, 383, 136, 722, 3.26e+03, 2.14e+04, 3.32e+03, 124,
01519    27.2, 108, 137, 170, 168, 1.96, 1.65, 3.65, 260, 1.45e+04, 6.28e+04,
```

```
01520    5.05e+04, 1.34e+05, 1.2e+05, 2.18e+04, 2.58e+04, 2.02e+04, 3.57e+03,
01521    1.59e+05, 2.5e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01522    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 75.6, 4.25e+03, 5.25e+03, 67.4,
01523    40.3, 1.89e+03, 2.33e+03, 3.17e+04, 3.23e+03, 1.14e+04, 6.03e+03, 9.55e+03,
01524    6.36e+03, 5.33e+03, 2.18e+04, 1.11e+05, 4.52e+04, 0, 21.5, 262, 784,
01525    1.79e+04, 1.35e+05, 1.66e+05, 1.51e+04, 0, 0, 0, 0, 0, 0, 0, 3.36e+04,
01526    2.87e+04, 5.16e+04, 1.07e+05, 2.37e+04, 9.03e+04, 6.75e+04, 2.06e+04, 0, 0,
01527    0, 0, 0, 0, 0},
01528   {395, 278, 2.6e+03, 1.21e+04, 2.66e+04, 8.55e+04, 3.89e+04, 1.39e+04,
01529    0.0601, 0, 0, 0, 0, 0, 0, 11.1, 2.88e+05, 1.09e+05, 1.51e+05, 1.01e+05,
01530    1.44e+05, 4.32e+04, 574, 714, 2.62e+05, 4.12e+05, 2.47e+04, 7.91e+03, 0, 0,
01531    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 434, 291, 783,
01532    3.75e+03, 968, 122, 142, 148, 690, 1.28e+03, 1.32e+03, 476, 409, 291, 168,
01533    407, 334, 181, 729, 8.62e+03, 4.01e+04, 5.3e+04, 3.6e+04, 1.72e+04, 785,
01534    1.43e+04, 2.7e+03, 716, 1.2e+03, 2.28e+03, 4.04e+03, 2.15e+03, 654, 190,
01535    224, 315, 82.2, 97.9, 2.79, 1.29, 18.5, 61.1, 6.03e+03, 4.37e+04, 1.57e+04,
01536    1.49e+05, 4.78e+04, 1.2e+05, 9.09e+03, 3.68e+04, 960, 3.76e+04, 1.4e+04, 0,
01537    0, 0, 0, 0, 0.00662, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01538    0, 0, 0, 0, 0, 0, 0, 0, 0, 4.49e+03, 4.14e+03, 4.48, 0, 3.44e+03, 3.88e+03,
01539    2.64e+04, 4.99e+03, 1.45e+04, 7.03e+03, 1.51e+04, 1.18e+04, 2.74e+04,
01540    4.53e+04, 5.11e+04, 0, 0, 0, 0, 0, 1.25e+03, 1.4e+05, 1.53e+05, 0, 0, 0, 0,
01541    0, 0, 0, 0, 3.34e+04, 4.74e+04, 1.83e+04, 6.62e+04, 1.78e+04, 8.02e+04,
01542    4.64e+04, 185, 0, 0, 0, 0, 0, 0, 0},
01543   {402, 293, 2.53e+03, 1.17e+04, 2.28e+04, 5.75e+04, 4.95e+04, 404, 0, 0, 0,
01544    0, 0, 0, 0, 0, 1.5e+04, 2.03e+05, 1.62e+05, 1.74e+05, 1.52e+04, 6.56e+03,
01545    0, 0.00444, 2.42e+04, 3.12e+05, 8.48e+04, 4.9e+03, 0, 0, 0, 0, 0, 0, 0, 0,
01546    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17.5, 28.6, 2.71e+03, 1.27e+04,
01547    1.8e+03, 210, 81.1, 227, 211, 360, 210, 302, 115, 783, 509, 204, 259, 223,
01548    393, 805, 1.98e+03, 3.06e+03, 2.68e+03, 156, 6.71e+03, 6.03e+03, 1.53e+03,
01549    1.85e+03, 1.32e+03, 7.5e+03, 4.55e+03, 4.19e+03, 2.34e+03, 1.46e+03, 200,
01550    349, 514, 87.3, 6.13, 8.94, 26.4, 120, 7.38e+03, 4.31e+04, 5.34e+03,
01551    1.03e+05, 1.66e+05, 3.39e+05, 2.61e+04, 3.24e+04, 6.57, 112, 3.9e+03,
01552    1.05e+03, 0, 0, 0, 8.32e+03, 0.0121, 0.00662, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01553    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.48e+03, 3.86e+03, 1.16,
01554    0.00999, 390, 706, 8.45e+03, 6.35e+03, 4.19e+03, 1.31e+04, 1.85e+04,
01555    3.06e+04, 3.16e+04, 2.11e+04, 0, 0, 0, 0, 0, 0, 0, 2.38e+04, 1.18e+05, 0,
01556    0, 0, 0, 0, 0, 0, 0, 7.8e+04, 5.43e+04, 4.08e+03, 4.3e+04, 2.96e+04,
01557    4.72e+04, 2.55e+04, 9.05, 0, 0, 0, 0, 0, 0, 0},
01558   {403, 307, 2.46e+03, 1.14e+04, 1.91e+04, 4.04e+04, 6.15e+04, 53.4, 0, 0, 0,
01559    0, 0, 0, 0, 0, 0, 0, 8.36e+04, 6.21e+04, 1.5e+04, 407, 0, 0, 0, 6.11e+03,
01560    3.01e+05, 2.69e+05, 1.26e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 182, 0, 0,
01561    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.74e+03, 2.22e+03, 1.29e+03, 140, 93.3, 191,
01562    786, 343, 393, 104, 30.3, 52.2, 168, 154, 85.6, 185, 291, 399, 787,
01563    1.96e+03, 1.26e+03, 1.34e+03, 9.69e+03, 1.01e+04, 1.02e+04, 1.55e+04,
01564    9.14e+03, 3.57e+03, 4.61e+03, 4.09e+03, 3.1e+03, 299, 160, 421, 607, 465,
01565    64.8, 666, 211, 252, 674, 6.2e+03, 1.81e+03, 7.59e+04, 1.28e+05, 8e+04,
01566    1.17e+04, 3.09e+04, 0.695, 657, 4.96e+03, 0, 3.04e+03, 1.65e+03, 2.77e+03,
01567    58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01568    0, 0, 0, 0, 248, 1.4e+04, 2.63e+04, 0, 0, 501, 7.83e+03, 3.51e+03,
01569    7.51e+03, 5.44e+03, 7.44e+03, 9.05e+03, 2.73e+03, 0, 0, 0, 0, 0, 0, 0, 0,
01570    1.69, 3.75e+03, 0, 0, 0, 0, 0, 0, 0, 0, 1.2e+05, 6.63e+04, 4.52e+03,
01571    2.91e+04, 6.07e+04, 4.59e+04, 1.78e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01572   {411, 322, 2.42e+03, 1.15e+04, 1.69e+04, 3.32e+04, 6.49e+04, 694, 0, 0, 0,
01573    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.11e+03, 5.09e+05, 1.18e+04,
01574    622, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.29e+03, 85.1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01575    0, 0, 510, 2.25e+03, 3.07e+03, 342, 85.4, 261, 346, 412, 429, 145, 89.8,
01576    121, 94.6, 92.9, 165, 104, 107, 154, 267, 1.07e+03, 1.22e+03, 1.54e+04,
01577    3.46e+03, 6.74e+03, 1.87e+04, 2.5e+04, 1.16e+04, 3.06e+03, 4.31e+03,
01578    1.62e+03, 4.24e+03, 844, 256, 726, 358, 635, 675, 393, 100, 1.99e+03,
01579    3.25e+03, 6.77e+03, 4.75e+03, 3.7e+04, 6.57e+04, 1.06e+05, 8.34e+03,
01580    1.57e+03, 0.0606, 0, 0, 0, 553, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01581    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 879, 1.99e+03, 0, 0,
01582    359, 1.09e+04, 1.46e+03, 1.85e+04, 1.78e+04, 1.39e+04, 198, 0, 0, 0, 0, 0,
01583    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.48e+05, 5.56e+04, 5.15e+03,
01584    2.04e+04, 1.21e+05, 5.66e+04, 2.06e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01585   {417, 336, 2.49e+03, 1.2e+04, 1.56e+04, 2.7e+04, 3.12e+04, 1.37e+04, 0, 0,
01586    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.6e+05, 0, 7.98e+03, 0, 0,
01587    0, 0, 0, 0, 0, 0, 0, 0, 1.51e+03, 306, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 86.3,
01588    4.98e+03, 3.01e+03, 160, 52.8, 130, 246, 191, 103, 236, 187, 161, 73.4,
01589    28.9, 195, 84.5, 228, 227, 110, 2.39e+03, 3.06e+03, 1.91e+04, 3.66e+03,
01590    1.49e+04, 2.69e+04, 4.54e+03, 2.93e+03, 3.56e+03, 5.39e+03, 2.47e+03,
01591    3.97e+03, 2.8e+03, 1.39e+03, 701, 560, 596, 1.15e+03, 435, 405, 2.27e+03,
01592    2.9e+04, 1.57e+04, 5.77e+03, 4.22e+04, 8.53e+04, 1.69e+04, 5.73e+03, 422,
01593    0, 0, 0, 0, 2.05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01594    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.93e+04, 5.4e+03, 0,
01595    5.82e+03, 6.57e+03, 5.69e+03, 5.2e+04, 1.53e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01596    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 262, 2.32e+05, 4.11e+04, 4.83e+03, 1.58e+04,
01597    1.37e+05, 1.31e+05, 2.37e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01598   {421, 355, 2.68e+03, 1.29e+04, 1.39e+04, 2.45e+04, 2.65e+04, 8.72e+04, 0, 0,
01599    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.52e+05, 2.93e+04,
01600    1.08e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 674, 83.5, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01601    0, 0, 1.07, 2.09e+03, 329, 38.4, 88.5, 1.78e+03, 632, 653, 369, 270, 233,
01602    54.6, 52.5, 57.1, 153, 56.3, 601, 2.76e+03, 735, 67.3, 7.31e+03, 9.57e+03,
01603    751, 5.12e+03, 2.43e+04, 6.92e+03, 1.79e+03, 2.7e+03, 4.3e+03, 4.09e+03,
01604    4.38e+03, 1.1e+03, 4.89e+03, 1.22e+03, 485, 557, 427, 867, 2.14e+03,
01605    4.99e+03, 5.16e+03, 1.04e+04, 1.41e+03, 2.06e+03, 5.6e+03, 685, 7.22, 0, 0,
01606    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
01607    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.3e+04, 2.8e+04, 4.03e+04, 735,
01608    8.63e+03, 6.63e+03, 1.61e+04, 5.53e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01609    0, 0, 0, 0, 0, 0, 1.28, 2.87e+04, 1.6e+05, 3.22e+04, 4.15e+03, 1.36e+04,
01610    9.03e+04, 9.68e+04, 4.87e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01611   {426, 375, 2.96e+03, 1.39e+04, 1.17e+04, 2.41e+04, 2.04e+04, 1.19e+05, 0, 0,
01612    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 122, 1.3e+04, 1.17e+04, 0, 0,
01613    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.31,
01614    26.5, 12, 102, 1.94e+03, 581, 2.57e+03, 2.67e+03, 1.14e+03, 62.4, 61.3,
01615    701, 4.09e+03, 2.07e+03, 2.98e+03, 4.09e+03, 7.97e+03, 2.07e+04, 9.6e+03,
01616    259, 154, 99.3, 1.2e+04, 1.23e+04, 5.85e+03, 2.28e+03, 5.41e+03, 3.41e+03,
01617    3.64e+03, 4.36e+03, 1.06e+04, 7.62e+03, 3.6e+03, 2.04e+03, 465, 356, 874,
01618    7.16e+03, 2.3e+03, 2.68e+03, 5.52e+03, 4.25e+03, 1.68e+04, 4.78e+03, 19, 0,
01619    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01620    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.36e+04, 1.09e+04, 2.42e+04,
01621    1.65e+04, 9.25e+03, 8.9e+03, 3.72e+03, 38.8, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01622    0, 0, 0, 0, 0, 0, 0, 0, 2.54e+03, 1.01e+05, 4.5e+04, 2.62e+04, 3.65e+03,
01623    1.31e+04, 4.64e+04, 6.79e+04, 6.19e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01624   {435, 388, 3.33e+03, 1.47e+04, 9.76e+03, 2.3e+04, 2.82e+04, 8.44e+04, 4.37,
01625    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.17e+03, 2.26e+04, 0,
01626    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01627    0, 487, 2.14e+03, 1.45e+03, 3.47e+03, 3.66e+03, 2.1e+03, 996, 1.91e+03,
01628    4.27e+03, 8.28e+03, 4.08e+03, 4.12e+03, 1.67e+04, 1.99e+04, 8.79e+03,
01629    6.78e+03, 223, 213, 2.39e+03, 9.01e+03, 1.54e+04, 3.9e+03, 2.85e+03,
01630    4.56e+03, 2.93e+03, 7.21e+03, 1.01e+04, 1.07e+04, 4.55e+03, 3.84e+03,
01631    6.14e+03, 1.92e+03, 782, 1.56e+03, 1.25e+04, 5.34e+03, 1.58e+03, 3.8e+03,
01632    1.15e+04, 1.02e+04, 1.34e+03, 65.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01633    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01634    0, 0, 337, 9.95e+03, 4.99e+03, 7.89e+03, 7.07e+03, 1.2e+04, 3.45e+03, 78.6,
01635    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.27e+03, 1.04e+04,
01636    3.76e+04, 1.08e+05, 5.28e+04, 1.47e+04, 3.51e+03, 1.26e+04, 2.24e+04,
01637    1.12e+05, 6.48e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01638   {443, 405, 3.77e+03, 1.53e+04, 8.2e+03, 1.9e+04, 2.64e+04, 1.07e+05,
01639    2.19e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.85e+03,
01640    0, 4.13e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01641    0, 0, 0, 0, 3.73e+03, 6.17e+03, 5.04e+03, 1.84e+03, 1.12e+03, 1.46e+03,
01642    7.18e+03, 4.67e+03, 1.05e+04, 3.52e+03, 1.99e+03, 2.94e+03, 5.59e+03,
01643    3.9e+03, 9.84e+03, 1.63e+04, 3.66e+04, 1.46e+04, 1.94e+04, 1.89e+04,
01644    1.95e+04, 2.58e+03, 2.34e+03, 3.92e+03, 3.67e+03, 9.18e+03, 5.08e+03,
01645    5.44e+03, 4.07e+03, 4.57e+03, 3.76e+03, 4.88e+03, 2.21e+03, 3.31e+03,
01646    9.16e+03, 5.22e+03, 1.6e+03, 2.11e+03, 9.57e+03, 4e+03, 3.2e+03, 90.3, 0,
01647    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01648    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.66e+03, 5.02e+03, 6.19e+03,
01649    68.5, 0, 37.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.55e+04,
01650    6.79e+03, 6.95e+04, 8.34e+04, 7.11e+04, 5.53e+04, 2.07e+04, 7.18e+03,
01651    3.56e+03, 1.17e+04, 1.16e+04, 1.5e+05, 6.81e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01652   {445, 425, 4.26e+03, 1.55e+04, 7.11e+03, 1.49e+04, 1.37e+04, 1.05e+05,
01653    1.09e+04, 1.49e+03, 1.07e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01654    0, 0, 0, 785, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01655    0, 0, 0, 0, 0, 43.4, 2.18e+03, 3.75e+03, 4e+03, 5.1e+03, 1e+04, 3.98e+03,
01656    8.86e+03, 1.61e+04, 2.34e+04, 2.03e+04, 2.15e+03, 1.38e+03, 665, 7.01e+03,
01657    8.76e+03, 1.06e+04, 1.45e+04, 1.89e+04, 2.18e+04, 6.76e+03, 8.44e+03,
01658    1.09e+03, 744, 2.44e+03, 2.92e+03, 3.71e+03, 5.28e+03, 6.97e+03, 7.01e+03,
01659    3.65e+03, 1.43e+03, 499, 5.87e+03, 3.8e+03, 3.24e+03, 2.1e+03, 2.2e+03,
01660    3.18e+03, 4.78e+03, 1.92e+03, 98.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01661    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01662    0, 0, 3.62e+03, 6.13e+03, 2.32e+03, 46.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01663    0, 0, 0, 0, 8.41e+03, 1.98e+04, 648, 2.02e+04, 2.03e+05, 1.57e+05,
01664    1.58e+05, 9.18e+04, 9.83e+03, 2.22e+04, 1.21e+04, 3.63e+03, 3.63e+03,
01665    1.07e+04, 7.5e+03, 1.43e+05, 8.17e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01666   {447, 445, 4.75e+03, 1.52e+04, 6.44e+03, 1.25e+04, 4.45e+03, 6.73e+04,
01667    2.77e+04, 1.12e+04, 1.13e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01668    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01669    0, 0, 0, 0, 0, 9.07, 35.9, 9.66e+03, 7.52e+03, 1.47e+04, 1.12e+04,
01670    6.81e+03, 1.94e+04, 1.95e+04, 1.42e+04, 1.13e+04, 2.58e+03, 2.25e+03,
01671    1.74e+03, 3.67e+03, 8.27e+03, 8.01e+03, 1.65e+04, 9.99e+03, 7.93e+03,
01672    1.77e+04, 377, 709, 1.96e+03, 5.45e+03, 7.39e+03, 6.54e+03, 5.93e+03,
01673    6.63e+03, 3.35e+03, 483, 499, 1.76e+03, 9.39e+03, 5.56e+03, 3.49e+03,
01674    1.11e+04, 1.12e+04, 3.72e+03, 2.32e+03, 0.896, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01675    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01676    0, 0, 0, 0, 0, 0, 0, 5.89e+03, 3.39e+03, 480, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01677    0, 0, 0, 0, 81.2, 1.77e+05, 8.78e+04, 1.49e+04, 3.08e+04, 2.36e+05,
01678    3.59e+05, 2.9e+05, 4.14e+04, 6.77e+03, 1.03e+04, 7.27e+03, 1.61e+03,
01679    3.67e+03, 9.69e+03, 5.61e+03, 9.39e+04, 8.25e+04, 0, 0, 0, 0, 0, 0, 0, 0},
01680   {448, 463, 5.26e+03, 1.45e+04, 6.17e+03, 1.14e+04, 1.35e+03, 3.96e+04,
01681    4.01e+04, 1.45e+04, 1.19e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01682    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01683    0, 0, 0, 0, 0, 0, 0, 2.54, 197, 1.05e+04, 1.12e+04, 3.6e+04, 8.79e+03,
01684    6.52e+04, 2.54e+04, 1.9e+04, 1.52e+04, 1.07e+04, 3.27e+03, 1.69e+03,
01685    5.4e+03, 4.8e+03, 1.07e+04, 1.95e+04, 1.8e+04, 8.46e+03, 7.67e+03,
01686    3.68e+03, 2.63e+03, 1.66e+03, 4.49e+03, 1.08e+04, 1.71e+03, 5.25e+03,
01687    9.37e+03, 2.72e+03, 574, 1.52e+03, 348, 4.68e+03, 1.04e+04, 2.13e+03,
01688    6.81e+03, 3.5e+03, 1.64e+03, 736, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01689    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01690    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.63e+03,
01691    4.27e+04, 2.91e+05, 3.72e+04, 1.6e+04, 1.17e+03, 1.92e+05, 2.64e+05,
01692    3.38e+05, 2.87e+04, 4.42e+03, 6.22e+03, 5.02e+03, 796, 3.74e+03, 8.68e+03,
01693    5.31e+03, 1.2e+05, 8.69e+04, 0, 0, 0, 0, 0, 0, 0},
```

```
01694   {452, 481, 5.72e+03, 1.34e+04, 6.17e+03, 1.07e+04, 4.9e+03, 4.23e+04,
01695    5.53e+04, 1.45e+04, 2.21e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01696    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01697    0, 0, 0, 0, 0, 0, 0, 1.4, 33, 1.17e+04, 3.95e+04, 8.42e+03, 1.93e+04,
01698    1.64e+04, 2.33e+04, 1.59e+04, 9.85e+03, 3.65e+03, 3.21e+03, 2.51e+03,
01699    7.06e+03, 9.57e+03, 1.05e+04, 1.13e+04, 2.1e+03, 1.02e+03, 1.72e+03,
01700    3.4e+03, 2.96e+03, 8.98e+03, 6.11e+03, 5.13e+03, 6.68e+03, 2.83e+03,
01701    3.42e+03, 1.15e+03, 698, 2.18, 1.56, 201, 1.17e+03, 3.96e+03, 1.46e+03,
01702    423, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01703    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01704    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.65e+04, 1.16e+05, 5.04e+04, 1.45e+05,
01705    2.68e+04, 1.93e+04, 2.17e+04, 8.14e+04, 1.99e+05, 1.38e+05, 1.52e+04,
01706    3.7e+03, 4.59e+03, 2.76e+03, 456, 3.77e+03, 7.38e+03, 6.95e+03, 1.38e+05,
01707    7.06e+04, 0.557, 0, 0, 0, 0, 0, 0, 0},
01708   {452, 503, 6.11e+03, 1.21e+04, 6.4e+03, 1.02e+04, 1.13e+04, 1.98e+04,
01709    9.37e+04, 1.19e+04, 5.57e+03, 2.55e+04, 1.5e+03, 0, 0, 0, 0, 0, 0, 0, 0,
01710    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01711    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.717, 2.39e+04, 2.16e+05, 5.01e+04,
01712    5.67e+04, 3.24e+03, 2.02e+04, 3.34e+04, 5.11e+03, 4.91e+03, 2.46e+03,
01713    2.58e+03, 4.93e+03, 4.37e+03, 7.51e+03, 2.16e+04, 2.41e+04, 2.25e+03,
01714    1.87e+03, 982, 901, 1.55e+03, 3.05e+03, 5.07e+03, 1.11e+04, 2.24e+04,
01715    4.19e+03, 1.99e+03, 735, 2.29, 5.4, 0.258, 1.87, 0.177, 0, 0, 0, 0, 0, 0,
01716    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01717    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01718    0, 0, 0, 5.95e+03, 1.14e+05, 1.51e+05, 7.83e+04, 1.87e+04, 2.69e+04,
01719    5.87e+04, 4.98e+04, 8.18e+04, 7.14e+04, 3.56e+04, 4.91e+03, 2.72e+03,
01720    2.14e+03, 1.67e+03, 342, 3.56e+03, 6e+03, 9.6e+03, 1.98e+05, 3.07e+04,
01721    0.291, 0, 0, 0, 0, 0, 0, 0},
01722   {453, 527, 6.42e+03, 1.07e+04, 6.78e+03, 9.98e+03, 1.53e+04, 2.63e+04,
01723    5.98e+04, 9.33e+03, 5.04e+03, 9.55e+03, 1.47e+04, 0, 0, 0, 0, 0, 0, 0, 0,
01724    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01725    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.34e+03, 1.48e+05, 6.98e+04,
01726    1.63e+05, 8.84e+04, 3e+04, 8.91e+03, 2.93e+03, 3.56e+03, 4.8e+03, 4.91e+03,
01727    6.69e+03, 9.73e+03, 7.97e+03, 2.01e+04, 9.91e+03, 1.3e+04, 2.38e+03,
01728    1.38e+03, 2.27e+03, 4.18e+03, 4.67e+03, 4.99e+03, 3.48e+03, 3.37e+03, 194,
01729    369, 1.35e+03, 12.7, 1.39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01730    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01731    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 856, 2e+05,
01732    8.17e+04, 7.9e+04, 3.4e+04, 3.09e+04, 4.71e+04, 1.7e+04, 3.66e+04,
01733    4.43e+04, 4.62e+04, 2.36e+04, 2.24e+03, 1.73e+03, 1.59e+03, 961, 409,
01734    3.23e+03, 5.34e+03, 1.19e+04, 1.46e+05, 5.76e+04, 3.28e+03, 0, 0, 0, 0, 0,
01735    0, 0},
01736   {456, 551, 6.66e+03, 9.27e+03, 7.21e+03, 9.9e+03, 1.56e+04, 2.3e+04,
01737    3.8e+04, 7.18e+03, 8.2e+03, 1.06e+04, 1.61e+04, 0, 0, 0, 0, 0, 0, 0, 0,
01738    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01739    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 31.6, 24.9, 0.00442, 4.37e+04,
01740    8.42e+04, 7.29e+03, 1e+04, 5.15e+03, 1.68e+04, 1.06e+04, 1.09e+04,
01741    7.99e+03, 9.96e+03, 1.72e+04, 5.09e+04, 1.49e+04, 9.51e+03, 1.54e+04,
01742    1.13e+04, 5.48e+03, 4.05e+03, 3.91e+03, 5.46e+03, 5.79e+03, 3.87e+03,
01743    1.8e+03, 480, 62, 0.0147, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01744    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01745    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.1e+05, 1.29e+05,
01746    3.03e+04, 2.06e+04, 2.88e+04, 1.53e+04, 1.1e+04, 1.06e+04, 1.02e+04,
01747    1.97e+04, 1.37e+04, 9.31e+03, 1.51e+03, 1.07e+03, 1.16e+03, 535, 769,
01748    2.99e+03, 4.83e+03, 1.35e+04, 1.47e+05, 6.94e+04, 1.65e+04, 0, 0, 0, 0, 0,
01749    0, 0},
01750   {457, 578, 6.87e+03, 8.06e+03, 7.62e+03, 9.97e+03, 1.48e+04, 6.11e+04,
01751    1.18e+04, 3.44e+03, 1.37e+04, 1.47e+04, 1.48e+04, 0, 0, 0, 0, 0, 0, 0, 0,
01752    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01753    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 335, 1.34e+05,
01754    2.14e+04, 4.48e+04, 1.36e+04, 2.65e+04, 3.3e+03, 1.67e+04, 1.92e+04,
01755    1.58e+04, 2.28e+04, 4.24e+04, 6.03e+03, 8.62e+03, 2.17e+04, 1.88e+04,
01756    1.58e+04, 7.59e+03, 7.59e+03, 5.44e+03, 3.59e+03, 5.96e+03, 2.38e+03, 512,
01757    102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01758    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01759    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2e+05, 9.03e+04, 2.07e+04, 6.36e+03,
01760    7.17e+03, 8.05e+03, 7.21e+03, 4.88e+03, 5.87e+03, 9.36e+03, 7.81e+03,
01761    6.57e+03, 1.01e+03, 842, 817, 335, 1.17e+03, 2.91e+03, 4.19e+03, 1.25e+04,
01762    1.66e+05, 5.86e+04, 3.28e+04, 0, 0, 0, 0, 0, 0, 0},
01763   {458, 609, 7.01e+03, 7.17e+03, 7.97e+03, 9.66e+03, 1.4e+04, 8.32e+04, 525,
01764    257, 2.54e+04, 2.99e+04, 1.26e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01765    0, 0, 0, 5.41e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01766    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.23e+05, 9.96e+04,
01767    2.31e+04, 1.21e+04, 1.1e+04, 1.24e+04, 1.12e+04, 2.49e+03, 7.91e+03,
01768    6.57e+03, 7.53e+03, 7.88e+03, 4.77e+03, 9.44e+03, 2.47e+04, 7.32e+03,
01769    5.81e+03, 7.33e+03, 7.99e+03, 6.06e+03, 2.71e+03, 2.37e+03, 497, 3.42, 0,
01770    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01771    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01772    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.63e+05, 9.49e+04, 8.79e+03, 3.36e+03,
01773    4.87e+03, 5.82e+03, 4.26e+03, 3.67e+03, 3.89e+03, 7.24e+03, 6.61e+03,
01774    4.13e+03, 791, 675, 525, 276, 1.5e+03, 2.87e+03, 3.81e+03, 1.04e+04,
01775    1.86e+05, 4.45e+04, 2.58e+04, 0, 0, 0, 0, 0, 0},
01776   {461, 644, 7.09e+03, 6.47e+03, 8.38e+03, 8.43e+03, 1.51e+04, 8.41e+04, 29.2,
01777    0, 4.94e+04, 5.74e+04, 7.99e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01778    0, 0, 3.71e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01779    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.42e+04, 2.16e+05,
01780    8.07e+04, 6.62e+03, 9.85e+03, 5.74e+03, 1.06e+04, 4.11e+03, 3.16e+03,
```

```
01781    5.52e+03, 4.22e+03, 4.08e+03, 1.13e+04, 9.16e+03, 1.15e+04, 8.63e+03,
01782    4.49e+03, 7.06e+03, 7.33e+03, 1.24e+03, 868, 966, 556, 0.00772, 0, 0, 0, 0,
01783    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01784    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01785    0, 0, 0, 0, 0, 0, 4.3e+04, 2.8e+05, 1.23e+05, 1.61e+04, 2e+03, 5.44e+03,
01786    3.88e+03, 2.93e+03, 2.8e+03, 3.22e+03, 5.92e+03, 4.96e+03, 2.68e+03, 641,
01787    564, 385, 305, 1.77e+03, 2.73e+03, 3.54e+03, 8.41e+03, 1.9e+05, 5.58e+04,
01788    6.5e+04, 0, 0, 0, 0, 0, 0, 0},
01789   {460, 687, 7.1e+03, 5.87e+03, 8.83e+03, 7.02e+03, 1.9e+04, 7.34e+04,
01790    1.5e+03, 0, 4.7e+04, 3.34e+04, 940, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01791    0, 0, 0, 67.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01792    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.41e+04, 9.32e+04,
01793    1.44e+04, 2.67e+04, 4.84e+04, 6.24e+04, 1.44e+04, 1.32e+04, 8.23e+03,
01794    1.11e+04, 1.66e+04, 1.56e+04, 5.08e+03, 5.52e+03, 1.61e+04, 7.41e+03,
01795    5.83e+03, 7.59e+03, 2.22e+03, 1.02e+03, 896, 2.23, 0, 0, 0, 0, 0, 0, 0, 0,
01796    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01797    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01798    0, 0, 0, 2.71e+04, 6.36e+05, 4.39e+04, 8.53e+04, 4.71e+03, 8.28e+03,
01799    1.01e+03, 1.79e+03, 2.08e+03, 2.36e+03, 4.88e+03, 3.34e+03, 1.93e+03, 451,
01800    323, 243, 672, 1.93e+03, 2.37e+03, 3.31e+03, 7.02e+03, 1.64e+05, 5.44e+04,
01801    1.13e+05, 0, 0, 0, 0, 0, 0, 0},
01802   {460, 736, 7.03e+03, 5.33e+03, 9.07e+03, 4.94e+03, 1.87e+04, 5.81e+04,
01803    1.18e+04, 0, 6.93e+03, 3.14e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01804    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01805    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.25e+04, 9.14e+04,
01806    4.15e+04, 4.82e+04, 3.19e+04, 3.17e+04, 3.2e+04, 2.56e+04, 3.82e+04,
01807    3.22e+04, 2.97e+04, 4.64e+04, 3.3e+04, 3.13e+03, 6.87e+03, 4.02e+03,
01808    2.95e+03, 3.37e+03, 1.99e+03, 732, 847, 0.397, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01809    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01810    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01811    0, 0, 1.18e+03, 1.25e+05, 3.47e+05, 4.53e+05, 4.72e+04, 2.2e+04, 9.76e+03,
01812    844, 1.36e+03, 1.91e+03, 3.63e+03, 2.52e+03, 1.49e+03, 418, 298, 251,
01813    1.47e+03, 1.96e+03, 2.09e+03, 2.88e+03, 5.74e+03, 9.74e+04, 7.23e+04,
01814    9.97e+04, 0, 0, 0, 0, 0, 0, 0},
01815   {459, 793, 6.93e+03, 4.82e+03, 8.97e+03, 2.61e+03, 1.12e+04, 4.61e+04,
01816    3.85e+04, 113, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01817    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01818    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 793, 9.48e+04, 9.38e+04, 6.89e+04,
01819    3.33e+04, 3.61e+04, 3.08e+04, 2.22e+04, 1.94e+04, 3.49e+04, 5.37e+04,
01820    3.17e+04, 2.6e+04, 6.79e+03, 9.14e+03, 5.37e+03, 6.89e+03, 5.19e+03,
01821    6.08e+03, 6.5e+03, 2.04e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01822    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01823    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10,
01824    1.71e+04, 1.8e+05, 3.82e+05, 6.71e+04, 5.48e+04, 6.41e+03, 776, 1.69e+03,
01825    2.74e+03, 1.9e+03, 1.13e+03, 348, 226, 581, 1.93e+03, 2.11e+03, 1.99e+03,
01826    2.36e+03, 4.17e+03, 2.05e+04, 1.72e+05, 5.53e+04, 0, 0, 0, 0, 0, 0, 0},
01827   {455, 857, 6.76e+03, 4.37e+03, 8.61e+03, 1.21e+03, 6.39e+03, 5.67e+04,
01828    4.99e+04, 5.55e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01829    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01830    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 358, 6.42e+04, 8.38e+04,
01831    2.71e+04, 2.04e+04, 2.38e+04, 4.68e+04, 2.69e+04, 3.77e+04, 6.58e+04,
01832    2.75e+04, 1.56e+04, 3.18e+04, 5.73e+03, 1.3e+04, 1.78e+04, 1.44e+04,
01833    1.88e+04, 6.22e+04, 1.4e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01834    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01835    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01836    1.64e+03, 8.39e+04, 1.8e+05, 7.15e+04, 1.92e+04, 4.52e+03, 1.47e+03,
01837    1.96e+03, 1.44e+03, 838, 352, 412, 1.12e+03, 1.92e+03, 2.42e+03, 2.07e+03,
01838    1.95e+03, 3.16e+03, 6.86e+03, 2.55e+05, 7.6e+04, 0, 0, 0, 0, 0, 0, 0},
01839   {448, 931, 6.57e+03, 3.98e+03, 7.95e+03, 1.05e+03, 6.4e+03, 2.4e+04,
01840    5.06e+04, 1.49e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01841    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01842    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26.2, 3.28e+03, 5.41e+03,
01843    7.8e+03, 3.52e+04, 2.97e+04, 3.24e+04, 3.85e+04, 4.47e+04, 1.56e+04,
01844    1.46e+04, 1.55e+04, 3.63e+03, 1.49e+04, 2.51e+04, 1.1e+04, 2.1e+04,
01845    1.99e+04, 2.58e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01846    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01847    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 409,
01848    6.32e+04, 2.59e+05, 2.64e+04, 9.82e+03, 1.42e+03, 1.51e+03, 1.07e+03, 665,
01849    501, 1.15e+03, 1.46e+03, 1.98e+03, 2.69e+03, 2.13e+03, 1.76e+03, 2.97e+03,
01850    7.34e+03, 2.31e+05, 1.31e+05, 0, 0, 0, 0, 0, 0},
01851   {448, 1.01e+03, 6.37e+03, 3.66e+03, 6.83e+03, 1.63e+03, 7.04e+03, 1.01e+04,
01852    4.8e+04, 3.01e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01853    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01854    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 37.5, 766, 2.96e+03,
01855    3.57e+03, 6.68e+03, 8.51e+03, 5.1e+03, 7.64e+03, 6.09e+03, 6.44e+03,
01856    1.37e+04, 2.53e+04, 1.76e+04, 9.45e+03, 1.99e+04, 397, 0, 0, 0, 0, 0, 0, 0,
01857    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01858    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01859    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.11, 2.4e+05, 1.09e+05, 9.57e+03, 1.83e+03,
01860    1.2e+03, 870, 507, 944, 2.29e+03, 1.61e+03, 2.13e+03, 2.91e+03, 2.21e+03,
01861    1.54e+03, 3.09e+03, 9.22e+03, 1.33e+05, 1.06e+05, 776, 0, 0, 0, 0, 0, 0},
01862   {443, 1.09e+03, 6.16e+03, 3.39e+03, 5.79e+03, 2.56e+03, 6.46e+03, 5.4e+03,
01863    4.36e+04, 5.25e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01864    0, 0, 0, 0, 0, 0, 0, 0, 4.48e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01865    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01866    0, 168, 4.87e+03, 1.07e+04, 5.13e+03, 1.63e+04, 1.4e+04, 2.27e+04,
01867    1.73e+04, 2e+03, 1.09e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
01868    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01869    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01870    0, 0, 0, 9.27e+04, 2.59e+05, 4.97e+03, 1.6e+03, 1.07e+03, 598, 297,
01871    1.58e+03, 3.28e+03, 2.03e+03, 2.5e+03, 3.15e+03, 2.39e+03, 1.47e+03,
01872    3.62e+03, 1.25e+04, 1.07e+05, 7.21e+04, 531, 0, 0, 0, 0, 0, 0},
01873   {435, 1.17e+03, 5.93e+03, 3.14e+03, 4.65e+03, 3.21e+03, 4.96e+03, 2.46e+03,
01874    3.87e+04, 7.76e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01875    0, 0, 0, 0, 0, 0, 0, 0, 8.77e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01876    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01877    0, 0, 0.00332, 2.88e+03, 1e+04, 3.58e+04, 1.42e+04, 3.8e+04, 2.15e+04,
01878    2.62e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01879    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01880    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01881    2.25e+05, 3.39e+05, 865, 710, 2.86e+03, 248, 485, 1.39e+03, 3.92e+03,
01882    2.47e+03, 2.73e+03, 3.52e+03, 2.67e+03, 1.28e+03, 3.66e+03, 1.95e+04,
01883    8.23e+04, 4.9e+04, 1.37e+04, 0, 0, 0, 0, 0, 0},
01884   {434, 1.24e+03, 5.67e+03, 2.92e+03, 3.45e+03, 3.33e+03, 3.82e+03, 2.3e+03,
01885    4.04e+04, 8.76e+04, 0, 1.26e+04, 9.93e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01886    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.5e+04, 2e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01887    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01888    0, 0, 0, 0, 0, 0, 0, 0, 0, 7.48e+03, 4.95e+04, 3.3e+04, 2.79e+04, 2.82e+04,
01889    3.37e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01890    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01891    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01892    2.06e+05, 3.57e+05, 1.42e+04, 3.97e+03, 6.69e+03, 365, 795, 699, 4.9e+03,
01893    2.3e+03, 3.06e+03, 3.68e+03, 2.8e+03, 1.34e+03, 3.3e+03, 3.98e+04,
01894    2.79e+04, 6.9e+04, 2.92e+04, 0, 0, 0, 0, 0, 0},
01895   {429, 1.32e+03, 5.35e+03, 2.78e+03, 2.51e+03, 3.1e+03, 3.39e+03, 2.93e+03,
01896    4.19e+04, 8.31e+04, 0, 3.71e+04, 3.39e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01897    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01898    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01899    0, 0, 0, 0, 0, 0, 0, 1.56e+03, 8.66e+03, 3.11e+03, 658, 16.2, 0, 0, 0, 0,
01900    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01901    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01902    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.49e+04, 3.55e+05,
01903    1.12e+05, 1.39e+04, 9.87e+03, 572, 1.02e+03, 638, 5.41e+03, 2.6e+03,
01904    4.85e+03, 3.57e+03, 2.96e+03, 1.42e+03, 2.75e+03, 4.85e+04, 2.61e+04,
01905    7.46e+04, 3.19e+04, 0, 0, 0, 0, 0, 0},
01906   {419, 1.39e+03, 5.02e+03, 2.71e+03, 1.71e+03, 2.4e+03, 3.6e+03, 3.43e+03,
01907    4.29e+04, 8.03e+04, 0, 6.04e+04, 3.61e+04, 3.16e+03, 0, 0, 0, 0, 0, 0, 0, 0,
01908    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01909    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01910    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01911    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01912    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01913    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.89e+05, 1.93e+05, 1.36e+05, 1.35e+04,
01914    1.07e+03, 1.43e+03, 1.27e+03, 5.63e+03, 2.5e+03, 6.41e+03, 3.65e+03,
01915    3.55e+03, 1.39e+03, 1.91e+03, 8.69e+04, 1.9e+04, 6.42e+04, 5.44e+04, 0, 0,
01916    0, 0, 0, 0},
01917   {414, 1.47e+03, 4.66e+03, 2.72e+03, 1.04e+03, 2.15e+03, 4.45e+03, 3.78e+03,
01918    4.19e+04, 6.44e+04, 0.0322, 6.54e+04, 1.36e+04, 2.4e+04, 0, 0, 0, 0, 0, 0,
01919    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01920    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01921    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01922    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01923    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01924    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 709, 2.78e+05, 4.94e+05, 7.59e+03,
01925    1.74e+03, 2.39e+03, 2.05e+03, 5.62e+03, 1.64e+03, 6.81e+03, 4.79e+03,
01926    4.67e+03, 1.54e+03, 1.09e+03, 1.22e+05, 4.77e+04, 1.01e+05, 6.72e+04, 0, 0,
01927    0, 0, 0, 0},
01928   {406, 1.53e+03, 4.29e+03, 2.81e+03, 687, 2.2e+03, 5.68e+03, 3.77e+03,
01929    3.79e+04, 3.21e+04, 326, 5.49e+04, 9.98e+03, 4.91e+04, 0, 0, 0, 0, 0, 0, 0,
01930    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01931    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01932    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01933    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01934    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01935    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.63e+05, 2.7e+05, 8.99e+03, 4.11e+03,
01936    9.14e+03, 3.52e+03, 3.93e+03, 1.44e+03, 7.38e+03, 6.36e+03, 5.53e+03,
01937    1.96e+03, 641, 1.17e+05, 8.73e+04, 7.28e+04, 4.72e+04, 0, 0, 0, 0, 0, 0},
01938   {400, 1.59e+03, 3.88e+03, 2.97e+03, 546, 2.11e+03, 6.87e+03, 3.51e+03,
01939    3.16e+04, 1.49e+04, 2.43e+03, 3.36e+04, 1.19e+04, 9.85e+04, 0, 0, 0, 0, 0,
01940    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01941    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01942    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01943    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01944    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.29e+03, 221, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01945    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.55e+05, 2.4e+05,
01946    6.85e+04, 3.3e+04, 2.59e+04, 8.24e+03, 1.04e+04, 2.43e+03, 7.69e+03,
01947    8.23e+03, 5.46e+03, 2.46e+03, 979, 1.15e+05, 9.16e+04, 1.16e+05, 3.3e+04,
01948    0, 0, 0, 0, 0, 0},
01949   {388, 1.65e+03, 3.45e+03, 3.16e+03, 478, 1.91e+03, 7.86e+03, 3.24e+03,
01950    2.37e+04, 2.6e+04, 3.23e+04, 1.22e+04, 3.56e+04, 1.6e+05, 0, 0, 0, 0, 0, 0,
01951    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01952    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01953    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01954    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
01955    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01956    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.86e+04, 3.67e+05, 3.05e+05,
01957    1.35e+05, 7.06e+04, 1.56e+04, 9.55e+03, 8.97e+03, 7.11e+03, 1.07e+04,
01958    8.79e+03, 3.18e+03, 1.58e+03, 1.12e+05, 9.59e+04, 9.5e+04, 1.66e+04, 0, 0,
01959    0, 0, 0, 0},
01960   {381, 1.7e+03, 2.97e+03, 3.28e+03, 461, 1.65e+03, 8.65e+03, 2.58e+03,
01961    1.76e+04, 3.36e+04, 5.77e+04, 7.69e+03, 5.95e+04, 1.56e+05, 273, 0, 0, 0,
01962    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01963    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01964    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01965    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01966    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 27.4, 2.17e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01967    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.59e+04, 3.4e+05,
01968    5.96e+05, 2.23e+05, 2.41e+05, 1.51e+04, 9.01e+03, 1.33e+04, 3.32e+03,
01969    1.48e+04, 1.29e+04, 3.66e+03, 2.45e+03, 1.04e+05, 1.06e+05, 7.18e+04,
01970    3.96e+03, 0, 0, 0, 0, 0, 0},
01971   {372, 1.74e+03, 2.5e+03, 3.3e+03, 485, 1.26e+03, 9.03e+03, 1.73e+03,
01972    1.26e+04, 6.54e+04, 9.51e+04, 1.33e+04, 8.24e+04, 9.75e+04, 6.71e+03, 0, 0,
01973    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01974    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01975    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01976    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01977    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.32e+03, 1.28e+04, 0, 0, 0, 0, 0, 0, 0,
01978    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 691,
01979    5.82e+05, 6.02e+05, 3.87e+05, 4.75e+05, 1.2e+05, 3.39e+04, 1.31e+04,
01980    2.27e+03, 2.23e+04, 1.95e+04, 3.83e+03, 6.32e+03, 6.06e+04, 9.22e+04,
01981    6.5e+04, 1.53e+03, 0, 0, 0, 0, 0, 0},
01982   {362, 1.78e+03, 2.07e+03, 3.22e+03, 566, 932, 9.01e+03, 1.16e+03, 1.26e+04,
01983    4.95e+04, 8.86e+04, 1.66e+04, 9.67e+04, 4.33e+04, 3.8e+04, 0, 0, 0, 0, 0,
01984    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 730, 0, 97.9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01985    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01986    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01987    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01988    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.33e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01989    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.56e+05, 4.29e+05,
01990    4.77e+05, 3.46e+05, 4.03e+05, 1.01e+05, 9.87e+03, 3.91e+03, 2.18e+04,
01991    2.49e+04, 3.08e+03, 1.66e+04, 4.56e+04, 6.55e+04, 5.69e+04, 300, 0, 0, 0,
01992    0, 0, 0},
01993   {355, 1.81e+03, 1.7e+03, 3.09e+03, 677, 788, 9.06e+03, 885, 1.59e+04,
01994    1.74e+04, 8.63e+04, 2.12e+04, 1.05e+05, 1.61e+04, 9.83e+04, 0, 0, 0, 0, 0,
01995    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.55e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01996    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01997    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01998    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
01999    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.98e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02000    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.72e+05, 4.09e+05,
02001    4.51e+05, 4.64e+05, 4.45e+05, 1.48e+05, 3.91e+03, 1.23e+04, 4.1e+04,
02002    1.6e+04, 1.96e+03, 6.98e+04, 2.55e+04, 3.74e+04, 5.29e+04, 0, 0, 0, 0, 0,
02003    0, 0},
02004   {344, 1.83e+03, 1.41e+03, 2.94e+03, 755, 727, 9.46e+03, 808, 1.84e+04,
02005    1.8e+04, 9.64e+04, 1.91e+04, 9.68e+04, 9.85e+03, 1.65e+05, 0, 0, 0, 0, 0,
02006    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02007    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02008    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02009    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.66e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02010    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 254, 8.36e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02011    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.134, 4.65, 0, 0, 4.81e+04,
02012    3.37e+05, 1.16e+04, 3.87e+05, 2.62e+05, 1.59e+05, 2.67e+04, 5.66e+04,
02013    3.22e+04, 5.04e+03, 3.81e+03, 9.24e+04, 4.34e+04, 3.33e+04, 3.46e+04, 0, 0,
02014    0, 0, 0, 0, 0, 0},
02015   {340, 1.84e+03, 1.2e+03, 2.85e+03, 696, 702, 9.86e+03, 908, 1.92e+04,
02016    1.48e+04, 6.53e+04, 1.61e+04, 5.5e+04, 5.46e+03, 2.02e+05, 0, 0, 0, 0, 0,
02017    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02018    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02019    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02020    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.59e+04, 0, 1.79e+03, 0, 0, 0, 0, 0, 0, 0,
02021    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02022    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.11e+04, 2.79e+04, 0, 0,
02023    1.1e+03, 2.57e+05, 2.87e+04, 8.51e+04, 1.47e+05, 2.07e+05, 5.17e+04, 1e+05,
02024    2.62e+04, 1.19e+04, 1.4e+04, 2.32e+04, 4.7e+04, 3.52e+04, 3.48e+04, 0, 0,
02025    0, 0, 0, 0, 0, 0},
02026   {331, 1.85e+03, 1.05e+03, 2.81e+03, 300, 743, 1.05e+04, 1.25e+03, 1.66e+04,
02027    1.85e+04, 6.06e+04, 2.43e+04, 2.92e+04, 4.87e+03, 2.19e+05, 0, 0, 0, 0, 0,
02028    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02029    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02030    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02031    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 323, 1.55e+03, 36.5, 0, 0, 0, 0, 0, 0, 0,
02032    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02033    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 52.5, 1.84e+04, 5.2e+04, 0, 0, 0,
02034    8.18e+04, 1.59e+04, 4.75e+04, 9.66e+04, 1.09e+05, 5.86e+04, 1.7e+05,
02035    2.59e+04, 3.1e+04, 4.43e+04, 1.92e+04, 5.86e+04, 3.88e+04, 3.43e+04, 0, 0,
02036    0, 0, 0, 0, 0, 0},
02037   {321, 1.83e+03, 943, 2.75e+03, 106, 1.15e+03, 1.08e+04, 1.77e+03, 1.13e+04,
02038    1.44e+04, 5.27e+04, 3.59e+04, 8.4e+03, 4.52e+03, 2.45e+05, 0, 0, 0, 0, 0,
02039    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02040    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02041    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
02042    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02043    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02044    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.9e+04, 3.94e+04, 4.44e+04, 0, 0, 0, 3.24e+03,
02045    5.84e+04, 6.93e+03, 3.37e+04, 1.4e+05, 1.49e+05, 1.62e+05, 1.73e+04,
02046    3.44e+04, 1.09e+04, 3.8e+04, 1.26e+05, 3.21e+04, 2.47e+04, 0, 0, 0, 0, 0,
02047    0, 0},
02048   {311, 1.81e+03, 881, 2.63e+03, 142, 1.66e+03, 1.07e+04, 2.3e+03, 6.94e+03,
02049    7.09e+03, 2.61e+04, 4.43e+04, 1.44e+03, 3.49e+03, 2.52e+05, 0.00444, 0, 0,
02050    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02051    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02052    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02053    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02054    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02055    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.02e+04, 5.14e+04, 1.12e+04, 0, 0, 0, 0,
02056    0, 0, 534, 1.36e+05, 1.36e+05, 1.43e+05, 4.36e+04, 1.39e+04, 6.3e+04,
02057    6.21e+04, 1.05e+05, 5.63e+03, 5.82e+03, 0, 0, 0, 0, 0, 0, 0},
02058   {304, 1.78e+03, 839, 2.47e+03, 184, 1.99e+03, 9.81e+03, 2.62e+03, 4.29e+03,
02059    6.33e+03, 1.62e+04, 4.71e+04, 1.33e+03, 2.18e+03, 2.06e+05, 0.113, 0, 0, 0,
02060    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02061    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02062    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02063    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02064    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02065    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.45e+04, 6.97e+04, 3.91e+04, 0, 0, 0, 0, 0,
02066    0, 0.36, 2.67e+04, 3.39e+04, 5.84e+04, 2.26e+04, 8.12e+03, 6.41e+04,
02067    1.64e+05, 4.84e+04, 448, 69, 0, 0, 0, 0, 0, 0, 0},
02068   {294, 1.73e+03, 804, 2.27e+03, 171, 1.92e+03, 8.18e+03, 2.75e+03, 2.5e+03,
02069    4.27e+03, 2.52e+04, 4.1e+04, 2.56e+03, 2.51e+03, 1.8e+05, 2.02e+03, 0, 0,
02070    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02071    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02072    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02073    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02074    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02075    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 270, 7.35e+04, 8.31e+04, 1.58e+05, 0, 0, 0,
02076    0, 0, 0, 0, 3.09e+03, 4.75e+03, 5.5e+03, 2.29e+04, 4.24e+03, 1.22e+03,
02077    9.87e+04, 9e+04, 347, 0, 0, 0, 0, 0, 0, 0, 0},
02078   {286, 1.69e+03, 765, 2.07e+03, 101, 1.77e+03, 6.6e+03, 2.58e+03, 1.94e+03,
02079    1.83e+03, 2.53e+04, 1.95e+04, 5.13e+03, 3.68e+03, 1.62e+05, 3.32e+04, 2.27,
02080    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02081    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02082    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02083    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02084    0, 1.06e+04, 5.27e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02085    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.511, 2.07e+05, 6.74e+04,
02086    1.13e+05, 2.66, 0, 0, 0, 0, 0, 0, 0, 811, 2.15, 2.11e+03, 529, 990,
02087    5.16e+04, 9.22e+04, 281, 0, 0, 0, 0, 0, 0, 0, 0},
02088   {277, 1.62e+03, 725, 1.9e+03, 109, 1.66e+03, 5.91e+03, 2.63e+03, 1.88e+03,
02089    587, 1.59e+04, 6.43e+03, 6.71e+03, 2.98e+03, 9.41e+04, 3.25e+04, 2.08e+03,
02090    1.71, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02091    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02092    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02093    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.56, 55.3, 35.8, 550, 67.6, 0, 0, 0, 0,
02094    1.18e+03, 1.9e+03, 0, 0, 0, 0, 0, 2e+05, 0, 0, 0, 0, 4.01e+04, 0, 0, 0, 0,
02095    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02096    0, 4e+05, 1.63e+05, 4.17e+04, 1.02, 0, 0, 0, 0, 0, 0, 0, 15.3, 0, 0, 0, 0,
02097    5.81e+04, 6.77e+04, 4.68e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02098   {267, 1.56e+03, 684, 1.75e+03, 183, 1.59e+03, 5.92e+03, 2.47e+03, 2.55e+03,
02099    305, 7.15e+03, 2.95e+03, 6.18e+03, 967, 5.61e+04, 4.78e+04, 3.65e+04,
02100    2.52e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02101    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02102    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02103    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.84, 177, 237, 158, 132, 190, 230, 57.1,
02104    117, 698, 7.4e+03, 5.22e+03, 6.96e+03, 997, 0, 0, 0, 4.17e+04, 4.53e+03, 0,
02105    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02106    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.18e+05, 7.4e+04, 4.57e+04, 10.7, 0, 0, 0, 0, 0,
02107    0, 0, 0, 0, 0, 0, 0, 0, 4.29e+03, 3.13e+04, 2.42e+03, 0, 0, 0, 0, 0, 0, 0, 0},
02108   {257, 1.48e+03, 650, 1.54e+03, 219, 1.53e+03, 6.06e+03, 2.23e+03, 2.73e+03,
02109    441, 3.9e+03, 2.73e+03, 4.73e+03, 486, 4.45e+04, 3.41e+04, 9.06e+04,
02110    1.11e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02111    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02112    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02113    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 25.9, 492, 380, 191, 170, 128, 175, 139, 133,
02114    429, 2.37e+03, 1.87e+03, 2.1e+03, 2.96e+03, 5.42e+03, 3.58e+03, 524, 0,
02115    601, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02116    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.2e+04, 6.36e+04, 6.18e+04, 0, 0, 0,
02117    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.51e+04, 7.19e+03, 0, 0, 0, 0, 0, 0, 0,
02118    0},
02119   {250, 1.39e+03, 618, 1.19e+03, 204, 1.48e+03, 6.01e+03, 1.9e+03, 2.07e+03,
02120    1.29e+03, 2.42e+03, 3.11e+03, 3.39e+03, 1.3e+03, 4.39e+04, 2.34e+04,
02121    3.41e+04, 7.42e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02122    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02123    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 535, 0, 0,
02124    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.86e+03, 9.96e+03, 4.5e+03,
02125    465, 166, 234, 424, 288, 259, 1.77e+03, 1.18e+03, 8.11e+03, 1.16e+03, 947,
02126    1.95e+03, 3.26e+03, 2.28e+03, 771, 4.13e+03, 5.32e+03, 0, 0, 0, 0, 0, 0, 0,
02127    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02128    0, 0, 0, 7.66e+04, 7.96e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
02129    3.15e+04, 2.57e+04, 0, 0, 0, 0, 0, 0, 0, 0},
02130   {242, 1.3e+03, 586, 851, 170, 1.55e+03, 6.01e+03, 1.78e+03, 1.4e+03,
02131    1.94e+03, 1.37e+03, 2.71e+03, 3.72e+03, 1.94e+03, 4.8e+03, 1.03e+05,
02132    1.59e+04, 5.47e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02133    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02134    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02135    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0277, 2.75e+03, 1e+03, 4.2e+04,
02136    5.33e+04, 4.67e+04, 1.25e+03, 418, 621, 459, 202, 309, 560, 1.02e+04,
02137    1.17e+04, 993, 3.25e+03, 1.32e+03, 3.48e+03, 6.29e+03, 2.87e+03, 1.2e+03,
02138    31.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02139    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02140    0, 0, 0, 1.92e+04, 2.43e+04, 0, 0, 0, 0, 0, 0, 0, 0},
02141   {231, 1.22e+03, 555, 583, 102, 1.71e+03, 5.77e+03, 1.87e+03, 966, 1.88e+03,
02142    486, 855, 3.85e+03, 2.99e+03, 1.56e+03, 1.89e+05, 9.14e+03, 4.84e+04, 22.1,
02143    0.0644, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02144    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.52e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02145    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02146    0, 0, 0, 0, 0, 0, 0, 0, 617, 4.57e+03, 2.52e+04, 4.83e+04, 1.64e+04, 5.7e+04,
02147    1.17e+03, 3.39e+03, 744, 7.71e+03, 1.62e+04, 1.22e+04, 8.08e+03, 1.09e+03,
02148    9.43e+03, 6.17e+03, 3.86e+03, 1.9e+03, 7.62e+03, 1.26e+04, 4.12e+03,
02149    1.31e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02150    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02151    0, 0, 0, 0, 0, 8.64e+03, 4.05e+03, 0, 0, 0, 0, 0, 0, 0, 0},
02152   {223, 1.13e+03, 530, 394, 72, 1.8e+03, 5.07e+03, 2.23e+03, 699, 1.5e+03,
02153    1.2e+03, 710, 3.53e+03, 3.39e+03, 427, 1.04e+05, 1.58e+04, 3.46e+04,
02154    2.53e+04, 649, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02155    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02156    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02157    0, 0, 0, 0, 0, 0, 581, 6.17e+03, 6.74e+03, 2.08e+04, 8.38e+03, 2.13e+04,
02158    2.59e+04, 8.8e+03, 8.15e+03, 5e+03, 3.51e+03, 3.58e+03, 1.48e+04, 3.31e+03,
02159    433, 7.25e+03, 8.2e+03, 1.04e+03, 3.95e+03, 1.98e+03, 5.43e+03, 5.21e+03,
02160    1.87e+04, 9.02e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02161    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02162    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02163   {214, 1.04e+03, 514, 282, 83.9, 1.81e+03, 4.3e+03, 2.61e+03, 443, 679,
02164    1.91e+03, 1.19e+03, 2.32e+03, 2.77e+03, 388, 2.73e+04, 5.34e+04, 2.78e+04,
02165    1.43e+04, 4.66e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02166    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 690, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02167    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02168    0, 0, 0, 0, 0, 0, 0, 0, 0, 38.9, 5.17e+03, 8.72e+03, 9.18e+03, 1.14e+04,
02169    3.03e+03, 1.35e+03, 7.08e+03, 3.37e+03, 4.77e+03, 1.62e+03, 1.65e+03,
02170    1.21e+03, 1.1e+03, 1.93e+03, 730, 154, 422, 1.77e+03, 1.43e+03, 492,
02171    1.28e+03, 5.85e+03, 1.34e+04, 7.88e+04, 2.34e+04, 5.84e+04, 3.14e+03, 0, 0,
02172    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.999, 1.26e+04, 509,
02173    8.02e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02174    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02175   {204, 954, 498, 225, 86.6, 1.72e+03, 3.51e+03, 2.83e+03, 297, 244, 2.58e+03,
02176    932, 1.27e+03, 2.61e+03, 760, 1.41e+03, 5.46e+04, 1.37e+04, 9.26e+04,
02177    1.11e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02178    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02179    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02180    0, 0, 0, 5.87e+03, 6.03e+03, 8.62e+03, 1.24e+04, 1.59e+04, 3.88e+03, 793,
02181    4.53e+03, 4.25e+03, 1.9e+03, 673, 569, 18.9, 1e+03, 516, 258, 217, 434,
02182    137, 437, 3.99e+03, 598, 414, 6.28e+03, 1.17e+05, 3.07e+05, 3.65e+05,
02183    2.17e+04, 4.03e+03, 0, 0, 0, 6.03e+03, 3.75e+03, 5.09e+03, 1.16e+04,
02184    2.12e+03, 1.8e+04, 2.37e+04, 0, 0, 0, 0, 0, 0, 0, 0, 9.2e+03, 3.49e+03,
02185    3.71e+03, 1.65e+03, 0, 0, 4.69, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2e+03,
02186    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02187   {195, 876, 490, 149, 72.9, 1.49e+03, 2.57e+03, 3.14e+03, 228, 226, 2.59e+03,
02188    746, 953, 2.9e+03, 1.73e+03, 709, 6.25e+04, 6.83e+03, 2.1e+05, 435, 0, 0,
02189    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02190    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02191    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 181,
02192    1.07e+04, 8.52e+03, 9.52e+03, 1.77e+04, 9.99e+03, 914, 609, 736, 3.25e+03,
02193    1.97e+03, 313, 901, 172, 1.39e+03, 159, 97.9, 51.7, 136, 264, 715, 482,
02194    454, 1.25e+03, 2.68e+03, 1.18e+04, 2.18e+05, 4.86e+05, 1.53e+04, 1.64e+04,
02195    330, 0, 0, 1.69e+04, 4.09e+03, 6.43e+03, 7.63e+04, 6.86e+04, 8.61e+04,
02196    3.43e+04, 1.95e+04, 0, 0, 0, 0, 0, 0, 0, 1.05e+04, 4.32e+03, 5.18e+03,
02197    1.12e+04, 0, 82.1, 0.0277, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.38e+04, 0,
02198    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02199   {187, 797, 479, 82, 55.6, 1.18e+03, 1.75e+03, 3.71e+03, 213, 383, 2.56e+03,
02200    458, 804, 3.54e+03, 1.85e+03, 695, 6.09e+04, 1.45e+04, 1.53e+05, 2.06e+04,
02201    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02202    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02203    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02204    86.9, 5.34e+03, 1.57e+03, 2.9e+03, 9.72e+03, 3.34e+03, 2.42e+03, 1.27e+03,
02205    445, 924, 744, 273, 622, 772, 119, 148, 707, 765, 260, 81.8, 217, 1.74e+03,
02206    2.9e+03, 3.3e+03, 1.33e+03, 8.85e+03, 1.41e+05, 1.8e+05, 4.63e+05,
02207    2.93e+04, 1.98e+04, 0, 0, 1.81e+03, 2.36e+04, 1.17e+04, 4.91e+04, 2.72e+04,
02208    1.24e+05, 9.26e+04, 5.77e+03, 0, 0, 0, 0, 0, 0, 0, 1.66e+03, 4.85e+03,
02209    2.48e+03, 9.91e+03, 0.25, 710, 2.87e+03, 0, 0, 7.69, 6.83e+03, 0, 0, 0, 0,
02210    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02211   {176, 720, 473, 41.3, 40.7, 925, 1.37e+03, 4.38e+03, 299, 969, 1.54e+03,
02212    114, 657, 3.56e+03, 1.18e+03, 1.8e+03, 9.8e+04, 1.38e+04, 1.07e+05,
02213    8.34e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02214    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02215    0, 0, 0, 0, 0, 0, 0, 286, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
02216    0, 0, 0, 0, 3.3e+03, 1.73e+03, 2.35e+03, 1.82e+03, 2.63e+03, 1.92e+03,
02217    3.83e+03, 2.47e+03, 405, 56, 365, 199, 49, 287, 669, 541, 376, 365, 290,
02218    64, 1.1e+03, 2.84e+03, 2.93e+03, 653, 5.84e+03, 6.3e+04, 1.54e+05,
02219    3.97e+05, 1.3e+05, 8.63e+04, 3.3e+04, 1.31e+04, 1.4e+04, 3.91e+04,
02220    1.54e+04, 1.31e+05, 1.13e+04, 2.73e+04, 1.86e+05, 6.09e+03, 0, 0, 0, 0, 0,
02221    195, 0, 10.7, 1.42e+04, 5.7e+03, 7.6e+03, 5.64e+03, 1.82e+04, 8.36e+03,
02222    1.96, 0, 0, 104, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02223    0, 0, 0, 0, 0, 0, 0, 0, 0},
02224   {170, 652, 473, 27.7, 29.6, 764, 1.32e+03, 4.83e+03, 433, 1.89e+03, 879,
02225    70.3, 438, 3.06e+03, 938, 3.47e+03, 8.54e+04, 2.49e+04, 5.82e+04, 2.78e+04,
02226    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02227    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02228    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02229    1.51e+03, 2.22e+03, 2.21e+03, 2.63e+03, 974, 1.52e+03, 7.08e+03, 2.41e+03,
02230    540, 49.3, 89.6, 321, 183, 37.4, 499, 601, 519, 591, 342, 143, 150,
02231    8.42e+03, 6.42e+03, 5.09e+03, 2.45e+03, 1.05e+04, 5.2e+04, 2.91e+05,
02232    1.1e+05, 2.43e+05, 1.97e+05, 8.37e+03, 7.61e+04, 4.45e+04, 1.51e+04,
02233    1.7e+05, 4.81e+04, 3.44e+03, 2.09e+05, 321, 0, 0, 0, 171, 106, 1.39e+03,
02234    3.03e+03, 2.64e+03, 107, 1.21e+03, 7.6e+03, 2.75e+04, 4.37e+04, 3.32e+04,
02235    1.59e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02236    0, 0, 0, 0, 0, 0, 0, 0, 0},
02237   {161, 582, 480, 29.9, 22.5, 691, 1.33e+03, 4.93e+03, 572, 2.68e+03, 566,
02238    85.1, 810, 1.72e+03, 1.21e+03, 2.65e+03, 4.79e+04, 1.89e+04, 1.25e+05,
02239    1.46e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02240    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02241    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02242    0, 0, 0, 1.25e+03, 1.26e+03, 1.7e+03, 2.23e+03, 2.11e+03, 2.18e+03,
02243    1.51e+03, 1.45e+03, 2.77e+03, 5.21e+03, 4.8e+03, 626, 1.32e+03, 546, 126,
02244    69.8, 301, 897, 898, 159, 25.9, 5.67e+03, 5.95e+03, 6.67e+03, 3.69e+03,
02245    5.37e+03, 9.15e+03, 5.64e+04, 1.42e+05, 3.08e+05, 2.58e+05, 0, 2.02e+05,
02246    8.15e+04, 1.26e+04, 1.65e+04, 7.83e+04, 6.06e+03, 1.47e+05, 3.61, 0, 0, 0,
02247    5.6e+03, 2.35e+03, 1.49e+03, 1.37e+04, 2.07e+04, 2.89e+04, 564, 2.21e+04,
02248    1.74e+04, 3.71e+04, 1.47e+04, 955, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02249    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02250   {151, 518, 491, 37, 17.4, 621, 1.35e+03, 4.8e+03, 728, 3.29e+03, 361, 257,
02251    727, 1.07e+03, 2.16e+03, 1.55e+03, 1.44e+04, 8.78e+04, 9.79e+04, 1.23e+04,
02252    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02253    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02254    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02255    1.47e+03, 2.67e+03, 1.99e+03, 5.88e+03, 5.89e+03, 2.22e+03, 1.09e+03, 738,
02256    702, 1.78e+03, 1.83e+04, 172, 272, 202, 90.4, 84, 152, 1.87e+03, 753,
02257    1.16e+03, 34.2, 2.53e+03, 2.37e+03, 1.49e+03, 5.8e+03, 3.44e+03, 1.4e+04,
02258    2.54e+04, 2.38e+04, 1.29e+05, 6.15e+04, 0, 5.24e+05, 1.21e+05, 1.66e+04,
02259    9.83e+03, 4.09e+04, 6.89e+04, 8.06e+04, 0, 0, 0, 10.8, 6.22e+03, 2.83e+03,
02260    0, 4.6e+03, 1.52e+04, 1.2e+04, 1.34e+04, 2.29e+04, 2.08e+04, 3.79e+04,
02261    4.73e+03, 638, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02262    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02263   {142, 464, 505, 43.7, 12.8, 547, 1.39e+03, 4.45e+03, 945, 3.43e+03, 222,
02264    672, 1.03e+03, 1.08e+03, 3.16e+03, 743, 1.42e+04, 1.81e+05, 4.06e+04,
02265    2.71e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02266    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02267    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02268    0, 0, 0, 1.98e+03, 3.47e+03, 3.75e+03, 6.87e+03, 4.2e+03, 2.76e+03,
02269    1.34e+03, 342, 1.65e+03, 960, 2.13e+03, 480, 1.01e+03, 63.5, 90.1, 211,
02270    240, 149, 520, 858, 267, 866, 1.33e+03, 1.23e+03, 4.79e+03, 4.37e+03,
02271    5.29e+03, 5.92e+04, 1.15e+04, 8.84e+03, 1.61e+05, 0, 1.04e+05, 1.19e+05,
02272    2.62e+04, 5.94e+04, 5.52e+04, 1.24e+05, 9.02e+04, 0.04, 0, 1.14, 471,
02273    1.16e+03, 621, 718, 3.3e+03, 1.74e+03, 9.19e+03, 1.89e+04, 2.33e+04,
02274    5.66e+03, 2.33e+03, 1.1e+03, 0, 13.5, 8.02, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02275    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02276   {133, 414, 518, 42, 18.1, 487, 1.36e+03, 4.07e+03, 1.21e+03, 2.8e+03, 285,
02277    766, 755, 1.53e+03, 2.35e+03, 345, 1.84e+04, 1.05e+05, 1.54e+05, 4.57e+03,
02278    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02279    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02280    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02281    2.27e+03, 8.21e+03, 1.06e+04, 918, 563, 841, 2.75e+03, 1.07e+03, 790, 471,
02282    475, 668, 884, 3.12e+03, 7.06e+03, 3.71e+03, 676, 19.5, 62.1, 242, 243,
02283    132, 469, 757, 2.25e+03, 3.99e+03, 2.38e+03, 4.26e+04, 2.32e+04, 4.8e+03,
02284    1.54e+05, 5.07e+03, 0, 4.01e+04, 6.06e+04, 1.4e+05, 1.11e+05, 3.91e+04,
02285    2.36e+05, 1.27e+03, 505, 375, 2.58e+03, 1.87e+03, 6.48e+03, 146, 2.55e+03,
02286    1.19e+03, 1.25e+03, 5.06e+03, 1.65e+03, 0, 0, 0, 0, 252, 125, 0, 0, 0, 0,
02287    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02288   {124, 373, 528, 35.4, 24.9, 441, 1.18e+03, 3.53e+03, 1.04e+03, 1.77e+03,
02289    427, 954, 824, 1.81e+03, 2.37e+03, 153, 2.4e+04, 5.61e+04, 2.19e+05,
02290    1.33e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02291    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02292    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02293    0, 0, 0, 5.79, 1.31e+04, 8.98e+03, 3.93e+03, 1.23e+03, 1.11e+03, 2.94e+03,
02294    964, 283, 328, 582, 637, 1.13e+03, 1.24e+03, 296, 1.22e+03, 705, 490, 164,
02295    95.3, 140, 1.09e+03, 1.3e+03, 840, 2.19e+03, 2.67e+03, 1.99e+03, 9.71e+03,
02296    3.09e+04, 6.38e+03, 8.04e+04, 2.3e+04, 0, 69.5, 5.41e+04, 1.4e+05,
02297    1.08e+05, 9.71e+04, 5.66e+05, 2.14e+03, 1.39e+03, 1.93e+03, 2.14e+03,
02298    3.36e+03, 7.96e+03, 834, 2.28e+03, 764, 584, 413, 0, 0, 0, 0, 0, 0, 0, 0,
02299    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02300    0, 0, 0},
02301   {117, 340, 517, 33.3, 27.6, 385, 991, 3.05e+03, 366, 1.88e+03, 332,
02302    1.12e+03, 1.38e+03, 1.72e+03, 2.61e+03, 370, 1.57e+04, 1.93e+04, 2.74e+05,
```

```
02303    3.53e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02304    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02305    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02306    0, 0, 0, 0, 1.05e+03, 2.59e+04, 2.86e+04, 1.03e+04, 1.35e+04, 5.93e+03,
02307    1.61e+03, 678, 2.3e+03, 994, 506, 944, 1.2e+03, 3.53e+03, 5.22e+03, 363,
02308    539, 1.21e+03, 3.22e+03, 4.65e+03, 4.06e+03, 5.03e+03, 2.41e+03, 1.34e+03,
02309    3.55e+03, 1.4e+03, 5.1e+03, 3.28e+04, 2.45e+04, 3.96e+04, 5.83e+04,
02310    1.84e+03, 0, 518, 0, 5.8e+04, 1.89e+05, 5.93e+05, 2.93e+03, 5.06e+03,
02311    8.19e+03, 795, 1.5e+03, 3.41e+03, 4.76e+03, 1.32e+03, 685, 507, 0, 0, 0, 0,
02312    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02313    0, 0, 0, 0, 0, 0, 0, 0, 0}},
02314   {108, 310, 464, 34.5, 28.6, 341, 836, 2.42e+03, 318, 2.99e+03, 473, 707,
02315    2.91e+03, 1.44e+03, 1.67e+03, 972, 4.97e+03, 8.52e+03, 2.56e+05, 8.49e+04,
02316    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02317    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02318    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02319    0, 587, 1.72e+03, 2.77e+03, 2.16e+03, 3.49e+03, 3.79e+03, 759, 885, 568,
02320    859, 859, 1.17e+03, 642, 2.41e+03, 3.47e+03, 5.93e+03, 773, 3.37e+03,
02321    1.91e+03, 1.32e+04, 3.99e+03, 5.74e+03, 6.73e+03, 4.57e+03, 1.86e+03,
02322    2.67e+03, 3.81e+03, 1.25e+04, 2.79e+04, 2.23e+04, 8.06e+04, 1.31e+04, 0,
02323    281, 412, 1.19e+04, 3e+05, 4.56e+05, 2.31e+04, 2.78e+04, 2.51e+04,
02324    3.38e+03, 406, 1.91e+03, 3.1e+03, 2.52e+03, 2.73, 0.561, 0, 0, 0, 0, 0, 0,
02325    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02326    0, 0, 0, 0, 0, 0}},
02327   {102, 293, 381, 25.6, 29.3, 313, 689, 1.76e+03, 325, 3.71e+03, 435, 745,
02328    4.17e+03, 1.53e+03, 624, 1.45e+03, 908, 6.82e+03, 2.62e+05, 1.69e+05, 0, 0,
02329    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 116, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02330    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02331    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02332    93.3, 1.41e+03, 4.37e+03, 3.35e+03, 2.74e+03, 745, 1.85e+03, 505, 590, 745,
02333    829, 2.21e+03, 325, 2.89e+03, 8.07e+03, 1.6e+03, 3.7e+03, 2.92e+03,
02334    7.36e+03, 1.16e+04, 6.9e+03, 1.25e+03, 9.62e+03, 4.87e+03, 1.82e+03,
02335    4.08e+03, 3.43e+03, 4.33e+03, 2.93e+04, 3.53e+04, 3.69e+04, 1.95e+04, 0,
02336    377, 1.46e+04, 0, 4.99e+04, 4.28e+04, 5.91e+04, 5.95e+04, 6.55e+04,
02337    2.48e+03, 3.37e+03, 1.66e+03, 1.16e+03, 262, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02338    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02339    0, 0, 0, 0}},
02340   {94.7, 276, 286, 17.9, 30.4, 303, 584, 1.11e+03, 509, 3.46e+03, 670, 537,
02341    3.65e+03, 1.41e+03, 296, 1.8e+03, 536, 4.25e+03, 3.02e+05, 1.89e+05, 0, 0,
02342    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02343    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02344    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02345    40.6, 3.74e+03, 3.73e+03, 4.34e+03, 3.01e+03, 2.91e+03, 949, 570, 551, 545,
02346    452, 771, 39.4, 212, 1.33e+04, 4.18e+03, 977, 7.62e+03, 1.22e+04, 3.36e+04,
02347    4.73e+04, 2.11e+03, 2.93e+03, 4.22e+03, 935, 2.06e+03, 6.56e+03, 1.18e+04,
02348    2.41e+04, 5.44e+04, 5.48e+04, 8.52e+04, 0, 0, 505, 0, 0, 5.59, 1.12e+05,
02349    7.21e+04, 5.88e+04, 1.26e+04, 4.6e+03, 2.24e+03, 4.35e+03, 1.67e+03, 1.11,
02350    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02351    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},
02352   {86.9, 265, 192, 21, 32.3, 306, 549, 857, 644, 2.06e+03, 1.04e+03, 631,
02353    3.09e+03, 1.46e+03, 435, 1.35e+03, 873, 1.49e+03, 4.15e+05, 1.63e+05, 0, 0,
02354    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02355    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02356    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11.6,
02357    354, 1.24e+04, 6.31e+03, 6.51e+03, 6.61e+03, 4.48e+03, 624, 762, 1.73e+03,
02358    2.61e+03, 2.12e+03, 678, 123, 39.3, 563, 1.86e+03, 1.22e+04, 6.58e+04,
02359    2.49e+04, 3.33e+04, 6.12e+04, 1e+04, 2.18e+03, 9.19e+03, 2.24e+03, 699,
02360    1.99e+03, 3.73e+03, 8.92e+03, 3.21e+04, 8.33e+04, 1.12e+05, 0, 0, 0, 0, 0,
02361    2.4e+03, 8.25e+04, 1.07e+05, 1.89e+04, 1.25e+04, 6.66e+03, 5.02e+03,
02362    8.97e+03, 3.59e+03, 27, 0.0796, 0, 0, 0, 0, 0, 139, 5.42e+03, 2.55e+04,
02363    3.15e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02364    0, 0, 0, 0, 0, 0}},
02365   {80.7, 261, 110, 36.7, 33.4, 331, 532, 669, 224, 1.36e+03, 2.06e+03, 380,
02366    2.74e+03, 1.96e+03, 309, 1.1e+03, 1.1e+03, 713, 3.93e+05, 1.59e+05, 0, 0,
02367    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02368    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02369    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.01e+03,
02370    7.43e+03, 2.86e+03, 8.99e+03, 1.22e+04, 2.6e+03, 4.5e+03, 7.96e+03,
02371    7.85e+03, 1.59e+03, 2.94e+03, 3.67e+03, 775, 466, 115, 14.2, 1.4e+03, 927,
02372    2.16e+04, 1.28e+05, 8.03e+04, 2.4e+04, 2.66e+04, 3.35e+04, 4.17e+03,
02373    5.43e+03, 1.6e+03, 1.27e+03, 687, 1.88e+03, 3.26e+03, 3.1e+05, 5.01e+04,
02374    1.02e+05, 0, 0, 0, 0, 0, 2.98e+04, 1.6e+05, 4.65e+05, 1.5e+05, 9.84e+04,
02375    4.44e+03, 3.12e+03, 1.04e+04, 2.28e+04, 203, 9.85, 0, 0, 0, 0, 4.31e+03,
02376    6.91e+04, 1.79e+05, 1.27e+05, 1.39e+05, 890, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02377    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},
02378   {75.1, 256, 52, 62.6, 34.4, 374, 487, 491, 138, 973, 3.72e+03, 405,
02379    2.03e+03, 2.58e+03, 778, 697, 933, 1.76e+03, 3.76e+05, 1.17e+05, 0, 0, 0,
02380    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02381    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02382    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.25e+04, 0, 390, 0,
02383    3.37e+03, 1.2e+04, 3.67e+03, 9.19e+03, 6.07e+03, 1.83e+04, 1.27e+04,
02384    2.7e+03, 4.4e+03, 4.07e+03, 954, 716, 57, 46.8, 93.1, 338, 1.43e+03,
02385    9.97e+03, 1.29e+04, 3.19e+04, 3.21e+04, 1.23e+04, 7.05e+04, 1.14e+04,
02386    1.43e+03, 1.88e+03, 1.24e+03, 1.26e+03, 1.41e+03, 1.25e+03, 2.69e+03,
02387    2.48e+05, 2.07e+04, 4.43e+04, 0, 0, 0, 0, 0, 1.23e+03, 6.27e+05, 6.15e+05,
02388    5.65e+05, 1.07e+05, 4.71e+04, 7.59e+03, 1.47e+04, 1.28e+04, 625, 102,
02389    0.917, 0, 0, 0, 1.97e+04, 6.51e+04, 1.26e+05, 2.54e+05, 2.69e+05, 3.55e+04,
```

```
02390    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02391    0, 0},
02392   {69.6, 257, 22.9, 87.6, 34, 437, 427, 363, 87.8, 912, 4.53e+03, 298,
02393    1.49e+03, 2.93e+03, 1.09e+03, 160, 513, 5.83e+03, 3.09e+05, 8.52e+04, 2.11,
02394    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02395    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02396    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.71e+03,
02397    0, 4.16e+03, 1.5e+03, 1.83e+03, 1.03e+04, 2.12e+04, 7.28e+03, 1.93e+03,
02398    3.49e+03, 1.49e+03, 524, 368, 129, 8.85e+03, 3.08e+04, 2.76e+04, 3.12e+04,
02399    3.73e+03, 2.79e+04, 1.35e+04, 1.11e+04, 3.59e+04, 7.39e+04, 3.01e+03,
02400    1.36e+03, 6.69e+03, 1.17e+03, 3.5e+03, 3.28e+03, 2.27e+03, 1.96e+04,
02401    6.08e+04, 4.57e+04, 3.8e+04, 0, 1.42e+03, 1.77e+03, 2.24e+03, 0, 0,
02402    2.65e+05, 2.46e+04, 8.28e+05, 5.7e+05, 7.95e+04, 3.45e+04, 1.47e+04,
02403    1.34e+04, 2.9e+03, 370, 2.42, 0.133, 93.4, 52.9, 1.15e+04, 6.13e+04,
02404    4.89e+04, 1.02e+05, 1.21e+05, 1.47e+05, 87.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02405    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02406   {65, 256, 10.6, 87.8, 31.1, 518, 361, 270, 60.4, 1.07e+03, 4.1e+03, 341,
02407    1.12e+03, 3.06e+03, 1.09e+03, 107, 391, 1.03e+04, 3.05e+05, 1.29e+05, 65.3,
02408    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02409    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02410    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 19.7, 162, 146,
02411    3.02e+03, 0, 3.56e+04, 2.71e+05, 2.94e+05, 8.06e+04, 1.58e+04, 4.77e+03,
02412    1.67e+05, 3.98e+04, 1.57e+04, 1.21e+03, 672, 666, 335, 474, 9.67e+03,
02413    4.45e+04, 6.36e+04, 2.6e+04, 1.13e+04, 2.08e+04, 1.67e+04, 4.51e+03,
02414    3.5e+04, 3.4e+04, 943, 2.36e+03, 9.37e+03, 7.73e+03, 1.5e+03, 1.07e+03,
02415    2.71e+03, 2.55e+04, 4.62e+04, 4.57e+04, 3.45e+04, 0, 3.33e+04, 9.52e+04,
02416    6.29e+04, 2.72e+05, 4.47e+04, 2.4e+04, 1.08e+05, 4.59e+05, 4.97e+05,
02417    1.99e+04, 2.14e+04, 1.49e+04, 1.26e+04, 8.76e+03, 258, 60.9, 326, 508, 151,
02418    132, 3.88e+04, 9.4e+04, 6.98e+04, 1.16e+05, 1.95e+05, 901, 0, 0, 0, 0, 0,
02419    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02420   {61, 252, 7.89, 67.6, 25.7, 603, 303, 210, 43.9, 1.38e+03, 2.88e+03, 718,
02421    1.34e+03, 3.27e+03, 1.2e+03, 232, 390, 1.36e+04, 2.01e+05, 1.98e+05, 20.5,
02422    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02423    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02424    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 40.1, 5.16e+03, 4.43e+03,
02425    5.31e+03, 5.78e+04, 5.09e+04, 1.17e+04, 2.53e+03, 2.79e+05, 1.91e+05,
02426    8.6e+04, 2.43e+03, 2.71e+04, 1.08e+04, 2.88e+03, 947, 137, 4.08e+03,
02427    2.12e+03, 963, 1.43e+03, 1.79e+04, 2.69e+04, 1.23e+03, 6.25e+03, 2.13e+03,
02428    2.21e+03, 5.11e+04, 8.56e+04, 6.82e+03, 2.36e+03, 2.45e+03, 2.18e+03,
02429    6.73e+03, 5.16e+03, 6.76e+03, 1.16e+04, 7.97e+03, 1.15e+04, 3.85e+04,
02430    1.03e+04, 0, 215, 2.25e+04, 9.5e+03, 6.87, 664, 1.77e+05, 8.96e+04,
02431    7.42e+05, 4.31e+05, 3.31e+04, 1.07e+04, 8.43e+03, 1.23e+04, 1.66e+04, 656,
02432    266, 455, 1e+03, 425, 43.3, 1.45e+04, 4.04e+04, 6.47e+04, 6.61e+04,
02433    1.15e+05, 1.85e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02434    0, 0, 0, 0, 0, 0, 0, 0},
02435   {58, 247, 14.1, 51.7, 20.6, 683, 254, 184, 25.3, 2.17e+03, 2.41e+03,
02436    1.11e+03, 1.35e+03, 3.01e+03, 1.26e+03, 763, 744, 1.61e+04, 1.32e+05,
02437    2.58e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02438    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02439    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.95e+03,
02440    4.54e+04, 2.11e+04, 2.46e+04, 1.48e+04, 9.22e+03, 8.63e+03, 2.54e+04,
02441    3.98e+04, 5.31e+04, 1.33e+05, 2.24e+05, 3.68e+04, 9.43e+03, 1.3e+04,
02442    5.54e+03, 3.8e+03, 239, 2.35e+03, 3.81e+03, 3.23e+03, 46.3, 1.2e+03,
02443    1.5e+03, 154, 179, 1.91e+03, 1.69e+04, 1.66e+04, 1.22e+04, 8.44e+03,
02444    1.13e+03, 2.12e+03, 1.15e+03, 2.2e+03, 1.97e+03, 3.53e+04, 5.15e+03, 43.6,
02445    1.6e+03, 556, 2.44e+03, 0, 0, 0, 0, 287, 3.07e+04, 1.21e+05, 4.85e+03,
02446    5.55e+05, 4.52e+05, 7.05e+04, 3.25e+03, 8.76e+03, 2.79e+04, 2.32e+04, 648,
02447    588, 311, 128, 20.8, 24.6, 2.37e+03, 2.25e+04, 6.41e+04, 4.48e+04,
02448    7.39e+04, 2.68e+04, 2.36e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 141,
02449    2.24e+04, 1.9e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02450   {56.1, 243, 20.8, 44, 13.3, 737, 209, 169, 29.8, 3.18e+03, 3.15e+03,
02451    1.36e+03, 1.16e+03, 1.94e+03, 1.03e+03, 463, 1.53e+03, 1.79e+04, 2.4e+05,
02452    2.26e+05, 36.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02453    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02454    0, 0, 0, 0, 0, 0, 4.14e+03, 1.04e+05, 6.59e+04, 1.75e+04, 1.84e+04, 3.35e+03,
02455    0, 0, 0, 0, 0, 0, 0, 0, 514, 2.85e+03, 1.14e+04, 2.51e+04, 2.17e+04, 1.78e+04,
02456    6.96e+03, 3.69e+03, 4.49e+03, 6.68e+03, 1.11e+04, 5.73e+03, 8.37e+03,
02457    8.85e+04, 6.53e+04, 5.11e+04, 2.66e+04, 8.67e+03, 2.7e+03, 145, 1.38e+03,
02458    756, 1.65e+03, 292, 1.96e+03, 592, 3.36e+03, 1.44e+03, 1.04e+04, 6.2e+03,
02459    1.18e+04, 7.2e+03, 1.67e+04, 942, 2.09e+03, 4.4e+03, 241, 706, 5.86e+04,
02460    730, 0, 0, 0, 168, 17.9, 0, 0, 0, 1.81e+04, 2.61e+04, 1.08e+05, 814,
02461    5.41e+05, 3.32e+05, 8.22e+04, 3.68e+03, 8.89e+03, 1.72e+04, 1.35e+03, 514,
02462    694, 270, 311, 715, 2.61e+03, 2.1e+03, 6.47e+03, 2.3e+04, 1.68e+04,
02463    3.87e+04, 4.39e+04, 3.24e+04, 6.78e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02464    299, 6.57e+04, 5.23e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02465   {54.6, 237, 24.3, 43.8, 8.11, 758, 171, 156, 69.5, 4.33e+03, 3.35e+03,
02466    1.09e+03, 1.11e+03, 1.13e+03, 1.09e+03, 240, 1.96e+03, 1.83e+04, 1.47e+05,
02467    2.05e+05, 2.93e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02468    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02469    0, 0, 0, 0, 0, 202, 5.06e+04, 8.09e+04, 5.87e+04, 9.02e+04, 9.45e+04,
02470    2.7e+05, 6.25e+04, 7.62e+04, 0, 0, 727, 1.27e+03, 7.58e+03, 2.67e+04,
02471    1.12e+04, 1.79e+04, 1.42e+04, 2.03e+04, 1.45e+04, 9.64e+03, 1.15e+04,
02472    5.55e+03, 3.48e+03, 1.72e+03, 1.59e+03, 1.89e+03, 5.17e+03, 1.52e+04,
02473    8.54e+04, 5.13e+04, 4.25e+04, 8.79e+03, 826, 37.6, 316, 388, 735, 632, 276,
02474    1.1e+03, 1.31e+03, 1.16e+04, 9.89e+03, 4.26e+03, 2.15e+03, 3.17e+03,
02475    4.3e+03, 6.43e+03, 5.15e+03, 1.34e+03, 944, 1.31e+04, 4.51e+04, 2.71e+03,
02476    0, 0, 0, 1.2e+03, 6.37e+04, 0, 0, 8.36e+03, 2.15e+05, 1.69e+05, 2.62e+04,
```

```
02477    0, 2.14e+05, 3.14e+05, 1.57e+05, 3.78e+04, 2.29e+04, 3.89e+04, 950, 379,
02478    652, 63.2, 2.7e+03, 4.76e+03, 4.43e+03, 1.45e+03, 1.06e+04, 1.29e+04,
02479    1.98e+04, 3.98e+04, 2.81e+04, 4.78e+04, 6.47e+04, 8.52e+03, 6.16e+03, 0, 0,
02480    0, 0, 0, 0, 0, 0, 0, 1.44e+04, 7.36e+04, 2.41e+04, 0, 0, 0, 0, 0, 0, 0, 0,
02481    0, 0},
02482   {54.1, 231, 23.9, 45.4, 7.16, 753, 140, 146, 136, 5.72e+03, 3.02e+03,
02483    1.12e+03, 1.27e+03, 975, 1.23e+03, 144, 3.18e+03, 1.82e+04, 1.59e+05,
02484    1.93e+05, 2.32e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02485    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02486    0, 0, 0, 0.804, 1.14e+04, 5.9e+04, 8.56e+04, 6.69e+04, 2.28e+04, 2.27e+04,
02487    1.2e+04, 2.09e+04, 1.2e+05, 6.07e+04, 2.43e+05, 1.55e+05, 1.88e+04,
02488    1.54e+04, 2.94e+04, 7e+03, 1.61e+04, 1.95e+04, 1.29e+04, 1.51e+03,
02489    7.98e+03, 9.23e+03, 5.81e+03, 1.28e+03, 3.03e+03, 2.36e+03, 3.2e+03,
02490    4.28e+03, 5.44e+03, 3.54e+04, 9.6e+04, 3.42e+03, 2.22e+04, 3.87e+04, 63.9,
02491    107, 194, 651, 380, 1.75e+03, 1.99e+03, 646, 647, 3.6e+03, 9.28e+03,
02492    4.43e+03, 876, 1.07e+03, 2.08e+03, 8.01e+03, 1.49e+04, 4.4e+03, 5.12e+03,
02493    4.28e+03, 1.34e+04, 9.34e+03, 0, 0, 129, 3.17e+04, 1.27e+05, 0, 0,
02494    6.23e+04, 2.77e+05, 5.79e+04, 225, 3.74e+04, 1.98e+05, 2.33e+05, 1.84e+05,
02495    3.62e+04, 1.52e+04, 2.55e+04, 782, 802, 383, 64.9, 2.63e+03, 1.18e+03,
02496    5.99e+03, 1.37e+03, 8.8e+03, 1.12e+04, 1.29e+04, 3.28e+04, 3.29e+04,
02497    4.83e+04, 8.13e+04, 1.67e+05, 5.87e+03, 349, 0, 0, 0, 0, 0, 0, 3.43e+03,
02498    2.78e+04, 7.34e+04, 1.91e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02499   {54.9, 224, 20.4, 34.9, 9.39, 732, 113, 139, 256, 6.86e+03, 2.28e+03,
02500    1.15e+03, 1.21e+03, 677, 1.28e+03, 185, 4.58e+03, 2.44e+04, 1.4e+05,
02501    1.25e+05, 3.61e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02502    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.21,
02503    1.22e+04, 7e+04, 8.37e+04, 5.2e+04, 6.15e+04, 8.8e+04, 4.76e+04, 2.17e+04,
02504    4.96e+03, 303, 118, 1.34e+03, 3.6e+03, 1.91e+04, 6.58e+04, 6.95e+04,
02505    2.67e+04, 1.2e+05, 5.3e+04, 3.59e+04, 3.5e+04, 2.67e+04, 1.37e+04,
02506    1.57e+04, 1.35e+03, 3.95e+03, 5.59e+03, 3.51e+03, 1.53e+03, 2.92e+03,
02507    4.28e+03, 785, 6.76e+03, 1.44e+04, 4e+04, 2.11e+04, 4.93e+03, 4.65e+03,
02508    245, 46.8, 85.5, 377, 437, 129, 422, 846, 1.35e+03, 958, 1.82e+03,
02509    1.69e+04, 2.01e+04, 1.49e+03, 2.57e+03, 1.25e+03, 776, 2.8e+03, 8.46e+03,
02510    1.04e+04, 3.07e+03, 1.55e+03, 1.5, 0, 0, 9.17, 4.06e+04, 2.16e+05, 173,
02511    6.31e+04, 1.04e+05, 5.36e+04, 0, 3.05e+04, 1.6e+05, 9.73e+04, 1.67e+05,
02512    1.31e+05, 1.46e+04, 1.66e+04, 3.17e+04, 1.32e+03, 682, 762, 399, 1.75e+03,
02513    2.5e+03, 5.27e+03, 2.17e+03, 6.74e+03, 8.74e+03, 8.52e+03, 3.51e+03,
02514    1.43e+04, 3.1e+04, 4.72e+04, 1.26e+05, 3.06e+04, 1.22e+04, 0, 0, 0, 0, 0,
02515    0, 0, 4.76e+04, 4.24e+04, 1.28e+05, 3.64e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02516    0},
02517   {56.1, 216, 15.5, 20.7, 14.5, 696, 93.2, 135, 404, 7.41e+03, 1.91e+03,
02518    1.19e+03, 1.3e+03, 469, 1.26e+03, 348, 5.47e+03, 2.73e+04, 1.08e+05,
02519    7.68e+04, 1.6e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02520    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 55.2,
02521    4.75e+04, 1.09e+05, 9.44e+04, 9.42e+04, 1.3e+05, 8.49e+04, 3.84e+04,
02522    2.53e+04, 8.89e+03, 1.1e+03, 103, 239, 3.26e+03, 8.16e+03, 7.96e+03,
02523    5.74e+03, 1.05e+04, 2.53e+04, 2.53e+04, 2.08e+04, 2.97e+04, 1.81e+04,
02524    1.02e+04, 4e+03, 1.67e+04, 2.19e+04, 1.29e+04, 3.18e+03, 276, 762,
02525    5.39e+03, 5.69e+03, 4.57e+03, 7.74e+03, 9.52e+03, 2.71e+04, 6.53e+03,
02526    1.99e+03, 1.52e+03, 142, 101, 75.6, 257, 286, 244, 192, 928, 2.55e+03,
02527    2.52e+03, 1.23e+05, 2.4e+04, 6.21e+03, 4.06e+03, 7.02e+03, 3.51e+03,
02528    3.3e+03, 1.17e+03, 1.53e+04, 8.22e+03, 4.47e+03, 1.9, 0, 0, 0, 0, 0,
02529    9.87e+04, 1.57e+05, 1.82e+05, 8.39e+04, 3.38e+04, 293, 1.37e+05, 9.39e+04,
02530    1.46e+04, 4.2e+04, 1.12e+05, 1.62e+04, 2.56e+04, 5.56e+04, 1.02e+03, 925,
02531    3.39e+03, 0, 543, 4.02e+03, 2.49e+03, 755, 3.13e+03, 7.37e+03, 8.78e+03,
02532    7.8e+03, 5.42e+03, 2.23e+04, 3.81e+04, 8.25e+04, 8.91e+04, 3.87e+04, 0, 0,
02533    0, 0, 0, 0, 0, 5.02e+04, 7.6e+04, 8.42e+04, 5.66e+04, 0, 0, 0, 0, 0, 0, 0,
02534    0, 0, 0},
02535   {58.6, 210, 10.1, 9.44, 23.2, 646, 83.8, 133, 618, 6.92e+03, 1.99e+03,
02536    1.43e+03, 1.68e+03, 420, 1.27e+03, 386, 6.11e+03, 2.34e+04, 1.28e+05,
02537    8.89e+04, 429, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02538    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 81.4, 8.95e+03,
02539    6.19e+04, 1.49e+05, 5.83e+04, 3.43e+04, 1.61e+04, 1.33e+04, 3.13e+04,
02540    2.88e+04, 6.36e+03, 7.71e+03, 4.22e+03, 258, 225, 2.61e+03, 6.17e+03,
02541    3.38e+03, 1.47e+04, 1.97e+04, 3.14e+04, 9.93e+03, 4.12e+04, 8.85e+03,
02542    1.41e+04, 1.71e+04, 1.39e+04, 1.09e+04, 2.98e+03, 3.89e+03, 643, 418, 205,
02543    946, 1.27e+03, 4.54e+03, 4.45e+03, 5.08e+03, 3.94e+03, 633, 1.43e+03, 857,
02544    434, 743, 142, 77.3, 55.8, 392, 204, 1.8e+03, 4.66e+03, 5.59e+04, 2.01e+05,
02545    4.9e+04, 3.37e+03, 8.96e+03, 1.22e+04, 2.25e+03, 1.4e+04, 2.39e+04,
02546    7.8e+03, 4.93e+03, 4.33e+03, 0, 0, 0, 0, 0, 0, 21.2, 1.61e+05, 1.75e+04,
02547    2.62e+04, 0, 5.27e+04, 1.13e+05, 6.07e+04, 6e+03, 2.29e+03, 5.15e+03,
02548    1.03e+04, 4.26e+04, 7.92e+03, 624, 517, 2.09e+03, 300, 0.872, 80.1, 158,
02549    463, 618, 3.52e+03, 1.12e+04, 8.42e+03, 5.86e+03, 7.04e+03, 1.74e+04,
02550    6.18e+04, 1.07e+05, 4.49e+04, 993, 0, 0, 0, 0, 0, 0, 3.49e+04, 5.78e+04,
02551    1.16e+05, 1.05e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02552   {61.6, 203, 6.15, 4.48, 38.9, 582, 85.6, 112, 1.23e+03, 6.08e+03, 2.41e+03,
02553    1.74e+03, 1.66e+03, 433, 1.18e+03, 216, 7.55e+03, 4.27e+04, 7.81e+04,
02554    1.03e+05, 694, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02555    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.13e+03, 2.57e+03, 3.04e+03,
02556    3.82e+04, 8.58e+04, 3.89e+04, 5.15e+04, 3.63e+04, 5.91e+03, 4.4e+03,
02557    1.71e+03, 1e+04, 4.41e+03, 9.56e+03, 1.75e+04, 1.03e+04, 139, 410,
02558    1.77e+03, 4.59e+03, 4.5e+03, 4.5e+03, 7.12e+03, 7.04e+03, 2.33e+04,
02559    1.4e+04, 1.28e+04, 2.27e+04, 9.87e+03, 1.56e+04, 2.06e+03, 641, 2.03e+03,
02560    3.65e+03, 2.43e+03, 872, 1.51e+03, 1.86e+03, 5.61e+03, 1.31e+04, 2.91e+03,
02561    883, 480, 4.94e+03, 1.89e+03, 8.71e+03, 8.15e+03, 584, 114, 175, 287, 280,
02562    2.34e+03, 7.91e+03, 2.58e+05, 1.41e+05, 5.13e+04, 3.52e+03, 7.8e+03,
02563    3.33e+03, 3.43e+03, 1.88e+04, 1.3e+04, 6.02e+03, 5.39e+03, 2.62e+03, 0, 0,
```

```
02564    0, 0, 0, 0, 0, 0, 1.37e+03, 1.57e+03, 0, 2.06e+05, 1.4e+05, 2.46e+04,
02565    1.06e+04, 8.68e+03, 3.68e+03, 2.84e+04, 2.01e+04, 803, 742, 592, 1.28e+03,
02566    2.22e+03, 0, 78.3, 104, 337, 173, 8.2, 0.415, 1.14e+04, 7.92e+03, 6.1e+03,
02567    5.89e+03, 9.32e+04, 9.13e+04, 8.77e+04, 1.26e+04, 0, 0, 0, 0, 0, 0, 242,
02568    4.27e+04, 6.99e+04, 1.85e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02569   {64.8, 196, 4.29, 8.14, 61.9, 507, 75.6, 123, 2.05e+03, 5.38e+03, 2.97e+03,
02570    1.56e+03, 1.13e+03, 425, 1.05e+03, 117, 1.08e+04, 7.99e+04, 7.94e+04,
02571    1.24e+05, 6.99e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02572    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 436, 1.09e+05, 1.96e+05,
02573    8.82e+04, 5.74e+04, 3.61e+03, 3.72e+04, 2.42e+04, 4.52e+04, 1.19e+03, 925,
02574    1.22e+03, 4.03e+03, 1.95e+03, 4.77e+03, 1.01e+03, 264, 746, 404, 1.23e+03,
02575    3.81e+03, 6.32e+03, 6.99e+03, 4.03e+03, 8.09e+03, 1.36e+04, 3.37e+03,
02576    2.99e+03, 1.02e+04, 1.29e+04, 9.51e+03, 4.02e+03, 460, 237, 1.86e+03,
02577    1.25e+03, 239, 1.21e+03, 824, 1.07e+04, 6.33e+03, 2.38e+03, 3.07e+03, 207,
02578    444, 779, 3.29e+03, 8.16e+03, 420, 209, 256, 242, 1.12e+03, 2.41e+03,
02579    9.9e+03, 2.64e+05, 2.09e+05, 1.17e+05, 3.65e+04, 1.48e+04, 7.35e+03,
02580    3.12e+03, 904, 1.3e+03, 3.08e+03, 5.4e+03, 0.982, 0, 0, 0, 0, 0, 0, 0, 0,
02581    1.39e+03, 5.26, 5.34e+04, 1.55e+05, 9.96e+04, 1.41e+04, 815, 1.21e+03,
02582    2.19e+04, 5.67e+04, 4.11e+04, 2.89e+03, 1.17e+03, 442, 1.71e+03, 14.4, 0,
02583    47.7, 106, 3.94, 89.9, 0, 0, 1.75e+03, 6.65e+03, 4.31e+03, 5.89e+03,
02584    2.73e+04, 8.62e+04, 1.39e+05, 5.16e+04, 0, 0, 0, 381, 21.6, 0, 0, 1.4e+04,
02585    4.65e+04, 2.04e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02586   {68.8, 188, 4.27, 22.1, 87.4, 440, 43.2, 233, 2.88e+03, 4.74e+03, 3.26e+03,
02587    1.18e+03, 619, 423, 837, 102, 1.31e+04, 6.36e+04, 1.21e+05, 1.09e+05, 0, 0,
02588    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02589    0, 0, 0, 0, 0, 0, 0, 0, 2.73e+03, 1.15e+05, 6.63e+04, 1.56e+05, 1.11e+04,
02590    1.34e+03, 1.76e+04, 9.24e+03, 3.27e+03, 747, 1.24e+03, 1.58e+03, 1.03e+03,
02591    4.96e+03, 4.43e+03, 726, 863, 776, 522, 697, 1.42e+03, 3.78e+03, 6.1e+03,
02592    4.83e+03, 3.37e+03, 5.7e+03, 4.21e+03, 8.85e+03, 1.19e+04, 4.2e+03,
02593    5.6e+03, 2.58e+03, 345, 178, 417, 570, 60.6, 872, 3.85e+03, 6.81e+03,
02594    3.02e+03, 2.37e+03, 3.03e+03, 1.28e+03, 3.46e+03, 121, 191, 447, 710, 805,
02595    594, 608, 863, 3.03e+03, 2.24e+03, 3.36e+03, 3.2e+03, 4.93e+03, 1.28e+04,
02596    1.02e+04, 2.83e+03, 1.31e+03, 430, 1.33e+03, 6.75e+03, 460, 844, 5.51e+03,
02597    0, 0, 0, 0, 0, 920, 3.15e+03, 2.1e+05, 2.66e+05, 2.52e+05, 1.42e+05,
02598    8.95e+04, 3.79e+03, 258, 418, 1.57e+04, 1.19e+05, 4.08e+04, 1.61e+03,
02599    2.67e+03, 1.07e+03, 2.27e+03, 359, 0, 0, 0, 0, 28.8, 0, 0, 0, 1e+04,
02600    2.98e+03, 7.87e+03, 1.06e+04, 1.56e+04, 8.11e+04, 8.21e+04, 0, 0, 0, 0, 0,
02601    0, 0, 0.36, 2.58e+04, 4.45e+04, 576, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02602   {71.8, 179, 5.85, 45.8, 108, 396, 27.3, 405, 3.77e+03, 4.26e+03, 3.18e+03,
02603    1.14e+03, 516, 684, 695, 152, 1.38e+04, 8.52e+04, 1.32e+05, 8.9e+04, 0, 0,
02604    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02605    0, 0, 0, 0, 0, 0, 0, 0, 0, 7.42e+04, 7.9e+04, 2.5e+04, 9.82e+03, 2.06e+03,
02606    5.71e+03, 2.47e+03, 766, 479, 1.33e+03, 1.54e+03, 614, 1.85e+03, 5.01e+03,
02607    3.78e+03, 1.09e+03, 304, 685, 1.03e+03, 2.2e+03, 3.91e+03, 3.52e+03,
02608    1.06e+03, 1.22e+03, 1.24e+03, 6.07e+03, 5.12e+03, 5.21e+03, 3.14e+03,
02609    5.5e+03, 2.08e+03, 1.53e+03, 127, 181, 261, 41.4, 50.8, 399, 2.82e+03,
02610    3.78e+03, 1.92e+03, 2.22e+03, 6.46e+03, 6.17e+03, 210, 289, 2.44e+03,
02611    3.07e+03, 1.52e+04, 7.87e+03, 2.05e+03, 2.21e+03, 5.77e+03, 3.32e+03,
02612    1.56e+03, 1.9e+03, 5.14e+03, 3.94e+03, 3.01e+03, 4.24e+03, 1.33e+03, 232,
02613    346, 1.13e+03, 285, 865, 1.81e+04, 2.24e+04, 0, 0, 0, 0, 1.85e+04,
02614    1.48e+05, 1.68e+05, 1.19e+05, 2.69e+05, 6.69e+04, 4.07e+04, 3.07e+03, 435,
02615    57.8, 1.24e+04, 5.22e+04, 1.91e+04, 3.35e+03, 1.28e+03, 582, 2.05e+03, 205,
02616    395, 266, 0, 0, 0.00332, 7.76, 54, 78.9, 2.58e+03, 8.65e+03, 5.38e+03,
02617    6.51e+03, 1.17e+04, 3.1e+04, 1.11e+05, 0, 0, 0, 0, 0, 0, 199, 2.8e+04,
02618    1.94e+04, 4.82e+04, 570, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02619   {75.9, 170, 9.8, 75.1, 127, 388, 50.5, 480, 4.46e+03, 3.34e+03, 2.7e+03,
02620    1.17e+03, 760, 1.37e+03, 984, 412, 1.41e+04, 7.1e+04, 1.25e+05, 3.78e+04,
02621    3.72e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02622    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.64e+04, 5.94e+04, 7.01e+04,
02623    1.65e+04, 3.7e+03, 6.73e+03, 5.61e+03, 3.25e+03, 202, 1.31e+03, 770,
02624    2.08e+03, 1.33e+03, 3.46e+03, 220, 387, 248, 145, 282, 628, 768, 226, 131,
02625    760, 1.17e+03, 5.34e+03, 4.04e+03, 3.55e+03, 1.33e+03, 2.96e+03, 4.96e+03,
02626    5.15e+03, 361, 222, 121, 22.6, 36.4, 4.99e+03, 3.92e+03, 4.45e+03, 1e+03,
02627    686, 3.75e+03, 7.64e+03, 2.89e+03, 1.08e+03, 4.47e+03, 9.15e+03, 1.15e+04,
02628    1.28e+04, 1.64e+04, 4.87e+04, 1.35e+04, 2.1e+03, 3.76e+03, 1.15e+03, 389,
02629    3.06e+03, 1.3e+04, 1.33e+03, 1.14e+03, 448, 330, 940, 412, 781, 1.23e+04,
02630    8.36e+04, 0, 0, 0.00111, 1.46e+05, 2.42e+05, 2.32e+05, 1.8e+05, 2.06e+05,
02631    1.19e+05, 1.16e+05, 5.05e+04, 7.58e+04, 3.51e+04, 2.11e+04, 787, 4.62e+04,
02632    9.11e+03, 1.3e+03, 404, 412, 908, 691, 1.47e+03, 1.1e+03, 302, 23.3, 34.4,
02633    526, 2.05e+03, 1.88e+03, 0.655, 25, 7.21e+03, 6.03e+03, 5.08e+03, 1.53e+04,
02634    1.13e+05, 3.54e+03, 0, 0, 0, 0, 0, 0, 3e+04, 4.6e+03, 3.38e+04, 0, 0, 0, 0,
02635    0, 0, 0, 0, 0, 0},
02636   {80.4, 160, 16.9, 104, 137, 409, 92.5, 340, 5.31e+03, 2.46e+03, 2.17e+03,
02637    1.11e+03, 996, 1.63e+03, 439, 812, 1.26e+04, 3.42e+04, 2.65e+05, 7.62e+04,
02638    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02639    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.75e+05, 1.83e+04, 5.27e+04, 8.95e+03,
02640    3.98e+03, 1.87e+04, 1.65e+04, 8.46e+03, 1.56e+03, 426, 516, 1.74e+03, 318,
02641    1.01e+03, 216, 174, 153, 112, 423, 420, 394, 6e+03, 6.15e+03, 3.06e+03,
02642    2.69e+03, 4.99e+03, 4.2e+03, 4.99e+03, 5.49e+03, 4.41e+03, 6.31e+03,
02643    4.41e+03, 658, 135, 63.1, 26.5, 16.9, 2.76e+03, 3.13e+03, 2.1e+03,
02644    1.05e+03, 1.81e+03, 7.64e+03, 2.01e+04, 3.03e+03, 2.31e+03, 1.29e+04,
02645    1.14e+04, 1.63e+03, 1.18e+03, 5.04e+03, 9.24e+03, 2.33e+04, 7.92e+03,
02646    1.31e+03, 1.33e+03, 502, 2.29e+03, 3e+03, 5e+03, 5.32e+03, 795, 390,
02647    1.13e+03, 373, 450, 4.14e+03, 8.15e+03, 0, 0, 1.24e+03, 1.19e+05, 9.26e+04,
02648    1.15e+05, 5.04e+04, 7.06e+04, 2.27e+05, 1.07e+05, 2.39e+04, 2.61e+05,
02649    1.02e+05, 1.39e+05, 2.17e+04, 4.69e+04, 5.33e+03, 1.48e+03, 212, 496,
02650    1.23e+03, 967, 795, 2.02e+03, 38.7, 30.6, 649, 478, 851, 913, 288, 0.00111,
```

```
02651    3.14e+03, 1.74e+03, 3.75e+03, 6.15e+03, 6.68e+04, 1.14e+04, 0, 0, 0, 0, 0,
02652    7.31e+03, 1.89e+04, 2.6e+03, 2.98e+04, 175, 0, 0, 0, 0, 0, 0, 0, 0},
02653   {86.8, 150, 27.6, 129, 123, 446, 79.8, 192, 6.98e+03, 2.08e+03, 1.85e+03,
02654    1.1e+03, 1.24e+03, 1.64e+03, 114, 1.19e+03, 1.02e+04, 2.29e+04, 3.61e+05,
02655    4.9e+04, 1.72e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02656    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.58e+05, 2.28e+04,
02657    1.03e+05, 1.01e+04, 9.22e+03, 1.21e+04, 2.11e+04, 7.95e+03, 2.44e+03, 811,
02658    251, 913, 274, 358, 77.1, 493, 1.52e+03, 3.5e+03, 1.4e+03, 1.19e+03,
02659    1.5e+03, 4.76e+03, 8.13e+03, 9.86e+03, 3.98e+03, 2.62e+03, 4.18e+03,
02660    8.6e+03, 5.89e+03, 2.81e+03, 3.48e+03, 959, 4.08e+03, 991, 24.8, 59.8, 122,
02661    927, 2.4e+03, 1.49e+03, 421, 5.31e+03, 2.95e+03, 8.75e+03, 3.98e+03,
02662    1.1e+03, 2.5e+04, 9.94e+04, 2.39e+04, 2.4e+03, 3.45e+03, 9.17e+03,
02663    7.84e+03, 6.34e+03, 1.24e+03, 1.74e+03, 2.74e+03, 4.34e+03, 4.64e+03,
02664    4.04e+03, 3e+03, 1.22e+03, 313, 1.07e+03, 1.51e+03, 765, 6.84e+03, 0, 0,
02665    1.14e+05, 600, 751, 3.12e+04, 3.11e+04, 1.27e+04, 1.7e+05, 2.7e+05,
02666    1.28e+05, 2.44e+03, 1.92e+05, 1.97e+05, 1.21e+04, 1.06e+05, 1.05e+05,
02667    2.53e+03, 459, 110, 401, 606, 677, 572, 173, 56.6, 308, 1.09e+03, 546, 975,
02668    1.63e+03, 1.93e+03, 28.5, 2.32e+03, 2.63e+03, 4.55e+03, 2.72e+03, 1.21e+04,
02669    1.57e+04, 0, 0, 0, 0, 0, 4.27, 6.38e+03, 8.27e+03, 6.29e+04, 8.52, 0, 0, 0,
02670    0, 0, 0, 0, 0, 0},
02671   {91.3, 141, 41.9, 150, 90.5, 481, 83.1, 261, 8.12e+03, 1.97e+03, 1.67e+03,
02672    1.17e+03, 1.48e+03, 1.35e+03, 78.7, 1.79e+03, 7.76e+03, 8.29e+03, 5.37e+05,
02673    7.92e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02674    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.96e+04, 5.33e+04, 8.79e+04,
02675    1.41e+04, 1.11e+04, 2.99e+03, 5.59e+03, 4.04e+03, 4.89e+03, 3.75e+03,
02676    3.31e+03, 525, 1.05e+03, 471, 792, 1.11e+03, 9.14e+03, 1.39e+03, 3.82e+03,
02677    601, 2.2e+03, 2.38e+03, 1.07e+04, 1.44e+04, 1.51e+04, 1.73e+04, 5.01e+03,
02678    5.59e+03, 4.18e+03, 266, 676, 1.75e+03, 3.49e+03, 63.1, 34, 101, 1.05e+03,
02679    4.99e+03, 707, 607, 224, 3.52e+03, 3.72e+03, 6.56e+03, 2.16e+03, 875,
02680    8.81e+03, 3.79e+04, 2.58e+04, 1.03e+04, 6.83e+03, 1.13e+03, 1.4e+03, 644,
02681    698, 1.26e+03, 1.72e+04, 8.85e+03, 2.65e+04, 9.37e+03, 3.64e+03, 1.01e+03,
02682    1.71e+03, 1.75e+03, 3.42e+03, 839, 6.89e+03, 0, 0, 1.17e+05, 0.00111,
02683    3.13e+03, 8.57e+03, 652, 167, 1.28e+05, 2.04e+05, 1.36e+05, 2.45e+03,
02684    1.6e+05, 1.68e+05, 1.17e+05, 1.25e+05, 5e+03, 4.5e+03, 570, 118, 553, 481,
02685    1.07e+03, 527, 1.93e+03, 484, 394, 538, 816, 1.28e+03, 978, 800, 925,
02686    2.18e+03, 1.65e+03, 3.76e+03, 3.72e+03, 3.94e+04, 1.82e+04, 0, 0, 0, 0,
02687    0.0499, 0.787, 0, 3.15e+03, 5.46e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02688   {98.4, 131, 60.6, 148, 36.4, 376, 81.8, 360, 7.81e+03, 2.76e+03, 1.7e+03,
02689    1.21e+03, 1.23e+03, 935, 65.5, 2.35e+03, 6.78e+03, 1.87e+04, 4.31e+05,
02690    7.64e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02691    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.7e+03, 6.37e+04, 5.87e+04,
02692    2.28e+04, 7.41e+03, 4.32e+03, 1.51e+03, 7.28e+03, 8.77e+03, 1.41e+04,
02693    5.88e+03, 6.36e+03, 2.11e+03, 2.93e+03, 713, 1.91e+03, 1.44e+04, 5.68e+03,
02694    2.63e+03, 2.52e+03, 2.64e+03, 2.27e+03, 4.82e+03, 9.49e+03, 8.87e+04,
02695    4.75e+04, 1.45e+04, 8.05e+03, 1.69e+03, 426, 3.12e+03, 1.26e+04, 1.85e+04,
02696    5.65e+03, 4.22e+03, 5.53e+03, 5.12e+03, 2.21e+03, 186, 70.3, 180, 1.63e+03,
02697    7.04e+03, 2.71e+03, 3.66e+03, 126, 1.08e+03, 1.76e+03, 2.19e+04, 2.04e+04,
02698    1.48e+04, 1.89e+03, 579, 132, 248, 1.4e+03, 8.05e+03, 1.89e+04, 2.28e+04,
02699    1.82e+04, 2.67e+03, 4.06e+03, 1.45e+03, 2.87e+03, 1.06e+03, 2.13e+03, 189,
02700    0, 0, 5.16e+04, 0, 3.02e+03, 9.13e+03, 1.99e+04, 1.45e+04, 4.33e+04,
02701    1.51e+04, 5.85e+04, 5.75e+03, 1.18e+05, 7.37e+04, 9.28e+04, 3.73e+04,
02702    2.26e+03, 2.29e+03, 621, 181, 530, 1.56e+03, 900, 895, 2.44e+03, 642, 453,
02703    257, 1.24e+03, 963, 892, 832, 880, 1.14e+03, 2.2e+03, 1.51e+03, 5.11e+03,
02704    2.66e+04, 1.95e+04, 0, 0, 0, 0, 0, 0, 0, 89.7, 3.75e+04, 0, 0, 0, 0, 0, 0,
02705    0, 0, 0, 0},
02706   {109, 116, 82, 160, 30.5, 180, 88.8, 351, 6.35e+03, 3.94e+03, 1.33e+03,
02707    1.06e+03, 944, 814, 105, 3.22e+03, 6.59e+03, 9.14e+04, 2.01e+05, 8.58e+04,
02708    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02709    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.36e+03, 1.17e+05, 5.69e+04, 4.54e+04,
02710    1.15e+04, 1.04e+04, 3.94e+03, 3.06e+03, 2.12e+04, 8.88e+03, 2.46e+03,
02711    1.02e+04, 7.8e+03, 3.99e+03, 5.9e+03, 2.22e+03, 4.38e+04, 5.76e+04,
02712    4.32e+03, 8.17e+03, 3.06e+03, 2.58e+03, 2.94e+03, 1.4e+04, 2.01e+04,
02713    9.9e+04, 5.27e+04, 1.41e+04, 2.62e+03, 9.31e+03, 1.61e+04, 2.14e+04,
02714    2.9e+04, 1.73e+04, 1.83e+04, 861, 5.55e+03, 971, 489, 418, 302, 1.51e+03,
02715    2.53e+03, 1.14e+03, 1.1e+03, 101, 700, 1.38e+03, 3.01e+03, 3.14e+04,
02716    5.86e+03, 2.25e+03, 4.09e+03, 1.72e+03, 1.39e+03, 2.25e+03, 4.54e+03,
02717    5.25e+03, 9.05e+03, 1.71e+04, 526, 4.63e+03, 350, 1.54e+03, 5.82e+03,
02718    4.29e+03, 3.01, 0, 0, 0, 682, 4.42e+03, 3.68e+04, 6.76e+04, 6.85e+04,
02719    9.38e+03, 1.96e+04, 2.01e+04, 1.38e+04, 1.23e+04, 8.18e+04, 8.63e+03,
02720    4.27e+03, 2.5e+03, 1.45e+03, 449, 96.7, 478, 1.73e+03, 550, 490, 438,
02721    2.65e+03, 723, 28.3, 1.2e+03, 453, 638, 826, 492, 1.29e+03, 1.74e+03,
02722    1.63e+03, 5.42e+03, 8.62e+03, 1.33e+04, 0, 0, 0, 0, 0, 0, 0, 345, 4.99e+03,
02723    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02724   {120, 97.1, 103, 218, 60.8, 78.3, 80.5, 602, 4.5e+03, 4.78e+03, 753, 958,
02725    833, 1.05e+03, 319, 4.43e+03, 4.53e+03, 9.21e+04, 1.56e+05, 1e+05, 0, 0, 0,
02726    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02727    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.58e+04, 6.87e+04, 1.72e+05, 1.35e+05,
02728    1.47e+05, 1.06e+04, 4.04e+03, 3.23e+04, 2.09e+04, 4.63e+04, 5.79e+03,
02729    4.6e+03, 2e+04, 1.15e+04, 3.15e+03, 2.05e+04, 2.97e+04, 3.07e+04, 3.97e+03,
02730    2.54e+03, 3.21e+03, 6.72e+03, 1.83e+04, 7.53e+04, 5.27e+04, 4.24e+04,
02731    9.35e+03, 1.26e+04, 1.59e+04, 1.39e+05, 5.27e+04, 3.37e+04, 4.78e+04,
02732    2.13e+04, 1.25e+04, 1.09e+04, 2.97e+03, 945, 1.65e+03, 1.86e+03, 1.45e+03,
02733    433, 681, 534, 1.04e+03, 1e+03, 928, 3.25e+03, 9.01e+03, 6.65e+03,
02734    2.86e+03, 3.14e+03, 2.39e+03, 3.68e+03, 1.68e+03, 1.46e+03, 4.99e+03,
02735    3.33e+03, 3.97e+03, 7.74e+03, 6.85e+03, 732, 5.31e+03, 4.32e+03, 6.3e+03,
02736    0, 0, 0, 0, 7.8e+03, 1.31e+05, 1.64e+05, 8.59e+04, 4.99e+04, 6.47e+03,
02737    2.39e+04, 1.13e+04, 7.16e+03, 5.18e+03, 5.1e+03, 7.19e+03, 2.81e+03,
```

```
02738    1.42e+03, 806, 347, 146, 287, 1.51e+03, 611, 721, 580, 1.22e+03, 558, 40.8,
02739    875, 508, 1.03e+03, 1.47e+03, 1.75e+03, 1.45e+03, 1.99e+03, 4.54e+03,
02740    3e+03, 7.34e+03, 2.49e+04, 0, 0, 0, 0, 0, 0, 0, 857, 9.34, 0, 0, 0, 0, 0,
02741    0, 0, 0, 0, 0},
02742   {132, 77.6, 122, 314, 96.7, 134, 212, 312, 3.22e+03, 5.23e+03, 485,
02743    1.01e+03, 858, 1.52e+03, 864, 8.54e+03, 5.71e+03, 7.82e+04, 7.78e+04,
02744    8.47e+04, 1.27e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02745    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 32.1, 9.27e+04,
02746    1.18e+05, 2.18e+05, 4.04e+05, 1.48e+04, 4.41e+03, 9.76e+03, 2.32e+04,
02747    3.01e+04, 2.04e+04, 5.15e+03, 1.91e+04, 7.98e+03, 5.04e+03, 2.49e+04,
02748    4.32e+04, 8.7e+04, 5.19e+04, 4.41e+03, 2.53e+03, 4.33e+03, 6.14e+03,
02749    2.33e+04, 1.75e+04, 6.85e+04, 6.08e+04, 4.36e+04, 1.47e+05, 4.24e+05,
02750    4.22e+05, 2.29e+05, 1.53e+05, 2.38e+05, 9.79e+03, 1.88e+03, 3.86e+03, 972,
02751    2.79e+03, 2.08e+03, 600, 1.02e+03, 118, 108, 3.22e+03, 4.97e+03, 686,
02752    1.11e+03, 2.79e+03, 2.98e+03, 2.84e+03, 1.77e+03, 3.01e+03, 854, 634, 401,
02753    216, 1.22e+03, 4.77e+03, 1.39e+04, 3.45e+03, 1.93e+03, 1.43e+03, 4.06e+03,
02754    1.66e+03, 0, 0, 0, 0, 3.05e+04, 3.22e+05, 1.34e+05, 7.58e+04, 1.79e+05,
02755    2e+04, 0, 16.6, 272, 2.27e+03, 2.51e+03, 3.97e+03, 3.49e+03, 938, 914, 417,
02756    191, 132, 293, 529, 224, 2.77e+03, 1.43e+03, 527, 2.46e+03, 1.15e+03, 588,
02757    1.74e+03, 1.04e+03, 848, 911, 3.44e+03, 5.44e+03, 2.22e+03, 1.25e+04,
02758    2.45e+04, 0, 0, 0, 0, 0, 0, 0, 97.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02759   {145, 58.8, 147, 455, 114, 152, 354, 125, 3e+03, 4.69e+03, 332, 1.21e+03,
02760    984, 1.44e+03, 1.66e+03, 1.29e+04, 8.03e+03, 6.05e+04, 6.38e+04, 3.8e+04,
02761    2.02e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02762    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 559, 1.57e+05, 1.28e+05,
02763    9.51e+04, 6.28e+04, 1.82e+04, 1.21e+04, 1.06e+05, 9.68e+04, 5.2e+04,
02764    4.33e+03, 2.56e+04, 2.78e+04, 1.09e+04, 1.53e+04, 1.07e+04, 5.18e+04,
02765    4.7e+04, 1.07e+05, 1.2e+04, 530, 936, 8.51e+03, 2.93e+04, 1.98e+05,
02766    2.34e+05, 8.81e+04, 6.58e+04, 1.24e+05, 4.26e+04, 6.74e+04, 1.24e+05,
02767    4.08e+05, 2.12e+05, 7.89e+04, 2.49e+04, 4.43e+03, 7.78e+03, 1.87e+03, 252,
02768    59.4, 151, 318, 1.36e+04, 4.62e+03, 2.59e+03, 2.39e+03, 2.88e+03, 2.43e+03,
02769    5.12e+03, 1.21e+03, 537, 1.38e+03, 1.91e+03, 3.09e+03, 556, 3.83e+03,
02770    1.05e+04, 1.25e+04, 3.01e+03, 800, 5.88e+03, 5.38e+03, 764, 0, 0, 0, 0,
02771    1.39e+05, 2.04e+05, 7.1e+04, 3.87e+04, 1.18e+05, 7.4e+04, 0, 0, 0, 8.84,
02772    1.38e+03, 2.88e+03, 1.54e+03, 558, 531, 459, 188, 138, 462, 713, 673,
02773    2.61e+03, 671, 385, 2.18e+03, 905, 652, 1.58e+03, 1.39e+03, 1.56e+03,
02774    1.94e+03, 5.92e+03, 9.39e+03, 4.59e+03, 1.9e+04, 1.22e+04, 0, 0, 0, 0, 0,
02775    0, 0, 36.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02776   {157, 44.6, 172, 639, 73.8, 73.5, 191, 119, 3.24e+03, 3.71e+03, 258,
02777    1.25e+03, 1.02e+03, 787, 2.38e+03, 1.21e+04, 9.72e+03, 6.06e+04, 4.99e+04,
02778    2.78e+04, 4.62e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02779    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 112, 5.42e+04,
02780    1.37e+05, 6e+04, 8.3e+04, 1.38e+05, 1.94e+05, 1.9e+04, 1.25e+04, 9.36e+03,
02781    1.38e+04, 3.39e+04, 7.5e+03, 8.57e+03, 2.07e+04, 3.36e+04, 2.02e+04,
02782    3.19e+04, 1.25e+05, 1.66e+04, 3.6e+03, 1.11e+04, 8.51e+04, 1.99e+05,
02783    5.19e+04, 2.91e+04, 6.48e+03, 8.59e+03, 1.52e+04, 2.03e+04, 1.82e+04,
02784    6.93e+03, 5.12e+04, 2.77e+05, 7.3e+04, 3e+04, 1.64e+04, 1.04e+03, 344,
02785    26.8, 565, 2.1e+03, 1.05e+04, 1.89e+04, 3.65e+03, 3e+03, 3.7e+03, 4.79e+03,
02786    7.36e+03, 1.28e+03, 1.43e+03, 1.62e+03, 4.94e+03, 3.39e+03, 1.66e+03,
02787    7.71e+03, 2.34e+04, 7.76e+03, 8.68e+03, 9.53e+03, 9.43e+03, 6.06e+03, 44.4,
02788    0.741, 0, 0, 0, 0, 2.94e+05, 1.37e+05, 5.1e+04, 5.59e+04, 2.19e+05, 0, 0,
02789    0, 0, 42.4, 1.55e+03, 1.67e+03, 973, 1.26e+03, 149, 209, 251, 443, 522,
02790    658, 1.45e+03, 913, 150, 558, 26.7, 4.31e+03, 902, 1.17e+03, 1.44e+03,
02791    2.42e+03, 1.51e+04, 1.13e+04, 6.04e+03, 9.39e+03, 205, 0, 0, 0, 0, 0, 0, 0,
02792    0, 0.165, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02793   {172, 35.6, 172, 783, 147, 102, 183, 128, 3.84e+03, 3.51e+03, 481, 838,
02794    1.1e+03, 317, 2.3e+03, 9.48e+03, 1.13e+04, 2.12e+04, 6.83e+04, 2.95e+04,
02795    7.71e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02796    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.06e+04, 4.03e+04,
02797    3.1e+04, 2.51e+04, 9.26e+03, 7.32e+03, 6.95e+03, 8.6e+03, 9.81e+03,
02798    3.72e+04, 3.92e+04, 3.87e+04, 2.8e+04, 1.79e+04, 3.53e+04, 1.64e+04,
02799    1.13e+04, 7.24e+04, 1.12e+04, 8.82e+03, 5.27e+04, 1.94e+05, 3.37e+04,
02800    7.26e+03, 2.04e+03, 2.85e+03, 4.18e+03, 4.11e+03, 1.12e+03, 1.95e+03,
02801    8.34e+03, 1.81e+04, 1.57e+04, 3.88e+04, 1.84e+04, 51.1, 747, 829, 1.94e+03,
02802    1.77e+03, 1.32e+03, 5.12e+03, 276, 187, 619, 1.74e+03, 2.69e+03, 2.91e+03,
02803    982, 750, 5.65e+03, 1.21e+04, 2.49e+03, 6.18e+03, 8.73e+03, 1.77e+04,
02804    2.32e+04, 2.28e+04, 8.08e+04, 3.61e+04, 159, 0.0676, 0, 0, 4.52e+03,
02805    3.26e+04, 4.53e+05, 6.64e+04, 2.05e+04, 6.24e+04, 2.52e+05, 5.17e+03, 0, 0,
02806    0, 10.8, 1.07e+03, 2.64e+03, 1.97e+03, 287, 131, 317, 316, 236, 636, 357,
02807    3.7e+03, 624, 421, 388, 296, 2e+03, 1.12e+03, 1.82e+03, 778, 844, 9.09e+03,
02808    2.13e+04, 4.88e+03, 6.66e+03, 0, 0, 0, 0, 0, 0, 0, 0, 1.27e+03, 0, 0, 0,
02809    0, 0, 0, 0, 0, 0, 0},
02810   {186, 27.5, 144, 830, 251, 125, 158, 137, 4.51e+03, 3.51e+03, 958, 686,
02811    1.06e+03, 147, 2.03e+03, 7.62e+03, 1.18e+04, 2.19e+04, 5.53e+04, 2.86e+04,
02812    8.74e+04, 2.88e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02813    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 99.8,
02814    678, 262, 544, 545, 602, 1.6e+03, 1.58e+04, 2.21e+05, 1.16e+05, 1.78e+05,
02815    3.62e+04, 1.32e+04, 4.08e+04, 2.36e+04, 9.72e+03, 1.73e+04, 1.23e+04,
02816    3.07e+04, 6.73e+04, 5.29e+04, 1.26e+04, 7.11e+03, 1.88e+03, 1.86e+03,
02817    1.39e+03, 658, 3.55e-08, 1.31e+03, 1.83e+03, 874, 5.12e+03, 1.59e+05,
02818    3.43e+04, 202, 2.45e+03, 118, 2.07e+03, 251, 1.15e+03, 1.05e+03, 891, 535,
02819    181, 1.97e+03, 1.7e+03, 2.55e+03, 2.89e+03, 1.28e+03, 9.67e+03, 7.92e+03,
02820    4.77e+03, 5.76e+03, 5.37e+03, 1.88e+04, 4.7e+04, 7.42e+04, 6.97e+04,
02821    5.44e+04, 1.65e+04, 0, 0, 0, 9.77e+04, 3.57e+05, 1.81e+05, 6.69e+03,
02822    1.29e+04, 4.51e+04, 1.08e+05, 2.04e+05, 0, 0, 731, 406, 742, 2.07e+03,
02823    1.64e+03, 601, 406, 492, 567, 442, 626, 756, 518, 3.04e+03, 1.95e+03, 282,
02824    1.78e+03, 877, 1.38e+03, 786, 438, 785, 1.39e+04, 1.98e+04, 5.49e+03,
```

```
02825    1.62e+03, 0, 0, 0, 0, 0, 0, 0, 0, 1.59e+03, 0, 0, 0, 0, 0, 0, 0, 0,
02826    0},
02827   {204, 18.4, 122, 897, 302, 119, 155, 325, 4.53e+03, 3e+03, 925, 560, 906,
02828    75.2, 1.87e+03, 6.55e+03, 1.22e+04, 2.69e+04, 5.2e+04, 2.21e+04, 2.46e+04,
02829    5.14e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02830    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.21e+03,
02831    853, 1.4e+03, 486, 2.59e+03, 4.36e+03, 3.54e+04, 3.44e+04, 1.93e+04,
02832    3.22e+04, 6.04e+04, 8.53e+04, 8.76e+04, 2.04e+05, 2.69e+05, 1.98e+05,
02833    1.87e+05, 2.57e+04, 1.15e+04, 9.94e+03, 1.07e+04, 1.35e+04, 7.25e+03,
02834    1.33e+04, 1.51e+04, 2.99e+03, 1.42e+05, 6.81e+03, 1.88e+04, 1.32e+05,
02835    1.45e+04, 2.64e+04, 7.84e+03, 6.75e+03, 1.02e+05, 6.84e+04, 1.43e+04,
02836    2.69e+03, 353, 391, 378, 1.63e+03, 264, 485, 540, 2.08e+03, 9.91e+03,
02837    6.32e+03, 8.47e+03, 3.83e+03, 1.21e+04, 4.85e+04, 5.21e+04, 2.14e+05,
02838    9.65e+04, 2.28e+04, 1.7e+04, 5.4, 0, 0, 5.21e+03, 4e+04, 4.81e+05,
02839    7.08e+04, 2.71e+04, 4.7e+04, 1.26e+05, 2.94e+05, 0, 1.05e+04, 6.91e+04,
02840    382, 810, 1.84e+03, 1.36e+03, 891, 413, 592, 647, 412, 852, 492, 732, 826,
02841    1.46e+03, 1.83e+03, 2.27e+03, 3.84e+03, 1.53e+03, 742, 1.03e+03, 1.1e+03,
02842    3.24e+03, 3.12e+04, 8.42e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.98, 0, 0, 0,
02843    0, 0, 0, 0, 0, 0, 0, 0},
02844   {222, 10, 136, 1.05e+03, 241, 92.1, 223, 410, 4.09e+03, 2.49e+03, 615, 468,
02845    723, 57.4, 1.73e+03, 5.35e+03, 1.11e+04, 2.23e+04, 5.56e+04, 1.42e+04,
02846    1.78e+05, 1.26e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02847    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02848    479, 3.31e+03, 1.44e+03, 2.45e+03, 1.07e+03, 91.5, 4.82e+03, 7.9e+03,
02849    1.27e+04, 9.76e+04, 9.88e+04, 6.81e+04, 1.07e+05, 5.29e+04, 1.04e+05,
02850    9.7e+04, 9.23e+04, 9.44e+04, 1.68e+04, 3.5e+04, 2.48e+04, 1.1e+05,
02851    4.61e+04, 4.02e+04, 4.64e+04, 1.69e+05, 2.51e+05, 1.08e+05, 1.2e+05,
02852    1.32e+04, 4.31e+04, 6.32e+04, 1.73e+05, 3.13e+05, 5.78e+04, 1.07e+05,
02853    2.71e+04, 3.45e+03, 819, 2.41e+03, 779, 598, 1.39e+03, 1.06e+03, 2.43e+03,
02854    4.52e+03, 7.89e+03, 3.45e+04, 3.05e+04, 6.04e+04, 4.57e+04, 552, 0,
02855    5.49e+04, 2.72e+05, 1.14e+05, 6.09e+04, 6.82e+04, 3.61e+04, 2.14e+03, 0,
02856    0.694, 4.05e+05, 2.19e+05, 5.82e+04, 3.28e+04, 1.13e+05, 1.79e+05, 0, 0,
02857    3e+04, 68.1, 1.19e+03, 1.93e+03, 830, 1e+03, 892, 843, 647, 640, 745, 702,
02858    574, 345, 269, 1.1e+03, 4.15e+03, 766, 2.31e+03, 1.92e+03, 74.2, 81.9,
02859    4.23e+03, 2e+04, 9.89e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02860    0, 0, 0, 0, 0},
02861   {242, 6.99, 133, 1.28e+03, 426, 207, 410, 520, 3.2e+03, 2.18e+03, 4.66e+03,
02862    458, 770, 120, 1.68e+03, 4.08e+03, 9.42e+03, 9.73e+03, 4.36e+04, 2.04e+04,
02863    2.61e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02864    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02865    0.00332, 17.3, 11.4, 0, 0, 120, 3.02e+03, 1.26e+04, 8.4e+04, 1.02e+04,
02866    1.17e+04, 5.21e+04, 2.25e+04, 3.35e+04, 1.83e+04, 9.23e+04, 3.55e+05,
02867    9.97e+04, 2.97e+04, 2.4e+04, 4.96e+04, 2.48e+05, 3.22e+05, 1.66e+05,
02868    1.94e+05, 2.21e+05, 6.43e+04, 1.64e+04, 2.66e+04, 3.65e+04, 1.22e+05,
02869    2.19e+05, 7.45e+04, 8.54e+04, 1.63e+05, 1.5e+05, 6.43e+04, 1.79e+04,
02870    1.83e+03, 1.14e+03, 759, 2.88e+03, 5.3e+03, 2.28e+04, 1.55e+04, 1.4e+04,
02871    6.43e+04, 3.36e+03, 66.5, 0, 135, 2.38e+04, 1.48e+05, 4.95e+04, 3.14e+04,
02872    7.88e+04, 9.03e+04, 1.17e+05, 4.36e+05, 1.11e+05, 1.03e+05, 1.55e+05,
02873    2.33e+05, 6.15e+04, 9.66e+04, 1.3e+05, 8.81e+03, 0, 0, 1.02e+03, 0,
02874    4.31e+03, 891, 1.2e+03, 951, 944, 850, 631, 986, 789, 1.47e+03, 422, 353,
02875    254, 1.21e+03, 6.08e+03, 1.61e+03, 1.77e+03, 4.02e+03, 43.3, 846, 2.95e+03,
02876    1.59e+03, 1.13e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02877    0, 0, 0},
02878   {264, 11.6, 93, 1.52e+03, 772, 145, 204, 652, 3.47e+03, 1.62e+03, 438, 457,
02879    958, 243, 1.77e+03, 3.55e+03, 6.24e+03, 6.63e+03, 3.28e+04, 2.76e+04,
02880    2.9e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02881    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02882    0, 0, 0, 0, 75.5, 5.88e+03, 3.68e+04, 6.55e+04, 1.99e+04, 1.24e+04,
02883    1.57e+04, 2.2e+04, 1.95e+04, 5.73e+04, 2.08e+05, 1.27e+05, 9.54e+04,
02884    1.98e+04, 3.99e+04, 4.32e+05, 4.65e+04, 1.98e+05, 3.36e+05, 1.5e+05,
02885    1.86e+05, 4.04e+04, 2.81e+04, 1.29e+05, 3.07e+05, 2.57e+05, 9.56e+04,
02886    1.14e+05, 1.91e+05, 1.39e+05, 9.25e+04, 2.14e+05, 1.22e+05, 2.68e+04,
02887    1.49e+04, 1.85e+04, 9.28e+04, 1.51e+05, 1.44e+05, 1.13e+05, 3.76e+04, 0,
02888    0.0899, 8.1e+03, 1.08e+05, 2.08e+05, 5.49e+04, 1.57e+04, 1.05e+04,
02889    1.67e+04, 3.09e+04, 6.32e+04, 9.51e+04, 2.92e+04, 1.36e+04, 7.66e+04,
02890    2.17e+05, 3.91e+04, 7.45e+04, 2.72e+05, 0, 0, 0, 1.01e+03, 0, 6.28e+03,
02891    1.14e+03, 1.18e+03, 1.21e+03, 1.14e+03, 673, 774, 836, 807, 767, 212, 617,
02892    411, 1.21e+03, 2.51e+03, 2.49e+03, 1.02e+03, 3.42e+03, 3.22e+03, 1.67e+03,
02893    691, 1.54e+03, 9.79e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.28, 0, 0, 0, 0, 0,
02894    0, 0, 0, 0, 0},
02895   {285, 26.4, 85.1, 1.71e+03, 816, 122, 297, 463, 3.56e+03, 1.1e+03, 269, 449,
02896    1.11e+03, 391, 1.87e+03, 3.39e+03, 4.69e+03, 1.21e+04, 3.68e+04, 4.84e+04,
02897    2.19e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02898    5.85e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02899    0, 0, 0, 0, 0, 0, 0.578, 2.81e+03, 2.25e+04, 1.59e+04, 7.28e+03, 8.2e+03,
02900    1.03e+04, 1.55e+04, 1.86e+04, 1.32e+04, 1.6e+04, 1.22e+05, 5.91e+04,
02901    1.17e+05, 5.95e+04, 1.02e+05, 4.75e+04, 5.88e+04, 2.26e+05, 4.07e+04,
02902    3.85e+04, 3.09e+04, 1.2e+05, 7.58e+04, 2.31e+05, 9.6e+04, 2.3e+05,
02903    1.46e+05, 2.23e+05, 2.46e+05, 1.65e+05, 4.57e+05, 1.33e+05, 6.48e+04,
02904    1.85e+05, 2.57e+05, 2.51e+05, 52.6, 0, 0, 0, 0, 5.6e+03, 9.29e+04,
02905    5.76e+04, 3.23e+04, 3.83e+03, 6.63e+03, 1.31e+04, 1.01e+04, 4.63e+03,
02906    3.48e+03, 2.03e+04, 5.04e+04, 3.45e+03, 1.41e+04, 2.37e+05, 1.42e+05,
02907    1.27e+05, 5.15e+05, 0, 0, 712, 1.86e+04, 84, 1.65e+03, 2.17e+03, 2.02e+03,
02908    1.18e+03, 1.19e+03, 650, 898, 583, 1.17e+03, 1.17e+03, 938, 428, 768, 523,
02909    712, 1.15e+03, 1.26e+03, 2.24e+03, 1.78e+03, 4.81e+03, 796, 3.49e+03,
02910    2.68e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02911   {306, 52.1, 83.6, 1.9e+03, 674, 242, 342, 651, 3.21e+03, 724, 218, 431,
```

```
02912    1.17e+03, 556, 1.82e+03, 3.19e+03, 3.85e+03, 1.54e+04, 3.87e+04, 7.58e+04,
02913    1.43e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02914    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02915    0, 0, 0, 0, 0, 517, 1.16e+04, 1.39e+04, 6.59e+03, 6.76e+03, 1.22e+04,
02916    3.35e+04, 1.96e+04, 9.76e+03, 1.31e+04, 6.64e+03, 5.29e+03, 5.58e+04,
02917    3.35e+04, 1.59e+04, 1.06e+04, 1.39e+04, 8.58e+03, 4.81e+03, 1.72e+03,
02918    1.54e+04, 3.54e+04, 5.08e+04, 1.25e+05, 1.75e+05, 3.35e+05, 1.65e+05,
02919    2.27e+05, 3.31e+05, 2.72e+05, 1.88e+05, 1.06e+05, 2.09e+05, 4.83e+05,
02920    1.22e+05, 1.29e+03, 0, 0, 569, 3.88e+03, 1.87e+04, 1.74e+05, 5.95e+04,
02921    6.55e+04, 6.64e+03, 3.09e+03, 3.22e+03, 5.49e+03, 7.01e+03, 1.08e+03,
02922    1.9e+03, 8.04e+03, 2.13e+04, 3.43e+03, 1.97e+04, 1.66e+05, 2.3e+05,
02923    1.65e+05, 6.59e+05, 0, 0, 4.44e+04, 1.07e+04, 233, 924, 6.06e+03, 1.86e+03,
02924    1.33e+03, 1.44e+03, 1.39e+03, 660, 514, 699, 720, 1.15e+03, 501, 801, 484,
02925    535, 626, 3.12e+03, 3.01e+03, 891, 5.01e+03, 1.02e+03, 1.15e+04, 0, 0, 0,
02926    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02927   {326, 88.9, 116, 1.95e+03, 598, 352, 872, 223, 2.74e+03, 686, 179, 334, 998,
02928    770, 1.57e+03, 2.66e+03, 3.65e+03, 1.51e+04, 3.42e+04, 9.28e+04, 9.31e+04,
02929    1.15e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02930    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02931    0, 0, 0, 0, 25.2, 3.14e+03, 8.56e+03, 7.74e+03, 1.48e+04, 3.17e+04,
02932    2.23e+04, 5.84e+03, 2.76, 0.0273, 1.74, 0.0121, 0.482, 3.84e+03, 4.63e+03,
02933    1.6e+03, 1.54e+03, 1.22e+03, 1.45e+03, 6.54e+03, 1.85e+04, 2.7e+04,
02934    5.73e+04, 4.54e+04, 5.97e+05, 2.81e+05, 2.36e+05, 7.98e+05, 5.57e+05,
02935    2.65e+05, 5.48e+05, 9.43e+05, 3.95e+05, 40.7, 0, 0, 0, 2.87e+03, 3.49e+05,
02936    1.53e+05, 8.45e+04, 2.76e+04, 2.91e+04, 3.25e+04, 8.88e+03, 2.94e+03,
02937    5.91e+03, 4.92e+03, 1.13e+03, 2.72e+03, 3.97e+03, 9.78e+03, 6.66e+03,
02938    1.24e+04, 2.8e+04, 5.07e+04, 1.66e+05, 1.32e+05, 8.77e+05, 0, 1.17e+05,
02939    4.09e+05, 8.62e+03, 359, 976, 4.06e+03, 1.81e+03, 1.8e+03, 1.17e+03, 356,
02940    463, 479, 256, 773, 437, 582, 813, 686, 691, 2.49e+03, 811, 916, 2.31e+03,
02941    2.84e+03, 1.31e+03, 1.3e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02942    0, 0, 0, 0, 0, 0, 0},
02943   {350, 132, 190, 1.92e+03, 578, 424, 843, 189, 1.95e+03, 891, 172, 130, 613,
02944    483, 1.24e+03, 1.98e+03, 3.9e+03, 1.32e+04, 3.07e+04, 5.2e+04, 2.61e+04,
02945    1.69e+05, 63.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02946    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02947    0, 0, 0, 0, 0, 0, 0, 0.00111, 712, 171, 12, 0.00332, 0.0492, 0, 0, 0, 0, 0, 0,
02948    0, 0.277, 645, 909, 763, 1.7e+03, 4.29e+03, 8.52e+03, 2.45e+04, 6.43e+04,
02949    5.45e+04, 9.87e+04, 4.82e+04, 1.16e+05, 3.04e+04, 2.23e+04, 2.12e+04,
02950    4.99e+04, 6.91e+04, 2.64e+03, 0.361, 0, 3.36, 2.51e+03, 4.33e+05, 1.21e+05,
02951    1.57e+04, 7.12e+03, 5.44e+03, 8.19e+03, 4.39e+03, 8.11e+03, 4.97e+03,
02952    2.15e+03, 2.8e+03, 5.91e+03, 4.35e+03, 4.18e+03, 8.98e+03, 1.03e+03,
02953    6.32e+03, 6.38e+04, 4.7e+04, 1.92e+05, 1.11e+05, 1.16e+06, 0, 7.76e+05,
02954    4.7e+05, 1.12e+04, 799, 1.27e+03, 2.28e+03, 1.42e+03, 1.86e+03, 1.42e+03,
02955    323, 367, 522, 366, 492, 346, 829, 794, 645, 491, 2.16e+03, 1.87e+03, 448,
02956    2.41e+03, 2.57e+03, 7.01e+03, 2.23e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02957    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02958   {371, 177, 292, 1.89e+03, 606, 825, 489, 199, 1.61e+03, 1.16e+03, 507, 14.5,
02959    380, 160, 808, 1.61e+03, 5.09e+03, 1.02e+04, 2.7e+04, 1.98e+04, 1.8e+04,
02960    4.14e+04, 2.1e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02961    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02962    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.01, 77.7,
02963    91.9, 1.88e+03, 4.45e+03, 1.68e+04, 6.13e+03, 6.92e+03, 4.1e+04, 2.56e+04,
02964    3.79e+04, 1.25e+05, 1.61e+05, 3.43e+04, 6.52e+04, 8.28e+04, 4.25e+04, 0,
02965    0.188, 27.5, 3.65e+05, 6.42e+05, 1.24e+05, 1.35e+04, 5.28e+03, 858,
02966    1.07e+03, 3.61e+03, 3.63e+03, 7.46e+03, 9.08e+03, 2.91e+03, 2.59e+03,
02967    2.73e+03, 3.45e+03, 3.96e+03, 7.3e+03, 1.63e+03, 2.29e+04, 4e+04, 2.59e+05,
02968    7.86e+04, 8.72e+04, 3.24e+05, 4.47e+05, 1.21e+06, 3.51e+05, 1.48e+04,
02969    1.6e+03, 1.67e+03, 1.55e+03, 1.26e+03, 2.07e+03, 1.21e+03, 676, 732, 560,
02970    524, 521, 467, 803, 1.1e+03, 846, 1.72e+03, 3.11e+03, 865, 481, 678,
02971    1.08e+03, 3.16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02972    0, 0, 0},
02973   {392, 220, 387, 1.84e+03, 635, 1.4e+03, 422, 175, 1.64e+03, 1.74e+03, 263,
02974    149, 173, 130, 421, 1.52e+03, 6.44e+03, 7.91e+03, 2.14e+04, 1.84e+04,
02975    9.02e+03, 4.03e+04, 1.74e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02976    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02977    0, 0, 0, 0, 467, 335, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.63e+04, 3.81e+03, 0, 0,
02978    0, 0, 0, 0, 0, 0, 0, 0, 209, 233, 4.89e+03, 6.56e+03, 6.34e+03, 7.61e+03,
02979    9.87e+03, 3.23e+04, 2.1e+04, 3.55e+04, 1.06e+05, 7.65e+04, 1.29e+05,
02980    1.32e+04, 4.12e+03, 7.58e+04, 1.22e+05, 5.84e+05, 4.14e+05, 4.04e+04,
02981    2e+04, 6.35e+03, 1.15e+03, 1.85e+03, 1.86e+03, 8.11e+03, 7.06e+03,
02982    5.31e+03, 986, 3.23e+03, 2.93e+03, 950, 2.73e+03, 4.72e+03, 2.66e+03,
02983    2.32e+03, 3.68e+03, 3.14e+05, 2.63e+05, 1.4e+05, 1.1e+05, 9.47e+04,
02984    4.35e+05, 7.3e+05, 3.26e+05, 9.29e+03, 3.37e+03, 1.51e+03, 1.12e+03,
02985    2.24e+03, 1.08e+03, 594, 634, 940, 1.67e+03, 828, 984, 351, 741, 478, 617,
02986    1.18e+03, 3.22e+03, 2.23e+03, 1.16e+03, 1.97e+03, 1.94e+03, 1.05, 0, 207,
02987    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
02988   {420, 269, 465, 1.78e+03, 621, 1.55e+03, 395, 173, 1.94e+03, 1.9e+03, 193,
02989    483, 74.2, 100, 212, 1.74e+03, 6.02e+03, 6.46e+03, 1.58e+04, 1.13e+04,
02990    4.45e+03, 4.54e+04, 2.84e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02991    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02992    0, 0, 7.02e+03, 1.45e+04, 5.94e+04, 4.09e+04, 9.37e+03, 70.2, 283,
02993    3.34e+03, 824, 0.462, 0, 0, 0, 7.3e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
02994    785, 1.58e+03, 1.85e+04, 4.58e+03, 1.71e+04, 1.15e+04, 2.92e+04, 1.59e+04,
02995    1.26e+04, 2.21e+05, 2.32e+03, 0, 2.5e+05, 4.55e+05, 3.88e+05, 2.95e+05,
02996    6.3e+04, 1.14e+05, 3.58e+04, 1.3e+04, 5.93e+03, 4.39e+03, 5.52e+03,
02997    3.61e+03, 4.89e+03, 2.22e+03, 2.78e+03, 2.16e+03, 3.1e+03, 1.35e+03,
02998    4.47e+03, 4.73e+03, 178, 347, 3.51e+03, 4.97e+04, 4.36e+05, 1.62e+05,
```

```
02999    1.12e+05, 3.42e+05, 2.15e+05, 2.85e+05, 1.05e+06, 6.12e+03, 2.46e+03,
03000    2.17e+03, 1.01e+03, 689, 1.68e+03, 1.26e+03, 954, 791, 895, 1.22e+03, 768,
03001    1.65e+03, 526, 369, 517, 669, 675, 3.24e+03, 2.7e+03, 2.37e+03, 270, 340,
03002    151, 2.21e+03, 1.38e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03003    0, 0, 0, 0},
03004   {442, 341, 504, 1.74e+03, 599, 1.45e+03, 346, 315, 2.24e+03, 1.59e+03, 171,
03005    856, 35.5, 47.6, 218, 2.03e+03, 4.19e+03, 5.7e+03, 7.71e+03, 7.26e+03,
03006    1.48e+03, 4.1e+04, 2.63e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03007    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03008    0, 621, 7.54e+03, 1.33e+04, 4.05e+04, 3.86e+04, 1.97e+04, 1.22e+04,
03009    4.44e+04, 3.94e+04, 2.09e+04, 919, 0, 0, 2.16e+03, 223, 0, 0, 0, 0, 0, 0,
03010    0, 0, 0, 0, 0, 0, 0, 1.9e+03, 4.69e+03, 8.61e+03, 4.8e+03, 5.44e+03,
03011    1.37e+04, 8.01e+03, 7.46e+03, 2.42e+05, 699, 0, 6.05e+04, 2.67e+05,
03012    1.54e+05, 1.14e+05, 7.81e+04, 5.5e+03, 7.9e+03, 5.97e+03, 5.8e+03,
03013    9.52e+03, 8.75e+03, 8.93e+03, 2.13e+03, 2.84e+03, 2.4e+03, 1.63e+03,
03014    1.3e+03, 2.97e+03, 4.91e+03, 43.6, 48.3, 4.43e+03, 7.5e+04, 2.01e+05,
03015    2.71e+05, 1.56e+05, 2.22e+05, 3.98e+05, 4.87e+05, 3.82e+05, 9.9e+05,
03016    9.13e+03, 1.81e+03, 1.96e+03, 705, 392, 1.95e+03, 3.36e+03, 2.45e+03,
03017    1.05e+03, 1.21e+03, 1.23e+03, 957, 1.51e+03, 398, 453, 533, 521, 446,
03018    2.83e+03, 843, 1.61e+03, 786, 429, 179, 3.84e+03, 197, 0, 0, 0, 0, 0, 0, 0,
03019    0, 0, 0, 0, 874, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03020   {466, 452, 485, 1.69e+03, 638, 1.34e+03, 298, 636, 2.22e+03, 1.11e+03, 171,
03021    1.13e+03, 79.6, 52.8, 228, 1.96e+03, 2.7e+03, 6.53e+03, 6.62e+03, 3.7e+03,
03022    3.17e+03, 6.78e+04, 1.96e+05, 1.76e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03023    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03024    0, 0, 0, 48, 1.45e+04, 1.03e+05, 4.65e+04, 2.21e+04, 7.87e+04, 1.18e+05,
03025    2.6e+04, 5.53e+04, 2.94e+04, 3.19e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03026    0, 0, 0, 0, 0, 814, 4.03e+03, 3.35e+03, 7.34e+03, 4.64e+03, 3.49e+03,
03027    4.69e+03, 9.58e+03, 7.11e+04, 3.31e+04, 0, 4.41e+03, 3.19e+05, 6.21e+04,
03028    6.33e+03, 6.82e+03, 3.39e+03, 2.73e+03, 2.72e+03, 7.98e+03, 9.63e+03,
03029    1.84e+04, 1.34e+04, 6.44e+03, 2.74e+03, 3.64e+03, 746, 969, 3.62e+03, 566,
03030    9.53, 6.73e+04, 3.09e+05, 3.28e+05, 1.03e+05, 4.9e+04, 2.03e+05, 3.92e+05,
03031    2.67e+05, 2.49e+05, 1.04e+06, 7.89e+05, 967, 18.9, 0.0828, 6.53, 67.6,
03032    97.7, 696, 5.69e+03, 8.02e+03, 1.39e+03, 3.43e+03, 2.33e+03, 1.42e+03, 552,
03033    396, 559, 1.14e+03, 1.81e+03, 3.1e+03, 725, 2.63e+03, 475, 475, 221,
03034    3.88e+03, 6.02, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.37, 6.61e+03, 0, 0, 0, 0,
03035    0, 0, 0, 0, 0},
03036   {492, 614, 441, 1.55e+03, 742, 1.24e+03, 250, 931, 2.17e+03, 677, 160,
03037    1.34e+03, 123, 39.1, 214, 1.7e+03, 2.09e+03, 6.86e+03, 1.16e+04, 1.77e+03,
03038    7.36e+03, 6.3e+04, 1.41e+05, 1.17e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03039    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03040    0, 0, 0, 0, 8.78e+04, 2.16e+05, 7.98e+04, 1.48e+05, 6.27e+04, 7.39e+04,
03041    6.37e+04, 7.42e+04, 1.02e+05, 3.68e+03, 1.48e+03, 0, 0, 0, 0, 0, 0, 0, 0,
03042    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.8e+03, 2.62e+03, 53.9, 3.71e+03, 3.73e+03,
03043    1.3e+04, 8.75e+03, 1.09e+05, 4.23e+05, 0, 1.32, 2.52e+05, 6e+04, 1.23e+04,
03044    7.59e+03, 2.78e+03, 2.54e+03, 2.55e+03, 2.39e+03, 3.19e+03, 8.21e+03,
03045    2.59e+04, 3.88e+03, 4.75e+03, 2.9e+03, 608, 1.49e+03, 3.02e+03, 48.5, 696,
03046    3.14e+05, 8.8e+04, 9.27e+04, 5.54e+04, 1.2e+05, 1.05e+05, 4.06e+05,
03047    3.4e+05, 5.22e+05, 6.11e+05, 4.34e+05, 1.2, 0, 0, 0.618, 175, 277, 165,
03048    410, 3.79e+03, 3.33e+03, 3.31e+03, 1.79e+03, 1.73e+03, 406, 346, 468,
03049    1.09e+03, 1.58e+03, 1.77e+03, 814, 2.21e+03, 2.28e+03, 402, 260, 22.9, 0,
03050    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.97e+03, 5.61e+03, 0, 0, 0, 0, 0, 0, 0, 0,
03051    0},
03052   {517, 829, 396, 1.4e+03, 886, 1.14e+03, 248, 1.03e+03, 1.81e+03, 407, 162,
03053    1.31e+03, 218, 50.4, 343, 1.25e+03, 1.61e+03, 4.56e+03, 7.57e+03, 6.03e+03,
03054    1.47e+04, 5.28e+04, 1.93e+05, 296, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03055    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03056    0, 0, 0, 0, 27.6, 459, 1.6e+04, 8.78e+04, 2.43e+05, 8.9e+04, 3.68e+04,
03057    5.12e+04, 1.35e+05, 4.24e+04, 7.85e+04, 466, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03058    0, 0, 0, 0, 0, 0, 2.26e+03, 5e+03, 750, 6.83e+03, 1.11e+04, 2.28e+04,
03059    1.77e+04, 4.63e+05, 0, 0, 9.43e+04, 7.1e+04, 3.52e+04, 1.31e+04, 3.93e+03,
03060    1.34e+03, 1.54e+03, 1.25e+03, 1.97e+03, 3.81e+03, 2.67e+03, 3.67e+03,
03061    3.61e+03, 5.72e+03, 5.62e+03, 5.91e+03, 978, 19, 3.09e+03, 3.29e+05,
03062    1.5e+05, 6.62e+04, 3.05e+04, 1.2e+05, 2.8e+05, 2.98e+05, 1.46e+03,
03063    1.02e+06, 6.09e+05, 1.99e+04, 0, 0, 0, 0, 1.49, 275, 152, 464, 2.59e+03,
03064    5.26e+03, 3.07e+03, 1.61e+03, 1.76e+03, 508, 477, 913, 806, 1.02e+03,
03065    1.27e+03, 1.32e+03, 1.06e+03, 1.47e+03, 433, 1.83e+03, 612, 199, 0, 0, 0,
03066    0, 0, 0, 0, 0, 0, 0, 9.14e+03, 3.38e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03067   {540, 1.1e+03, 351, 1.26e+03, 1.06e+03, 1.06e+03, 344, 1.2e+03, 1.37e+03,
03068    261, 213, 1.06e+03, 373, 216, 859, 1.22e+03, 1.24e+03, 2.33e+03, 2.98e+03,
03069    1.93e+04, 2.03e+04, 3.5e+04, 1.29e+05, 2.12e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03070    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03071    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0442, 9.04e+03, 6.16e+04, 1.7e+05,
03072    1.2e+05, 4.06e+04, 5.35e+04, 2.92e+05, 4.64e+04, 6.48e+03, 0, 0, 0, 0, 0,
03073    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 177, 6.19e+03, 7.23e+03, 1.52e+04,
03074    2.75e+04, 1.77e+04, 1.94e+05, 0, 0, 3.99e+04, 2.59e+05, 1.48e+04, 1.37e+04,
03075    3.01e+03, 1.1e+03, 729, 611, 458, 4.02e+03, 3.18e+03, 2.57e+03, 4.99e+03,
03076    2.16e+03, 76, 0, 9.69, 869, 7.99e+04, 4.88e+05, 7.65e+04, 5.34e+04,
03077    6.14e+04, 6.39e+05, 6.83e+05, 7.69e+04, 2.42e+04, 2.75e+05, 2.05e+03, 0, 0,
03078    0, 0, 0, 0.834, 65.3, 55.6, 1.21e+03, 1.13e+03, 1.75e+03, 3.56e+03,
03079    2.11e+03, 1.64e+03, 910, 1.66e+03, 1.34e+03, 1.09e+03, 491, 944, 712,
03080    1.09e+03, 1.27e+03, 1.22e+03, 2.81e+03, 1.1e+03, 1.56e+03, 30.2, 0, 0, 0,
03081    0, 0, 0, 0, 0, 0, 8.41e+03, 3.05e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03082   {561, 1.42e+03, 313, 1.15e+03, 1.26e+03, 984, 543, 1.53e+03, 1.11e+03, 204,
03083    320, 732, 599, 857, 820, 566, 884, 970, 2.93e+03, 1.66e+04, 1.68e+04,
03084    1.68e+04, 7.52e+04, 6.5e+04, 840, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03085    0, 0, 0, 0, 0, 53.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
03086    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.06e+03, 3.3e+04, 8.07e+04, 1.05e+05,
03087    1.75e+04, 327, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 269,
03088    2.02e+04, 2.2e+04, 2.4e+04, 1.77e+05, 1.71e+04, 0, 0, 3.56e+05, 2.06e+04,
03089    1.56e+04, 7.08e+03, 2.35e+03, 629, 370, 297, 338, 657, 1.75e+03, 1.57e+03,
03090    75.9, 0, 0, 1.1e+04, 4.77e+05, 7.12e+05, 1.46e+05, 6.39e+04, 1.62e+05,
03091    9.57e+04, 8.32e+05, 0, 0, 829, 195, 0, 0, 121, 444, 0, 0, 0.37, 9.76, 93.6,
03092    2.24e+03, 1.06e+03, 3.24e+03, 1.48e+03, 1.52e+03, 1.04e+03, 1.05e+03,
03093    1.21e+03, 990, 766, 785, 858, 443, 1.74e+03, 3.43e+03, 1.12e+03, 2.41e+03,
03094    1.63e+03, 225, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.42e+03, 1.11e+04, 0, 0, 0,
03095    0, 0, 0, 0, 0, 0, 0},
03096   {584, 1.71e+03, 286, 1.04e+03, 1.48e+03, 924, 812, 1.79e+03, 817, 185, 474,
03097    454, 841, 1.17e+03, 1e+03, 589, 639, 321, 2.91e+03, 8.27e+03, 5.64e+03,
03098    2.82e+04, 3.87e+04, 9.65e+04, 3.61e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03099    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03100    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03101    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.66e+03, 1.88e+04, 6.59e+04, 6.42e+04,
03102    0, 0, 1.05e+05, 5.01e+04, 1.29e+04, 1.49e+04, 1.76e+03, 440, 496, 548,
03103    1.03e+03, 2.15e+03, 1.05e+03, 129, 0, 2.9e+03, 7.78e+04, 1.74e+05,
03104    3.93e+05, 1.54e+05, 8.4e+04, 3.04e+05, 3.11e+04, 6.53e+05, 7.06e+05, 0, 0,
03105    0, 0, 0, 0.506, 3.85e+03, 1.49e+03, 0, 0, 1.47, 12.3, 70, 1.05e+03,
03106    1.74e+03, 1.96e+03, 2.23e+03, 1.61e+03, 2.1e+03, 796, 701, 636, 642, 866,
03107    563, 393, 3.95e+03, 3.82e+03, 1.2e+03, 2.19e+03, 1.19e+03, 6.35, 0, 0,
03108    1.73, 0, 0, 0, 0, 0, 0, 0, 4.25e+03, 1.3e+04, 0, 0, 0, 0, 0, 0, 0, 0},
03109   {611, 1.93e+03, 274, 956, 1.69e+03, 879, 1.08e+03, 2.01e+03, 455, 171, 893,
03110    421, 1.25e+03, 1.67e+03, 1.28e+03, 661, 605, 67.3, 2.17e+03, 4.2e+03,
03111    7.52e+03, 1.21e+04, 6.62e+04, 2.75e+04, 1.22e+05, 0, 0, 0, 0, 0, 0, 0, 0,
03112    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.23e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03113    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03114    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 124, 0, 0,
03115    795, 5.14e+04, 4.54e+04, 1.33e+04, 2.34e+03, 457, 391, 214, 563, 391, 2.98,
03116    0.916, 0, 9.16e+04, 2.77e+05, 2.66e+05, 7.44e+04, 1.35e+05, 1.28e+05,
03117    1.94e+05, 5.46e+04, 7.36e+05, 5.52e+04, 0, 0, 0, 0, 11.7, 2.31e+03,
03118    9.8e+03, 5.4, 0, 0, 9.32, 40.8, 247, 873, 2.24e+03, 2.25e+03, 3.7e+03,
03119    3.48e+03, 2.13e+03, 887, 921, 829, 441, 778, 802, 185, 3.39e+03, 3.83e+03,
03120    2.03e+03, 1.82e+03, 1.14e+03, 0.372, 0, 28.6, 2.07e+03, 1.26, 0, 0, 0, 0,
03121    0, 0, 60, 347, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03122   {632, 2.09e+03, 271, 887, 1.85e+03, 858, 1.28e+03, 2.19e+03, 249, 167,
03123    1.8e+03, 891, 1.64e+03, 2.24e+03, 1.33e+03, 729, 498, 289, 675, 6.14e+03,
03124    8.13e+03, 5.66e+03, 7.75e+04, 2.91e+04, 1.88e+05, 0, 0, 0, 0, 0, 0, 0, 0,
03125    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03126    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03127    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 393, 1.33e+03, 0,
03128    0, 0.00444, 1.27e+05, 9.87e+03, 2.92e+03, 1.02e+03, 453, 164, 410,
03129    2.26e+03, 0, 0, 6.69e+04, 7.63e+04, 1.75e+05, 1.2e+05, 1.92e+05, 4.19e+04,
03130    1.88e+05, 5.94e+04, 1.02e+05, 7.73e+05, 1.18e+04, 0, 0.00444, 3.81e+03,
03131    40.9, 4.29e+03, 4.56e+03, 8.93e+03, 46.2, 0, 0, 17.7, 85, 631, 1.15e+03,
03132    3.1e+03, 1.95e+03, 4e+03, 4.33e+03, 1.94e+03, 2.01e+03, 1.62e+03, 1.12e+03,
03133    638, 636, 1.08e+03, 1.16e+03, 1.75e+03, 1.6e+03, 2.01e+03, 301, 616, 40, 0,
03134    367, 1.63e+04, 1.31e+04, 0, 0, 0, 0, 0, 0, 8.92e+03, 132, 0, 0, 0, 0, 0, 0,
03135    0, 0, 0},
03136   {652, 2.21e+03, 261, 841, 1.95e+03, 847, 1.46e+03, 2.34e+03, 192, 177,
03137    3.1e+03, 1.49e+03, 1.13e+03, 3.15e+03, 1.62e+03, 1.12e+03, 1.66e+03,
03138    1.8e+03, 74.9, 1.07e+04, 3.78e+03, 1.84e+04, 7.64e+04, 6.15e+04, 1.94e+05,
03139    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03140    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03141    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03142    0, 0, 1.02e+04, 2.71e+04, 0, 0, 0, 1.44e+05, 7.26e+03, 1.41e+03, 581,
03143    1.53e+03, 1.51e+03, 1.03e+03, 2.55e+03, 0, 0, 1.77e+05, 8.09e+04, 1.65e+05,
03144    1.27e+05, 7.63e+04, 3.09e+05, 4.69e+04, 2.88e+04, 6.33e+04, 1.91e+05,
03145    5.93e+05, 129, 294, 3.55e+04, 4.31e+03, 1.19e+04, 6.3e+03, 5.06e+03,
03146    6.51e+03, 0, 112, 786, 937, 1.75e+03, 1.79e+03, 4.12e+03, 2.22e+03,
03147    4.21e+03, 2.49e+03, 1.51e+03, 1.89e+03, 1.29e+03, 1.26e+03, 1.1e+03,
03148    1.47e+03, 1.24e+03, 1.55e+03, 673, 1.04e+03, 1.13e+03, 1.46e+03, 1.02e+03,
03149    23.2, 0, 2.62e+03, 1.25e+04, 4.34e+04, 6.87e+03, 0, 0, 0, 0, 0, 1.13e+03,
03150    2.98e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03151   {674, 2.37e+03, 239, 810, 2.02e+03, 845, 1.64e+03, 2.38e+03, 203, 186,
03152    4.5e+03, 2.51e+03, 950, 4.38e+03, 1.74e+03, 3.17e+03, 1.2e+03, 2.16e+03,
03153    927, 4.33e+03, 4.38e+03, 7.28e+04, 5.16e+04, 9.61e+04, 1.85e+05, 0, 0, 0,
03154    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03155    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.01e+05, 0, 0, 0, 0, 0, 0,
03156    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03157    0, 0, 0, 0, 0, 0, 0, 1.79e+05, 2.22e+04, 1.21e+03, 294, 564, 1.83e+03, 576,
03158    1.9e+03, 2.18e+03, 0, 1.62e+05, 1.36e+05, 9.5e+04, 7.03e+04, 1.26e+05,
03159    1.59e+05, 9.39e+04, 3.98e+04, 4.26e+04, 8.91e+04, 7.85e+05, 8.82e+03,
03160    2.05e+04, 6.1e+04, 1.14e+04, 2.38e+03, 4.78e+03, 1.69e+03, 1.9e+03,
03161    1.66e+03, 1.62e+03, 2.05e+03, 1.6e+03, 1.2e+03, 1.92e+03, 3.84e+03, 4e+03,
03162    5.25e+03, 2.24e+03, 1.21e+03, 1.87e+03, 1.5e+03, 589, 580, 1.2e+03,
03163    1.14e+03, 2.09e+03, 594, 710, 1.05e+03, 534, 2.53e+03, 8.33, 0, 4.36e+03,
03164    9.48e+03, 4.56e+04, 3.09e+04, 0, 0, 0, 0, 0, 1.55e+03, 8.93e+03, 0, 0, 0,
03165    0, 0, 0, 0, 0, 0},
03166   {692, 2.55e+03, 211, 802, 2.06e+03, 868, 1.82e+03, 2.32e+03, 186, 177,
03167    6.11e+03, 3.1e+03, 1.95e+03, 3.49e+03, 1.37e+03, 7.03e+03, 2.67e+03,
03168    1.31e+03, 2.74e+03, 2.07e+03, 4.01e+03, 7.57e+04, 9.11e+04, 1.14e+05,
03169    3.86e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03170    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.4e+05,
03171    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03172    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.33e+04, 776, 226, 994, 953,
```

```
03173    6.02e+03, 7.54e+04, 4.86e+04, 8.37e+04, 6.65e+04, 2.05e+05, 1.16e+05,
03174    1.62e+05, 1.22e+05, 1.66e+05, 5.65e+04, 3.16e+04, 682, 7.9e+04, 1.12e+05,
03175    1.08e+05, 4.92e+04, 9.33e+03, 6.18e+03, 2.71e+03, 2.13e+03, 664, 1.1e+03,
03176    1.86e+03, 1.06e+03, 1.51e+03, 1.43e+03, 1.91e+03, 2.06e+03, 3.84e+03,
03177    7.02e+03, 2.65e+03, 2.21e+03, 1.84e+03, 3.38e+03, 3.22e+03, 1.43e+03, 641,
03178    896, 469, 1.58e+03, 750, 636, 687, 630, 1.36e+03, 374, 0, 2.65e+03, 107,
03179    3.45e+04, 5.13e+04, 1.43e+04, 0, 0, 0, 0, 6.08e+03, 1.45e+04, 0, 0, 0, 0,
03180    0, 0, 0, 0, 0},
03181   {709, 2.71e+03, 181, 802, 2.1e+03, 912, 2.04e+03, 2.31e+03, 141, 119,
03182    7.59e+03, 3.17e+03, 3.64e+03, 3.24e+03, 1.83e+03, 4.73e+03, 593, 2.69e+03,
03183    1.72e+03, 1.13e+03, 2.84e+03, 3.47e+04, 8.35e+04, 1.16e+05, 1.01e+03, 0, 0,
03184    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03185    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.33e+03, 0, 0, 0, 0,
03186    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03187    0, 0, 0, 0, 0, 0, 0, 0, 0, 77.6, 3.9e+03, 983, 575, 4.76e+03, 1.75e+05,
03188    1.11e+05, 6.99, 5.27e+03, 3.12e+04, 2.34e+05, 2.65e+05, 1.97e+05, 4.64e+05,
03189    8.93e+04, 9.47e+04, 1.13e+05, 5.24e+04, 8.99e+04, 1.43e+05, 1.11e+05,
03190    9.32e+04, 609, 4.66e+03, 2.41e+03, 2.52e+03, 633, 1.64e+03, 2.15e+03,
03191    1.79e+03, 1.73e+03, 1.22e+03, 1.28e+03, 5.23e+03, 1.44e+04, 1.2e+04,
03192    2.97e+04, 1.53e+04, 4.13e+03, 3.2e+03, 1.86e+03, 2.26e+03, 1.3e+03,
03193    3.4e+03, 544, 312, 891, 1.95e+03, 514, 381, 2.31e+03, 48.3, 0, 791, 0,
03194    2.92e+03, 5.49e+04, 7.56e+04, 0, 0, 0, 0, 6.3e+03, 1.53e+04, 1.83e+03, 0,
03195    0, 0, 0, 0, 0, 0, 0, 0},
03196   {728, 2.88e+03, 149, 808, 2.12e+03, 974, 2.33e+03, 2.39e+03, 86, 75.8,
03197    8.52e+03, 3.26e+03, 4.51e+03, 3.8e+03, 6.51e+03, 3.1e+03, 1.54e+03,
03198    1.49e+03, 2.78e+03, 536, 2.25e+03, 2.57e+04, 1.12e+05, 6.62e+04, 0, 0, 0,
03199    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03200    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26, 5.25e+03, 0, 0, 0,
03201    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03202    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 663, 1.79e+03, 1.06e+03, 5.13e+03, 1.44e+05,
03203    4.26e+03, 0, 1.12e+05, 1.37e+05, 9.49e+04, 4.67e+05, 6.45e+05, 3.73e+04,
03204    1.48e+04, 4.25e+04, 8.37e+04, 6.42e+04, 7.91e+04, 1.37e+05, 6.17e+05, 79.4,
03205    417, 791, 1.36e+03, 2.62e+03, 2.24e+03, 3.65e+03, 1.26e+03, 2.66e+03,
03206    2.83e+03, 2.67e+03, 1.77e+03, 5.29e+03, 6.25e+03, 9.74e+03, 2.98e+04,
03207    2.48e+04, 4.1e+03, 2.59e+03, 4.15e+03, 8.03e+03, 1.49e+04, 4.65e+03,
03208    2.61e+03, 7.82e+03, 7.09e+03, 1.59e+03, 1.08e+03, 428, 1.11e+03, 1.1e+03,
03209    0.292, 0.0166, 0, 0, 2.42e+04, 6.5e+04, 1e+04, 0, 0, 0, 1.04e+04, 1.14e+04,
03210    4.38e+03, 0, 0, 0, 0, 0, 0, 0, 0},
03211   {749, 3.04e+03, 119, 820, 2.12e+03, 1.06e+03, 2.54e+03, 2.64e+03, 33.8,
03212    83.4, 8.43e+03, 3.82e+03, 4.5e+03, 3.86e+03, 1.22e+04, 5.48e+03, 2.04e+03,
03213    4.3e+03, 1.02e+03, 1.4e+03, 5.07e+03, 3e+04, 1.58e+05, 8.49e+03, 0, 0, 0,
03214    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03215    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03216    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03217    0, 0, 0, 0, 0, 0, 0, 0, 0, 34.2, 1.7e+03, 1.49e+04, 2.28e+05, 954, 4.47,
03218    4.57e+04, 7.92e+04, 1.35e+05, 4.57e+04, 2.55e+04, 4.36e+04, 1.08e+05,
03219    8.96e+04, 1.23e+05, 9.88e+04, 1.82e+05, 3.16e+05, 1.4e+05, 234, 643, 467,
03220    72.7, 403, 39.8, 35.9, 2.33e+03, 1.12e+03, 1.9e+03, 5.6e+03, 1.35e+03,
03221    1.24e+03, 1.62e+03, 2.31e+03, 1.64e+04, 4.01e+04, 5.2e+03, 3.29e+03,
03222    4.41e+03, 1.33e+04, 2.91e+04, 3.09e+04, 3.52e+04, 3.1e+04, 4.91e+04,
03223    6.55e+04, 2.01e+03, 640, 1.7e+03, 162, 1.56e+03, 0, 0, 0, 655, 6.94e+04,
03224    8.31e+04, 0, 0, 0, 3.64e+03, 3.95e+03, 3.82e+03, 0, 0, 0, 0, 0, 0, 0, 0},
03225   {766, 3.19e+03, 91.5, 835, 2.13e+03, 1.15e+03, 2.65e+03, 2.9e+03, 35.1, 104,
03226    7.79e+03, 4.55e+03, 5.12e+03, 4.07e+03, 1.44e+04, 2.96e+03, 2.18e+03,
03227    7.67e+03, 6.14e+03, 2.54e+03, 5.2e+03, 4.21e+04, 1.59e+05, 5.82e+03, 0, 0,
03228    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03229    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03230    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03231    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 501, 226, 0, 30.8, 6.87e+04, 9.71e+04,
03232    1.23e+05, 2.17e+05, 1.57e+05, 4.19e+04, 9.04e+04, 1.8e+05, 8.88e+04,
03233    1.3e+05, 1.08e+05, 3.59e+05, 422, 147, 559, 242, 1.22e+03, 192, 20.5,
03234    8.02e-11, 611, 4.79e+03, 1.31e+03, 1.32e+03, 1.16e+03, 950, 864, 1.06e+03,
03235    2.53e+03, 1.51e+04, 5.39e+03, 5.02e+03, 5.56e+03, 2.68e+03, 9.49e+03,
03236    1.54e+04, 2.09e+04, 1.74e+04, 3.08e+04, 8.78e+04, 2.17e+04, 773, 2.35e+03,
03237    96.2, 720, 0, 0, 0, 0, 5.84e+04, 1.09e+05, 0, 0, 0, 677, 2.18e+04, 0, 0, 0,
03238    0, 0, 0, 0, 0, 0},
03239   {782, 3.34e+03, 69.8, 843, 2.16e+03, 1.22e+03, 2.75e+03, 2.9e+03, 80.7, 135,
03240    6.97e+03, 4.74e+03, 6.39e+03, 5.01e+03, 1.75e+04, 4.93e+03, 1.07e+03,
03241    7.63e+03, 6.49e+03, 8.29e+03, 6.03e+03, 3.08e+04, 1.81e+05, 1.14e+03, 0, 0,
03242    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03243    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03244    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03245    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 935, 1.02e+05, 9.01e+04,
03246    4.35e+04, 1.37e+05, 2.16e+05, 5.63e+04, 6.48e+04, 2e+04, 3.04e+04,
03247    2.62e+04, 1.17e+05, 6.17e+03, 337, 241, 367, 360, 1.99e+03, 291, 51.8,
03248    41.6, 109, 2.33e+03, 2.35e+03, 1.83e+03, 314, 1.03e+03, 1.26e+03, 733,
03249    1.26e+03, 965, 1.44e+03, 858, 1.16e+03, 554, 784, 933, 1.79e+03, 2.41e+03,
03250    2.61e+03, 9.22e+04, 2.13e+04, 1.26e+03, 2.38e+03, 2.08e+03, 1.71e+03, 0, 0,
03251    0, 0, 2.43e+04, 1.2e+05, 0, 0, 0, 1.53e+03, 1.92e+04, 0, 0, 0, 0, 0, 0, 0,
03252    0, 0},
03253   {798, 3.47e+03, 52.3, 853, 2.21e+03, 1.28e+03, 3e+03, 2.62e+03, 140, 173,
03254    5.89e+03, 4.89e+03, 8.34e+03, 6.63e+03, 1.89e+04, 5.69e+03, 5.45e+03,
03255    1.01e+04, 4.77e+03, 9.36e+03, 1.04e+04, 1.46e+04, 2.52e+05, 0, 0, 0, 0, 0,
03256    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03257    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03258    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03259    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 593, 5.69e+04, 4.11e+04, 1.67e+05,
```

```
03260    2.52e+04, 3.53e+03, 3e+03, 8.67e+03, 5.59e+04, 6.88e+04, 9.35e+04,
03261    1.64e+04, 3.88e+03, 265, 258, 335, 254, 481, 169, 75, 88.4, 107, 258, 223,
03262    1.25e+03, 1.66e+03, 527, 436, 416, 392, 394, 245, 779, 556, 420, 669, 679,
03263    1.72e+03, 1.4e+03, 2.49e+03, 3.9e+04, 5.85e+04, 1.27e+03, 2.64e+03,
03264    4.12e+03, 3.47e+03, 0, 0, 0, 0, 1.94e+03, 1.01e+05, 0, 0, 0, 0.0178,
03265    8.67e+03, 405, 0, 0, 0, 0, 0, 0, 0, 0},
03266   {814, 3.59e+03, 39.8, 859, 2.29e+03, 1.32e+03, 3.31e+03, 2.32e+03, 210, 119,
03267    4.53e+03, 5.46e+03, 1.09e+04, 8.57e+03, 1.71e+04, 1.19e+04, 1.68e+04,
03268    1.69e+04, 4.51e+03, 9.18e+03, 4e+03, 9.46e+03, 2.53e+05, 0, 0, 0, 0, 0, 0,
03269    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03270    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20.2, 0, 0, 0, 0, 0,
03271    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03272    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.87e+03, 6.83e+04, 4.04e+04, 2.44e+04,
03273    8.84e+04, 5.09e+04, 7.71e+03, 4.09e+04, 1.3e+05, 2.52e+05, 3.23e+05,
03274    1.82e+04, 1.5e+04, 1.11e+03, 167, 91.4, 2.57e+03, 2.81e+03, 8.48e+03, 333,
03275    60.7, 160, 403, 792, 789, 778, 793, 578, 259, 308, 347, 505, 514, 547, 411,
03276    281, 542, 801, 1.26e+03, 1.32e+03, 952, 4.17e+04, 3.98e+03, 3e+03,
03277    2.07e+03, 1.84e+03, 0, 0, 0, 0, 0, 7.74e+04, 0, 0, 0, 0, 1.06e+04,
03278    4.25e+03, 0, 0, 0, 0, 0, 0, 0, 0},
03279   {822, 3.7e+03, 30.1, 863, 2.38e+03, 1.36e+03, 3.6e+03, 2.02e+03, 233, 86.7,
03280    3.49e+03, 6.15e+03, 1.36e+04, 1.07e+04, 1.67e+04, 3.64e+04, 1.04e+04,
03281    1.39e+04, 9.39e+03, 1.26e+04, 3.31e+03, 1.82e+04, 1.51e+05, 0, 0, 0, 0, 0,
03282    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03283    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03284    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03285    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 266, 6.97e+04, 2.59e+04, 9.7e+03,
03286    9.55e+04, 5.91e+04, 7.96e+03, 6.09e+04, 2.4e+05, 1.14e+05, 3.09e+05,
03287    2.07e+05, 2.98e+04, 203, 598, 210, 1.14e+04, 1.94e+04, 6.5e+03, 259, 75.1,
03288    1.03e+03, 1.04e+03, 742, 560, 618, 2.13e+03, 2.04e+03, 351, 1.1e+03, 357,
03289    526, 561, 348, 178, 190, 610, 1.28e+03, 796, 2.41e+03, 2.42e+03, 8.24e+03,
03290    4.53e+04, 922, 4.14e+03, 333, 0, 0, 0, 0, 0, 6.64e+04, 0, 0, 0, 0,
03291    1.39e+04, 172, 0, 0, 0, 0, 0, 0, 0, 0},
03292   {838, 3.8e+03, 23.4, 867, 2.46e+03, 1.41e+03, 4.05e+03, 1.74e+03, 250, 96,
03293    3.27e+03, 6.91e+03, 1.59e+04, 1.4e+04, 1.69e+04, 6.34e+04, 1.62e+04,
03294    1.62e+04, 1.78e+04, 1.57e+04, 1.26e+04, 2.64e+04, 1.79e+05, 0, 0, 0, 0, 0,
03295    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03296    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03297    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03298    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.78e+03, 6.16e+04, 5.52e+04, 1.13e+05,
03299    6.44e+04, 1.08e+04, 3.62e+03, 8.79e+04, 2.88e+05, 6.44e+04, 1.2e+05,
03300    4.71e+05, 1.02e+04, 970, 705, 9.37e+03, 8.66e+03, 7.67e+03, 1.5e+03, 304,
03301    57.5, 1.91e+03, 2.05e+03, 1.46e+03, 1.17e+03, 1.04e+03, 1.82e+03, 2.93e+03,
03302    1.69e+03, 1.29e+03, 1.34e+03, 716, 640, 426, 179, 144, 735, 1.01e+03, 725,
03303    1.58e+03, 874, 1.53e+03, 5.36e+04, 3.45e+03, 4.42e+03, 0, 0, 0, 0, 0, 0,
03304    7.02e+04, 282, 0, 0, 0, 4.84e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03305   {843, 3.88e+03, 17.7, 868, 2.52e+03, 1.5e+03, 4.71e+03, 1.56e+03, 347, 174,
03306    3.3e+03, 7.75e+03, 1.72e+04, 1.76e+04, 1.59e+04, 6.82e+04, 3.65e+04,
03307    2.91e+04, 2.65e+04, 3e+04, 1.29e+04, 1e+04, 1.74e+05, 0, 0, 0, 0, 0, 0, 0,
03308    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03309    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03310    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03311    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12.8, 5.05e+04, 1.17e+05, 6.89e+04, 2.08e+05,
03312    8.44e+04, 3.66e+04, 1.4e+05, 2.47e+05, 1.96e+05, 1.34e+05, 7.53e+05,
03313    1.19e+04, 7.53e+04, 1.15e+04, 5.63e+04, 4.94e+04, 364, 367, 184, 48.3,
03314    1.29e+03, 4.59e+03, 7.08e+03, 2.32e+03, 1.47e+03, 1.54e+03, 624, 116, 285,
03315    145, 971, 136, 173, 228, 135, 361, 2.25e+03, 1.94e+03, 191, 183, 225,
03316    5.72e+04, 5.44e+04, 3.07e+03, 0, 0.0654, 0, 0, 0, 0, 2.67e+04, 3.22e+04, 0,
03317    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03318   {853, 3.95e+03, 14.2, 857, 2.58e+03, 1.57e+03, 5.44e+03, 1.44e+03, 498, 293,
03319    3.05e+03, 8.83e+03, 1.76e+04, 2e+04, 1.44e+04, 5.4e+04, 4.09e+04, 3.41e+04,
03320    4.62e+04, 9.77e+04, 5.51e+04, 5.27e+03, 1.55e+05, 0, 0, 0, 0, 0, 0, 0, 0,
03321    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03322    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03323    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03324    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.967, 9.04e+03, 6.91e+04, 2.04e+05, 2.39e+05,
03325    5.56e+05, 2.23e+05, 1.11e+05, 7.17e+04, 2.39e+05, 1.76e+05, 1.24e+05,
03326    6.31e+05, 1.36e+05, 4.06e+05, 6.28e+05, 5.38e+04, 7.95e+04, 946, 124,
03327    7.71e+03, 7.25e+03, 1.79e+03, 2.02e+03, 4.97e+03, 8.45e+03, 3.29e+03,
03328    1.22e+03, 1.89e+03, 2.78e+03, 331, 163, 120, 897, 40.8, 54.8, 446, 311,
03329    1.51e+03, 1.23e+03, 1.82e+03, 398, 200, 2.33e+04, 2.78e+04, 2.04e+03, 229,
03330    51.5, 0, 0, 0, 0, 2.08e+04, 7.24e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03331    0},
03332   {867, 4e+03, 11.8, 842, 2.64e+03, 1.62e+03, 6.05e+03, 1.38e+03, 647, 401,
03333    2.65e+03, 1e+04, 1.69e+04, 2.2e+04, 1.28e+04, 3.69e+04, 4.54e+04, 1.08e+05,
03334    6.85e+03, 6.26e+04, 2.85e+04, 6.93e+04, 1.56e+05, 0, 0, 0, 0, 0, 0, 0, 0,
03335    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03336    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03337    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03338    0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 15.6, 6.16e+03, 2.6e+04, 90.5, 126, 9.53e+03,
03339    5.87e+05, 9.53e+04, 1.57e+04, 1.84e+05, 1.4e+05, 4.14e+05, 5.17e+05,
03340    7.39e+04, 1.73e+05, 9.02e+05, 5.61e+05, 620, 1.81e+03, 717, 7.67e+04,
03341    1.32e+03, 861, 1.04e+03, 3.66e+03, 3.33e+03, 6.23e+03, 540, 1.58e+03,
03342    2.93e+03, 1.66e+03, 153, 104, 390, 162, 35.8, 164, 224, 755, 766, 630, 481,
03343    668, 1.96e+03, 2.85e+03, 52.5, 270, 66.4, 5.85, 0, 0, 0, 5.72e+03,
03344    4.81e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03345   {875, 4.05e+03, 10.2, 825, 2.66e+03, 1.67e+03, 6.3e+03, 1.39e+03, 780, 493,
03346    2.11e+03, 1.15e+04, 1.55e+04, 2.33e+04, 1.26e+04, 2.69e+04, 1.32e+04,
```

```
03347    1.33e+05, 1.22e+04, 8.03e+03, 4.41e+03, 7.91e+04, 9.16e+04, 0, 0, 0, 0, 0,
03348    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.72e+04, 6.39e+04, 0, 0, 0, 0, 0, 0,
03349    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03350    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03351    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 113, 3.09e+03, 785, 37.2, 126,
03352    252, 53.3, 9.31e+04, 4.52e+04, 6.02e+04, 1.36e+05, 8.02e+04, 1.6e+05,
03353    7.44e+05, 6.16e+05, 7.51e+04, 1.29e+05, 8.4e+05, 1.13e+06, 1.06e+04,
03354    1.61e+04, 1.68e+04, 8.72e+04, 1.19e+03, 817, 1.1e+03, 2.7e+03, 1.11e+03,
03355    2.94e+03, 903, 2.35e+03, 5.62e+03, 2.14e+03, 404, 416, 432, 516, 734, 56.2,
03356    517, 431, 310, 375, 606, 286, 760, 375, 298, 264, 190, 129, 16.9, 0, 0, 18,
03357    2.04e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03358   {879, 4.09e+03, 8.9, 809, 2.65e+03, 1.73e+03, 6.2e+03, 1.48e+03, 894, 511,
03359    1.73e+03, 1.32e+04, 1.44e+04, 2.39e+04, 1.3e+04, 1.53e+04, 4.75e+03,
03360    4.14e+04, 8.79e+04, 0, 0, 3.88e+03, 2.14e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03361    0, 0, 0, 0, 0, 0, 0, 0, 6.85e+03, 7.51e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03362    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03363    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03364    0, 0, 0, 0, 0, 0, 0, 0, 818, 2.48e+03, 2.74e+03, 705, 1.07e+03, 1.14e+03,
03365    1.49e+03, 270, 6.01e+04, 2.68e+05, 3.08e+05, 3.58e+05, 3.63e+05, 6.35e+05,
03366    1.27e+06, 8.45e+05, 4.93e+05, 5.98e+05, 6.62e+05, 1.68e+06, 5.17e+05,
03367    5.64e+05, 6.78e+04, 6.75e+03, 1.75e+03, 1.72e+03, 1.67e+03, 1.89e+03,
03368    1.25e+03, 1.06e+03, 540, 1.27e+03, 6.03e+03, 815, 52.3, 402, 154, 253, 179,
03369    536, 249, 494, 367, 304, 502, 386, 196, 231, 143, 195, 159, 128, 58.3, 0,
03370    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03371   {886, 4.11e+03, 8.37, 790, 2.61e+03, 1.8e+03, 5.88e+03, 1.63e+03, 956, 435,
03372    1.79e+03, 1.4e+04, 1.36e+04, 2.48e+04, 1.16e+04, 8.38e+03, 9.42e+03,
03373    5.14e+04, 1.11e+05, 1.19e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03374    0, 0, 0, 0, 0, 0, 0.00662, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03375    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03376    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03377    0, 0, 0, 9.8e+03, 2.4e+03, 1.97e+03, 876, 3.13e+03, 2.07e+03, 2.37e+03,
03378    278, 57.2, 384, 780, 1.44e+04, 6.99e+04, 3.84e+05, 1.19e+06, 5.33e+05,
03379    6.53e+05, 5.56e+05, 6.09e+05, 1.74e+06, 8.48e+05, 3.97e+05, 1.12e+05, 767,
03380    1.56e+03, 1.57e+03, 1.98e+03, 3.06e+03, 2.54e+03, 998, 542, 985, 3.69e+03,
03381    646, 46.3, 212, 449, 378, 124, 289, 182, 291, 260, 140, 371, 426, 150, 226,
03382    440, 177, 272, 112, 72.1, 0.011, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03383    0, 0},
03384   {890, 4.14e+03, 8.21, 771, 2.55e+03, 1.87e+03, 5.56e+03, 1.75e+03, 1.08e+03,
03385    320, 2.19e+03, 1.42e+04, 1.31e+04, 2.45e+04, 7.77e+03, 3.16e+03, 657,
03386    5.92e+04, 4.85e+04, 6.45e+04, 0.00999, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03387    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03388    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03389    0.00222, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03390    0, 0, 0, 0, 0, 0, 600, 2.44e+03, 588, 3.34e+03, 6.53e+03, 3.56e+03, 1.03e+03,
03391    592, 113, 92.2, 227, 4.11e+03, 3.04e+04, 4.32e+04, 8.47e+05, 1.02e+06,
03392    6.06e+05, 6.05e+05, 7.8e+05, 1.54e+06, 1.13e+05, 5.66e+05, 4.21e+05,
03393    3.17e+03, 2.62e+03, 2.33e+03, 4.12e+03, 1.14e+04, 1.64e+03, 2.3e+03,
03394    1.64e+03, 1e+03, 1.57e+03, 656, 132, 160, 365, 425, 141, 300, 226, 115,
03395    319, 316, 448, 222, 68, 160, 412, 67.5, 75.9, 163, 63.4, 0, 0, 0, 0, 0, 0,
03396    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03397   {897, 4.14e+03, 8.51, 756, 2.48e+03, 1.94e+03, 5.25e+03, 1.87e+03, 1.29e+03,
03398    226, 2.8e+03, 1.49e+04, 1.3e+04, 2.27e+04, 5.04e+03, 7.12e+03, 568,
03399    4.18e+04, 1.06e+04, 1.18e+05, 4.22e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03400    0, 0, 0, 0, 9.7e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03401    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03402    0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 0, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03403    0, 0, 0, 0, 114, 3.67e+03, 1.7e+04, 2.96e+04, 2.59e+04, 4.9e+03, 8.46e+03,
03404    2.91e+03, 2.11e+04, 4.34e+04, 1.51e+03, 803, 2.46e+03, 394, 85.2, 82.5,
03405    165, 2.36e+04, 2.11e+05, 1.11e+06, 6.45e+05, 4.28e+05, 2.68e+05, 3.04e+05,
03406    7.8e+05, 4.32e+05, 6.89e+05, 7.13e+05, 6.39e+03, 7.87e+03, 2.8e+03,
03407    9.63e+03, 5.45e+03, 6.43e+03, 2.22e+03, 5.45e+03, 4.48e+03, 2.41e+03, 834,
03408    177, 86.1, 633, 232, 104, 275, 234, 59.1, 314, 260, 514, 258, 93.6, 238,
03409    159, 8.42, 60.9, 38.1, 8.97, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03410    0, 0},
03411   {890, 4.14e+03, 8.93, 730, 2.38e+03, 2.03e+03, 4.88e+03, 1.98e+03, 1.54e+03,
03412    155, 3.63e+03, 1.6e+04, 1.29e+04, 1.99e+04, 3.04e+03, 6.73e+03, 1.64e+03,
03413    2.53e+04, 1.14e+04, 2.57e+04, 2.25e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03414    0, 0, 0, 0, 2.57e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03415    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03416    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26.9,
03417    4.44e+04, 1.06e+05, 5.65e+04, 4.39e+04, 3.01e+04, 1.65e+04, 4.86e+04,
03418    3.32e+04, 9.91e+03, 1.88e+04, 1.86e+04, 1.43e+04, 1.01e+03, 1.73e+03, 288,
03419    85.3, 135, 161, 1.07e+05, 6.81e+05, 8.21e+05, 9.45e+05, 3.08e+05, 1.36e+05,
03420    2.45e+05, 3.76e+05, 4.68e+05, 4.05e+05, 8.36e+05, 3.32e+04, 4.37e+03, 326,
03421    4.77e+03, 9.69e+03, 8.05e+03, 3.39e+03, 3.17e+03, 5.41e+03, 680, 251, 163,
03422    231, 505, 420, 261, 347, 250, 155, 369, 258, 376, 194, 436, 65.7, 269, 732,
03423    344, 395, 2.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03424   {901, 4.14e+03, 10.3, 702, 2.27e+03, 2.14e+03, 4.54e+03, 1.93e+03, 1.81e+03,
03425    119, 4.52e+03, 1.72e+04, 1.22e+04, 1.71e+04, 2.14e+03, 2.55e+03, 4.68e+03,
03426    1.32e+04, 7.81e+03, 1.06e+04, 4.01e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03427    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03428    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03429    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.2e+04, 8.56e+04,
03430    7.13e+04, 1.11e+04, 4.27e+03, 5.96e+03, 5.91e+03, 8.33e+03, 9.34e+03,
03431    8.04e+03, 2.06e+04, 5.76e+03, 3.35e+03, 3.41e+03, 2.54e+03, 2.54e+03,
03432    4.84e+03, 438, 119, 184, 667, 8.07e+05, 9.46e+05, 4.57e+05, 1.35e+06,
03433    4.27e+05, 3.14e+05, 1.26e+06, 4.62e+05, 5.37e+05, 3.21e+05, 7.14e+05,
```

```
03434    5.96e+04, 288, 155, 7.72e+03, 1.2e+04, 1.01e+04, 3.77e+03, 5.26e+03,
03435    3.23e+03, 140, 122, 68.9, 589, 538, 195, 340, 237, 406, 219, 352, 173, 195,
03436    416, 164, 409, 462, 296, 262, 86.3, 38.7, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0,
03437    0, 0, 0, 0, 0, 0, 0, 0},
03438   {899, 4.14e+03, 11.7, 666, 2.15e+03, 2.31e+03, 4.23e+03, 1.7e+03, 2.11e+03,
03439    70.1, 5.12e+03, 1.74e+04, 1.14e+04, 1.49e+04, 2.04e+03, 743, 5.05e+03,
03440    1.06e+04, 417, 2.12e+04, 2.65e+05, 2.41e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03441    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03442    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03443    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.426, 665, 8.5e+04,
03444    1.13e+05, 3.95e+04, 1.26e+04, 4.4e+03, 2.71e+03, 4.06e+03, 5.5e+03,
03445    1.41e+04, 4.63e+03, 2.12e+04, 1.75e+04, 3.18e+03, 2.59e+03, 5.16e+03,
03446    1.6e+03, 6.99e+03, 8.24e+03, 228, 137, 3.15e+03, 5.02e+05, 1.3e+06,
03447    5.7e+05, 6.14e+05, 5.93e+05, 5.83e+05, 4.96e+05, 6.87e+05, 8.02e+05,
03448    3.89e+05, 4.21e+05, 6.82e+05, 1.97e+04, 466, 127, 5.76e+03, 1e+04,
03449    1.06e+04, 8.72e+03, 1.17e+04, 1.4e+03, 114, 88.1, 61.6, 115, 211, 364,
03450    69.5, 269, 410, 183, 284, 217, 283, 217, 232, 120, 357, 322, 175, 49.6,
03451    1.85, 0.793, 0, 0, 0, 0, 0, 0.00332, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03452   {898, 4.13e+03, 13.6, 618, 2.06e+03, 2.5e+03, 3.88e+03, 1.43e+03, 2.36e+03,
03453    18.3, 5.41e+03, 1.72e+04, 1.06e+04, 1.21e+04, 1.5e+03, 1.06e+03, 935,
03454    6.17e+03, 5.66e+03, 1.78e+04, 7.87e+04, 5.6e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03455    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03456    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03457    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.42e+04, 1.34e+05,
03458    3.21e+05, 2.48e+05, 6.43e+04, 6.03e+03, 7.54e+03, 8.12e+03, 2.47e+03,
03459    2.95e+03, 2.52e+03, 5.99e+03, 2.7e+03, 2.31e+04, 1.59e+04, 8.27e+03,
03460    2.53e+03, 6.28e+03, 7.07e+03, 2.54e+03, 2.7e+03, 141, 748, 4.46e+05,
03461    2.21e+06, 3.53e+05, 3.62e+05, 6.59e+05, 3.65e+05, 1.1e+06, 2e+05, 3.11e+03,
03462    3.04e+05, 2.14e+05, 8.71e+05, 1.3e+06, 1.27e+04, 7.65e+03, 108, 1.09e+04,
03463    7.37e+03, 5.68e+03, 2.25e+04, 4.55e+03, 197, 77.6, 78.5, 63.9, 47.4, 236,
03464    327, 346, 292, 406, 131, 317, 104, 119, 195, 248, 114, 133, 333, 138, 99.4,
03465    0.226, 0, 0, 0, 0, 0, 0, 0.058, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03466   {909, 4.12e+03, 15.6, 560, 2e+03, 2.67e+03, 3.46e+03, 1.22e+03, 2.55e+03,
03467    27.9, 5.59e+03, 1.65e+04, 9.63e+03, 8.31e+03, 1.32e+03, 1.09e+03, 2.04e+03,
03468    1.98e+03, 8.12e+03, 1.04e+04, 4.87e+04, 1.55e+05, 55.1, 0, 0, 0, 0, 0, 0,
03469    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03470    0.0544, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03471    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03472    3.37e+03, 3.04e+03, 7.75e+04, 2.21e+04, 4.61e+04, 1.29e+04, 3.16e+04,
03473    1.04e+04, 4.03e+03, 6.67e+03, 4.48e+03, 3.31e+03, 4.41e+03, 6.41e+03,
03474    2.08e+04, 1.01e+04, 2.39e+03, 5.11e+03, 6.45e+03, 517, 90.1, 141, 1.69e+05,
03475    1.22e+06, 1.03e+06, 2.15e+05, 4.48e+05, 3.69e+05, 1.91e+05, 1.54e+06,
03476    2.68e+04, 501, 1.58e+05, 5.73e+05, 7.92e+05, 5.69e+05, 1.24e+05, 7.48e+04,
03477    3.16e+03, 1.99e+04, 2.02e+03, 8.26e+03, 1.43e+04, 1.99e+03, 306, 268, 69,
03478    76.9, 80, 154, 324, 291, 565, 200, 121, 226, 184, 191, 84.2, 296, 93.1,
03479    395, 262, 187, 89.7, 1.96, 0, 0, 0, 0, 0, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0,
03480    0, 0, 0},
03481   {910, 4.1e+03, 19.1, 501, 1.95e+03, 2.76e+03, 3e+03, 1.17e+03, 2.66e+03,
03482    107, 5.71e+03, 1.51e+04, 8e+03, 4.73e+03, 1.46e+03, 1.98e+03, 1.16e+03,
03483    2.06e+03, 3.37e+03, 1.29e+04, 4.61e+04, 8e+04, 5.78e+04, 0, 0, 0, 0, 0, 0,
03484    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03485    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03486    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 27.1, 1.76e+03,
03487    1.2e+04, 4e+04, 4.35e+04, 3.66e+04, 1.93e+04, 3.65e+04, 1.83e+04, 1.07e+04,
03488    9.33e+03, 6.19e+03, 5.21e+03, 2.22e+04, 1.48e+04, 3.98e+03, 4.19e+03, 648,
03489    219, 154, 5.95e+04, 3.42e+05, 1.53e+06, 4.05e+05, 2.78e+05, 2.54e+05,
03490    6.79e+04, 3.8e+05, 8.97e+05, 1.12e+03, 1.38e+03, 5.47e+04, 4.78e+05,
03491    4.41e+05, 4.51e+05, 3.08e+05, 6.58e+05, 2.7e+03, 7.83e+03, 2.06e+04,
03492    3.21e+04, 6.19e+03, 772, 182, 297, 472, 144, 37.2, 60.4, 275, 274, 286,
03493    206, 159, 205, 245, 91, 166, 442, 334, 329, 194, 279, 179, 6.04, 0, 0, 0,
03494    0, 0, 0, 0, 197, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03495   {917, 4.08e+03, 21.8, 433, 1.93e+03, 2.75e+03, 2.59e+03, 1.24e+03, 2.57e+03,
03496    226, 5.65e+03, 1.28e+04, 6.42e+03, 3.15e+03, 868, 1.95e+03, 1.86e+03,
03497    1.17e+03, 2.99e+03, 1.07e+04, 3.98e+04, 2.7e+04, 2.26e+05, 0, 0, 0, 0, 0,
03498    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03499    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03500    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.52e+03, 3.42e+04,
03501    3.59e+03, 741, 0.712, 25.9, 669, 3.38e+03, 8.2e+03, 1.25e+03, 3.99e+03,
03502    6.63e+03, 4.55e+03, 4.17e+03, 2.74e+03, 2.94e+03, 1.36e+04, 6.99e+03,
03503    3.67e+03, 1.02e+04, 198, 64, 127, 3.31e+05, 2.12e+06, 5.01e+05, 3.14e+05,
03504    2.1e+05, 2.64e+05, 6.58e+04, 1.13e+06, 6.6e+05, 575, 1.59e+03, 628,
03505    3.85e+04, 1.33e+06, 3.83e+05, 1.43e+05, 8.05e+05, 1.67e+04, 3.4e+03,
03506    2.05e+04, 9.25e+03, 6.75e+03, 2.57e+03, 1.4e+03, 1.03e+03, 855, 65.8, 31.8,
03507    44.4, 391, 490, 365, 553, 268, 597, 357, 153, 95.1, 237, 417, 328, 566,
03508    276, 293, 3.15, 2.55, 0, 0, 0, 0, 0, 0, 2.25e+03, 0, 0, 0, 0, 0, 0, 0, 0,
03509    0},
03510   {926, 4.06e+03, 24.9, 366, 1.93e+03, 2.64e+03, 2.34e+03, 1.28e+03, 2.29e+03,
03511    409, 4.96e+03, 1.03e+04, 5.2e+03, 2.5e+03, 710, 1.63e+03, 2.46e+03,
03512    2.2e+03, 2.45e+03, 6.67e+03, 1.83e+04, 3.26e+04, 3.02e+05, 0, 0, 0, 0, 0,
03513    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03514    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03515    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.77e+04, 2.52e+05,
03516    2.11e+04, 1.36e+03, 0.0665, 0, 0, 0, 0, 0, 31.8, 9.24e+03, 1.44e+04,
03517    2.85e+04, 2.1e+04, 3.72e+03, 2.88e+04, 2.01e+04, 4.82e+03, 1.01e+04, 293,
03518    106, 758, 5.6e+05, 1.88e+06, 1.72e+05, 1.15e+05, 1.57e+05, 1.35e+05,
03519    1.6e+05, 1.78e+06, 7.65e+04, 1.11e+03, 1.14e+03, 647, 7.65e+03, 1.32e+06,
03520    4.38e+05, 2.71e+05, 5.07e+05, 4.52e+05, 1.02e+05, 2.33e+04, 2.39e+04,
```

```
03521    4.52e+03, 3.91e+03, 920, 1.15e+03, 1.19e+03, 346, 82.1, 158, 373, 392, 298,
03522    453, 177, 899, 840, 164, 138, 138, 249, 269, 515, 417, 306, 3.28e+03, 278,
03523    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03524   {921, 4.04e+03, 30.2, 305, 1.93e+03, 2.44e+03, 2.18e+03, 1.17e+03, 1.95e+03,
03525    672, 3.95e+03, 7.79e+03, 4.02e+03, 2.67e+03, 1.71e+03, 2.76e+03, 606,
03526    5.57e+03, 1.79e+03, 4.9e+03, 1.62e+04, 3.97e+04, 2.66e+05, 0, 0, 0, 0, 0,
03527    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03528    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03529    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.6, 2.35e+03, 760,
03530    0, 0, 0, 0, 0, 0, 0, 0.133, 7.95e+03, 8.87e+04, 2.12e+04, 1.3e+04,
03531    5.77e+03, 3.76e+03, 2.19e+04, 1.94e+04, 1.11e+04, 64.7, 133, 2.45e+05,
03532    1.13e+06, 1.23e+06, 1.14e+05, 1.01e+05, 9.38e+04, 8.99e+04, 6.97e+04,
03533    9e+05, 2.06e+05, 5.1e+03, 715, 680, 495, 9.39e+05, 8.01e+05, 4.11e+05,
03534    5.95e+05, 3.51e+05, 3.07e+05, 7.67e+04, 2.02e+04, 1.76e+04, 1.02e+04,
03535    1.84e+03, 702, 2.57e+03, 639, 148, 239, 629, 321, 330, 285, 213, 1.38e+03,
03536    768, 588, 86.2, 231, 341, 396, 732, 433, 557, 4.47e+03, 864, 12.4, 3.83, 0,
03537    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03538   {924, 4.03e+03, 34.9, 242, 1.94e+03, 2.19e+03, 2.07e+03, 995, 1.77e+03, 840,
03539    3.21e+03, 6.29e+03, 2.49e+03, 2.79e+03, 1.66e+03, 1.86e+03, 2.26e+03,
03540    3.01e+03, 4.24e+03, 4.75e+03, 1.74e+04, 4.02e+04, 2.13e+05, 61.3, 0, 0, 0,
03541    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03542    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03543    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03544    0, 0, 0, 0, 0, 338, 1.81e+05, 8.81e+04, 3.92e+04, 6.77e+03, 5.21e+03,
03545    2.23e+04, 2.05e+04, 1.34e+04, 108, 119, 6.55e+05, 1.22e+06, 1.69e+05,
03546    1.36e+05, 1.16e+05, 5.7e+04, 3.48e+04, 2.95e+04, 1.18e+05, 7.06e+05,
03547    2.76e+03, 1.37e+03, 1.22e+03, 179, 4.35e+05, 9.31e+05, 7.84e+05, 3.35e+05,
03548    4.33e+05, 3.34e+05, 2.8e+04, 9.51e+04, 7.01e+04, 7.55e+04, 1.32e+04, 902,
03549    2.06e+03, 2.01e+03, 254, 69.6, 574, 288, 434, 230, 383, 1.87e+03, 1.09e+03,
03550    1.34e+03, 126, 199, 539, 1.24e+03, 1.12e+03, 236, 1.08e+03, 5.25e+03,
03551    1e+03, 0, 4.43, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03552   {928, 4.02e+03, 39.4, 188, 1.96e+03, 1.87e+03, 1.99e+03, 904, 1.72e+03, 767,
03553    2.3e+03, 5.3e+03, 1.36e+03, 2.2e+03, 720, 366, 1.27e+03, 2.83e+03,
03554    7.27e+03, 3.81e+03, 1.44e+04, 3.54e+04, 1.1e+05, 1.83e+04, 0, 0, 0, 0, 0,
03555    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03556    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03557    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03558    0, 0, 0, 0, 526, 7.16e+04, 5.34e+04, 1.58e+04, 7.76e+03, 4.42e+04,
03559    3.42e+04, 1.42e+04, 40.2, 3.95e+03, 7.28e+05, 8.02e+05, 1.5e+05, 2e+05,
03560    1.14e+05, 3.59e+04, 6.46e+04, 2.49e+04, 1.88e+04, 1.33e+06, 8.46e+03,
03561    1.76e+03, 1.83e+03, 126, 8.67e+05, 3.45e+05, 1.1e+06, 1.17e+05, 1.74e+05,
03562    2.54e+05, 3.54e+04, 9.11e+04, 2.63e+05, 1.73e+05, 5.6e+04, 1.24e+03,
03563    2.23e+03, 3.79e+03, 1.81e+03, 480, 400, 590, 401, 228, 212, 1.18e+03, 752,
03564    719, 628, 336, 687, 709, 1.07e+03, 1.52e+03, 588, 4.78e+03, 1.28e+03, 0, 0,
03565    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03566   {936, 4.01e+03, 45.6, 144, 1.98e+03, 1.52e+03, 2.09e+03, 959, 1.75e+03, 615,
03567    1.66e+03, 4.16e+03, 814, 1.14e+03, 402, 113, 708, 2.48e+03, 6.88e+03,
03568    3.21e+03, 1.15e+04, 2.31e+04, 7.57e+04, 5.38e+04, 0, 0, 0, 0, 0, 0, 0, 0,
03569    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03570    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03571    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03572    0, 0, 3.08e+03, 1.51e+04, 1.26e+04, 3.17e+04, 1.38e+04, 1.85e+04, 2.97e+03,
03573    55.7, 4.4e+04, 1.95e+06, 2.8e+05, 1.2e+05, 8.17e+04, 1.14e+05, 4.94e+04,
03574    3.42e+04, 3.4e+04, 1.53e+04, 1.31e+06, 5.24e+04, 3.22e+03, 2.36e+03, 265,
03575    8.89e+05, 2.6e+05, 1.72e+06, 4.03e+03, 1.03e+05, 2.72e+05, 3.58e+04,
03576    2.7e+05, 1.9e+05, 7.58e+04, 1.06e+05, 1.52e+03, 4.6e+03, 6.28e+03,
03577    2.48e+03, 1.26e+03, 652, 440, 243, 146, 454, 1.43e+03, 254, 540, 734, 609,
03578    608, 674, 1.72e+03, 1.28e+03, 396, 2.73e+03, 1.19e+03, 77.2, 0, 0, 0, 0, 0,
03579    0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03580   {940, 3.99e+03, 52.8, 104, 2e+03, 1.2e+03, 2.42e+03, 982, 1.59e+03, 576,
03581    1.31e+03, 3.03e+03, 544, 714, 335, 74.6, 494, 1.51e+03, 6.58e+03, 3.07e+03,
03582    1.14e+04, 2.15e+04, 8.04e+04, 6.28e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03583    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03584    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03585    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03586    820, 2.18e+04, 3.36e+04, 1.09e+04, 9.97e+03, 1.19e+04, 83.9, 3.12e+05,
03587    2.84e+06, 1.26e+05, 9.83e+04, 8.34e+04, 5.34e+04, 3.64e+04, 7.49e+04,
03588    3.46e+04, 2.81e+04, 2.87e+05, 7.37e+05, 2.06e+03, 2.47e+03, 275, 2.94e+05,
03589    3e+05, 1.3e+06, 684, 2.22e+04, 3.22e+05, 5.76e+04, 2.17e+05, 2.27e+05,
03590    1.61e+05, 1.01e+05, 2.83e+03, 7.69e+03, 1.65e+03, 2.05e+03, 1.36e+03, 757,
03591    365, 255, 383, 1.81e+03, 1.2e+03, 1.05e+03, 1.7e+03, 537, 1.05e+03,
03592    1.67e+03, 735, 579, 930, 530, 2.92e+03, 229, 203, 0, 0, 0, 0, 0, 0, 0, 0,
03593    0, 0, 0, 0, 0, 0, 0, 0},
03594   {942, 3.98e+03, 59.9, 72.9, 1.99e+03, 1.03e+03, 2.87e+03, 750, 1.33e+03,
03595    586, 891, 2.21e+03, 286, 898, 538, 111, 303, 1.94e+03, 5.99e+03, 3.35e+03,
03596    1.11e+04, 2.98e+04, 4.62e+04, 1.08e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03597    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03598    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03599    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03600    0.653, 1.42e+03, 5.68e+03, 3.63e+03, 3.74e+03, 7.15e+03, 169, 6.19e+05,
03601    1.78e+06, 1.47e+05, 7.92e+04, 1.03e+05, 8.47e+04, 5.99e+04, 3.42e+04,
03602    1.47e+04, 5.11e+04, 1.21e+05, 1.36e+06, 5.68e+03, 1.81e+03, 4.24e+04,
03603    1.54e+05, 3.21e+05, 3.37e+05, 1.83e+03, 9.07e+03, 5.84e+04, 2.13e+05,
03604    2.63e+05, 3.05e+05, 1.19e+05, 1.19e+05, 1.23e+04, 4.58e+03, 8.68e+03,
03605    6.2e+03, 892, 474, 603, 139, 552, 1.38e+03, 2.18e+03, 150, 102, 374,
03606    1.05e+03, 2.64e+03, 5.38e+03, 1.22e+03, 563, 527, 2.28e+03, 180, 570, 0, 0,
03607    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

```
03608   {957, 3.96e+03, 68, 47.8, 1.96e+03, 915, 3.22e+03, 523, 998, 362, 300,
03609    1.82e+03, 304, 728, 636, 425, 271, 3.88e+03, 3.87e+03, 4.75e+03, 1.02e+04,
03610    3.9e+04, 3.33e+04, 1.41e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03611    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03612    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03613    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13.3,
03614    197, 566, 1.97e+03, 204, 8.86e+05, 9.91e+05, 2.39e+05, 1.2e+05, 7e+04,
03615    8.28e+04, 4.6e+04, 2.48e+04, 1.86e+04, 1.79e+04, 9.1e+04, 6.15e+05,
03616    4.09e+05, 975, 1.82e+05, 8.07e+04, 6.13e+05, 8.79e+05, 1.02e+04, 1.14e+04,
03617    6.4e+03, 5.91e+05, 1.54e+05, 2.07e+05, 1.64e+05, 2.57e+05, 2.09e+04,
03618    6.38e+04, 4.13e+04, 1.18e+04, 754, 387, 609, 566, 489, 881, 3.04e+03,
03619    5.3e+03, 3.9e+03, 4.09e+03, 1.48e+04, 1.17e+04, 2.12e+04, 4.52e+03, 853,
03620    571, 1.14e+03, 1.91e+03, 3.28e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03621    0, 0},
03622   {965, 3.96e+03, 76.5, 30.1, 1.91e+03, 831, 3.29e+03, 420, 680, 315, 55.1,
03623    1.28e+03, 345, 201, 964, 575, 493, 5.38e+03, 2.95e+03, 8.03e+03, 8.43e+03,
03624    3.62e+04, 4.13e+04, 1.46e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03625    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03626    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03627    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17.2,
03628    21.8, 39.1, 41.7, 165, 1.73e+06, 5.13e+05, 2.19e+05, 1.33e+05, 5.71e+04,
03629    1.89e+04, 2.97e+04, 4.96e+04, 2.5e+04, 1.9e+04, 9.65e+04, 1.89e+05,
03630    1.26e+06, 120, 6.17e+04, 8.83e+04, 8.31e+05, 1.16e+06, 4.51e+04, 4.71e+03,
03631    7.7e+04, 3.35e+05, 1.23e+05, 1.14e+05, 1.06e+05, 2.79e+05, 3.22e+04,
03632    8.23e+04, 4.26e+04, 4.28e+03, 541, 317, 584, 585, 954, 3.04e+03, 4.37e+03,
03633    5.62e+03, 1.59e+04, 1.97e+04, 2.26e+04, 2.73e+04, 9.27e+03, 4.62e+04, 728,
03634    618, 1.24e+03, 2.4e+03, 5.1e+03, 1.22, 23.9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03635    0, 0, 0},
03636   {978, 3.94e+03, 86.2, 17.7, 1.84e+03, 797, 3.08e+03, 447, 566, 380, 65.8,
03637    742, 305, 122, 306, 548, 407, 4.63e+03, 4.18e+03, 1.11e+04, 6.92e+03,
03638    2.55e+04, 2.83e+04, 2.38e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03639    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03640    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03641    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12.6,
03642    15.9, 35.4, 7.81e+03, 2.64e+04, 1.5e+06, 1.29e+06, 2.19e+05, 1.64e+05,
03643    3.8e+04, 1.38e+04, 1.66e+04, 2.46e+04, 2.98e+04, 3.23e+04, 1.4e+05,
03644    2.91e+05, 8.2e+05, 1.07e+04, 7.5e+04, 7.21e+04, 1.07e+06, 4.73e+05,
03645    9.57e+04, 3.07e+04, 3.76e+05, 1.17e+05, 1.66e+05, 2.44e+05, 4.02e+05,
03646    2.08e+05, 4.84e+04, 3.62e+04, 8.81e+03, 3.17e+03, 607, 381, 311, 1.46e+03,
03647    7.05e+03, 5.5e+03, 4.04e+03, 1.03e+04, 1.47e+04, 1.55e+04, 2.29e+04,
03648    5.46e+04, 6.27e+04, 5.23e+04, 744, 637, 791, 3e+03, 3.36e+03, 65.5, 8.69,
03649    0, 0, 0, 732, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03650   {982, 3.91e+03, 96.7, 10.7, 1.75e+03, 754, 2.79e+03, 557, 608, 367, 150,
03651    376, 220, 86.6, 136, 346, 510, 3.53e+03, 4.97e+03, 1.14e+04, 6.53e+03,
03652    2.09e+04, 2.48e+04, 2.17e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03653    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03654    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03655    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.16,
03656    104, 285, 1.06e+05, 1.51e+05, 1.13e+06, 1.05e+06, 2.58e+05, 3.13e+05,
03657    7.45e+04, 3.45e+04, 7.43e+04, 5.41e+04, 1.74e+04, 5.68e+04, 1.18e+05,
03658    2.86e+05, 4.73e+05, 6e+04, 1.26e+04, 2.43e+04, 2.71e+05, 1.9e+05, 3.06e+05,
03659    5.3e+04, 2.29e+05, 2.6e+05, 2.97e+05, 2.98e+05, 2.94e+05, 1.35e+05,
03660    1.22e+05, 2.87e+04, 1.06e+04, 7.52e+03, 1.93e+03, 369, 1.3e+03, 1.97e+04,
03661    1.15e+04, 4.35e+03, 6.45e+03, 9.12e+03, 1.56e+04, 1.1e+04, 2.69e+04,
03662    6.07e+04, 9.12e+04, 5.7e+04, 2.24e+03, 928, 1.1e+03, 5.47e+03, 3.78e+03,
03663    531, 0.136, 0, 0, 55.8, 4.24e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03664   {992, 3.9e+03, 107, 6.89, 1.65e+03, 715, 2.49e+03, 708, 634, 364, 200, 192,
03665    178, 21.7, 41.1, 714, 477, 3.27e+03, 4.63e+03, 8.45e+03, 6.55e+03,
03666    2.21e+04, 2.41e+04, 1.89e+05, 565, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03667    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03668    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03669    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 199, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03670    3.4e+03, 1.02e+04, 8.2e+03, 1.95e+04, 2.38e+05, 2.45e+05, 1.49e+06,
03671    8.27e+05, 2.3e+05, 1.64e+05, 9.34e+04, 2.9e+04, 3.95e+04, 2.64e+04,
03672    1.59e+04, 9.96e+04, 2.24e+05, 4.47e+04, 3.09e+05, 8.59e+04, 2.61e+04,
03673    4.86e+04, 2.53e+05, 1.24e+05, 1.41e+05, 3.12e+05, 3.09e+05, 1.71e+05,
03674    1.31e+05, 3.44e+05, 1.86e+05, 1.95e+05, 1.61e+05, 8.22e+03, 2.66e+04,
03675    1.33e+04, 2.63e+03, 1.17e+03, 1.44e+04, 1.26e+04, 8.93e+03, 8.21e+03,
03676    7.88e+03, 1.16e+04, 1.2e+04, 1.33e+04, 3.04e+04, 5.57e+04, 1.26e+05,
03677    6.72e+04, 1.02e+04, 1.58e+03, 1.19e+03, 5.84e+03, 4.19e+03, 3.83e+03, 0,
03678    0.471, 4.91, 641, 6.48, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03679   {1e+03, 3.88e+03, 119, 5.41, 1.55e+03, 725, 1.97e+03, 764, 617, 364, 217,
03680    92.6, 119, 14.9, 26.9, 1.18e+03, 612, 2.82e+03, 4.45e+03, 6.09e+03,
03681    5.67e+03, 2.38e+04, 2.27e+04, 2.37e+05, 341, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03682    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03683    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03684    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13.6, 2.69, 0, 42.5, 720, 1.88e+03,
03685    5.64, 0, 0, 0, 0, 0, 1.2e+03, 6.82e+04, 2.49e+05, 1.91e+05, 1.69e+05,
03686    1.69e+05, 4.91e+04, 4.84e+05, 2.02e+06, 3.46e+05, 2.05e+05, 6.19e+04,
03687    6.77e+04, 5.93e+04, 2.23e+04, 2.15e+04, 1.04e+05, 1.39e+05, 2.37e+03,
03688    2.11e+05, 2.01e+05, 3.59e+04, 6.83e+04, 6.77e+05, 3.13e+05, 2.56e+05,
03689    3.01e+05, 3.55e+05, 1.44e+05, 2.15e+04, 2.03e+05, 2.44e+05, 1.36e+05,
03690    2.42e+05, 1.93e+04, 3.06e+04, 2.37e+04, 3.02e+03, 3.05e+03, 1.19e+04,
03691    1.44e+04, 1.66e+04, 2.12e+04, 6.52e+03, 1.55e+04, 1.16e+04, 2.6e+04,
03692    3.38e+04, 6.55e+04, 5.34e+04, 4.64e+04, 1.3e+04, 957, 1.65e+03, 8.85e+03,
03693    7.99e+03, 5.05e+03, 0, 0, 6.77, 6.55e+03, 7.72e+03, 0, 0, 0, 0, 0, 0, 0,
03694    0},
```

```
03695   {1.01e+03, 3.87e+03, 131, 4.62, 1.44e+03, 773, 1.33e+03, 844, 696, 354, 228,
03696    37.2, 63.9, 5.99, 45.6, 1.45e+03, 739, 2.11e+03, 4.29e+03, 5.23e+03,
03697    4.52e+03, 2.26e+04, 3.16e+04, 2.4e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03698    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03699    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03700    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.83e+03, 31, 0, 0, 0, 0, 0, 0, 0, 751,
03701    563, 215, 3.11e+04, 1.12e+05, 2.9e+05, 3.46e+05, 3.47e+05, 2.57e+05,
03702    1.68e+05, 3.08e+05, 1.5e+05, 6.54e+05, 6.58e+05, 3.82e+05, 2.07e+05,
03703    1.15e+05, 2.4e+04, 2.53e+04, 2.7e+04, 4.63e+05, 1.17e+04, 1.78e+04,
03704    2.77e+05, 1.48e+05, 2.73e+04, 5.54e+04, 8.7e+05, 7.23e+05, 2.83e+05,
03705    1.79e+05, 2.65e+05, 3.87e+04, 1.07e+05, 9.87e+04, 2.11e+05, 9.34e+04,
03706    1.86e+05, 7.97e+04, 6.86e+04, 9.8e+03, 1.55e+04, 4.49e+03, 1.03e+04,
03707    1.23e+04, 1.76e+04, 1.91e+04, 9.01e+03, 1.09e+04, 1.42e+04, 1.86e+04,
03708    4.59e+04, 5.46e+04, 6.46e+04, 4.6e+04, 1.29e+04, 1.67e+03, 2.58e+03,
03709    1.2e+04, 7.04e+03, 1.75e+04, 0.00552, 0.614, 142, 3.19e+04, 6.05e+04, 0, 0,
03710    0, 0, 0, 0, 0, 0, 0},
03711   {1.03e+03, 3.85e+03, 142, 3.76, 1.31e+03, 839, 841, 1.06e+03, 870, 387, 246,
03712    15.5, 45.6, 13.8, 148, 1.44e+03, 795, 1.48e+03, 4.27e+03, 5.66e+03,
03713    3.51e+03, 1.85e+04, 3.47e+04, 1.69e+05, 0.0277, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03714    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03715    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03716    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.8e+04, 560, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03717    90.8, 1.59e+03, 6.45e+04, 6.54e+04, 1.13e+04, 1.48e+04, 9.96e+03, 1.24e+04,
03718    1.27e+04, 3.09e+05, 4.37e+05, 2.66e+05, 3.05e+05, 1.18e+06, 8.68e+05,
03719    2.31e+05, 1.18e+05, 6.67e+04, 6.15e+04, 3.12e+04, 5.71e+05, 1.73e+04,
03720    2.94e+05, 2e+05, 5.54e+05, 8.84e+04, 8.8e+04, 3.57e+05, 8.59e+04, 3.46e+05,
03721    3.07e+05, 8.37e+04, 3.59e+04, 8.83e+04, 4.64e+04, 1.59e+05, 2.01e+05,
03722    6.18e+04, 1.14e+05, 1.39e+05, 6.24e+03, 9.58e+03, 7.05e+03, 6.8e+03,
03723    8.2e+03, 1.5e+04, 1.47e+04, 1.12e+04, 1.55e+04, 1.2e+04, 2.75e+04,
03724    2.98e+04, 6.34e+04, 8.3e+04, 7.64e+04, 1.85e+03, 1.24e+03, 1.44e+03,
03725    1.41e+04, 4.47e+03, 1.25e+04, 5.9, 0, 4.38e+04, 4.95e+04, 9.44e+04, 0, 0,
03726    0, 0, 0, 0, 0, 0, 0},
03727   {1.05e+03, 3.83e+03, 156, 3.36, 1.17e+03, 894, 624, 1.3e+03, 1.03e+03, 450,
03728    235, 14.8, 56.4, 41.3, 379, 1.21e+03, 754, 1.22e+03, 4.32e+03, 6.32e+03,
03729    3.15e+03, 1.36e+04, 2.98e+04, 2.53e+05, 1.22e+03, 0, 0, 0, 0, 0, 0, 0, 0,
03730    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03731    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03732    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 94.6, 1.13e+03, 1.27e+04, 2.69e+05, 2.29e+03, 0,
03733    0, 0, 0, 0, 0, 0, 0, 0, 0, 30.6, 177, 1.06e+04, 1.73e+04, 4.96e+04, 6.99e+04,
03734    7.73e+04, 8.97e+04, 3.79e+04, 3.29e+04, 7.42e+04, 2.58e+05, 9.93e+05,
03735    9.8e+05, 4.59e+05, 1.97e+05, 1.2e+05, 8.62e+04, 6.34e+04, 1.87e+04,
03736    4.43e+05, 1.17e+05, 2.17e+05, 1.76e+05, 8.86e+05, 1.08e+05, 5.7e+04,
03737    9.54e+04, 1.21e+05, 1.49e+05, 2.54e+05, 1.72e+05, 8.08e+04, 5.62e+04,
03738    1.08e+05, 1.99e+05, 1.33e+05, 5.33e+04, 1.2e+05, 1.17e+05, 5.38e+03,
03739    1.94e+03, 6.23e+03, 1.22e+04, 1.36e+04, 7.38e+03, 1.45e+04, 1.82e+04,
03740    2.53e+04, 2.1e+04, 2.17e+04, 3.12e+04, 3.73e+04, 5.35e+04, 9.07e+04, 979,
03741    880, 1.33e+03, 2.26e+04, 6.84e+03, 7.53e+03, 45.7, 0, 7.98e+04, 3.27e+04,
03742    5.65e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03743   {1.06e+03, 3.82e+03, 167, 3.37, 1.05e+03, 893, 581, 1.22e+03, 1.11e+03, 444,
03744    373, 22.9, 74.8, 51.1, 801, 880, 700, 1.26e+03, 4.47e+03, 6.72e+03,
03745    4.23e+03, 1.21e+04, 2.06e+04, 2.33e+05, 3.57e+03, 0, 0, 0, 0, 0, 0, 0, 0,
03746    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03747    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03748    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.25e+03, 56.8, 6.33e+04, 6.45e+05, 3.46e+05, 0, 0,
03749    0, 0, 0, 0, 0, 0, 0, 0, 0, 0.36, 4.58e+03, 1.23e+05, 1.9e+05, 9.83e+04,
03750    7.5e+04, 6.89e+04, 1.68e+05, 1.28e+05, 2.61e+04, 1.71e+05, 5.17e+05,
03751    1.42e+06, 1.16e+06, 5.68e+05, 2.2e+05, 9.97e+04, 1.06e+05, 3.59e+04,
03752    2.08e+04, 6.29e+05, 3.53e+04, 7.65e+04, 1.92e+05, 6.73e+05, 4.76e+04,
03753    3.24e+04, 3.77e+04, 3.32e+04, 3.99e+05, 2.22e+05, 8.9e+04, 9.22e+04,
03754    3.9e+04, 1.11e+05, 1.07e+05, 1.03e+05, 9.03e+04, 1.54e+05, 5.12e+04,
03755    1.1e+04, 2.78e+03, 6.05e+03, 8.21e+03, 8.12e+03, 1.02e+04, 1.15e+04,
03756    1.04e+04, 3.43e+04, 3.18e+04, 2.61e+04, 2.35e+04, 2.5e+04, 4.7e+04,
03757    4.41e+04, 1.68e+03, 493, 1.34e+03, 2.91e+04, 5.67e+03, 1.16e+04, 17.9, 0,
03758    1.43e+04, 4.08e+04, 6.04e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03759   {1.09e+03, 3.79e+03, 180, 3.71, 912, 878, 565, 836, 980, 361, 722, 33.3, 76,
03760    105, 1.22e+03, 785, 670, 1.45e+03, 5.19e+03, 6.37e+03, 5.76e+03, 1.18e+04,
03761    1.24e+04, 1.99e+05, 849, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03762    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03763    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.3,
03764    83.1, 8.76e+03, 4.42e+04, 2.67e+05, 3.79e+05, 1.4e+03, 0, 0, 805, 6.99e+03,
03765    21.1, 66.4, 801, 211, 3.36e+04, 5.35e+04, 5.58e+04, 1.04e+05, 1.16e+05,
03766    9.92e+04, 1.03e+05, 5.76e+04, 3.37e+04, 9.36e+04, 9.66e+04, 3.81e+05,
03767    7.6e+05, 5.96e+05, 1.19e+06, 5.21e+05, 2.68e+05, 1.49e+05, 1.57e+05,
03768    9.74e+04, 2.92e+04, 2.91e+04, 3.57e+05, 1.25e+05, 6.6e+04, 2.54e+05,
03769    4.86e+05, 1.14e+04, 1.28e+04, 4.6e+04, 8.09e+04, 2.3e+05, 4.55e+04,
03770    6.86e+04, 1.17e+05, 3.4e+04, 4.23e+04, 1.52e+05, 9.09e+04, 9.79e+04,
03771    1.12e+05, 4.02e+04, 5.58e+03, 3.47e+03, 4.42e+03, 3.58e+03, 7.62e+03,
03772    7.91e+03, 4.8e+03, 1.17e+04, 3.32e+04, 2.63e+04, 1.68e+04, 1.75e+04,
03773    3.2e+04, 3.61e+04, 4.48e+04, 2.48e+03, 344, 1.48e+03, 1.11e+04, 7.09e+03,
03774    6.9e+03, 14.9, 0, 2.64e+04, 8.47e+04, 10.2, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03775   {1.11e+03, 3.76e+03, 191, 3.65, 793, 786, 541, 487, 832, 453, 1.08e+03,
03776    53.4, 130, 205, 542, 658, 668, 1.77e+03, 5.68e+03, 6e+03, 6.54e+03,
03777    1.29e+04, 3.51e+04, 2.43e+05, 314, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03778    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03779    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03780    792, 2.13e+03, 508, 7.68e+04, 2.35e+05, 2.67e+05, 194, 0, 0, 90.9,
03781    6.47e+03, 3.47e+04, 2.01e+04, 1.64e+04, 3.07e+04, 8.4e+04, 6.15e+04,
```

```
03782    8.75e+04, 1.38e+05, 8.54e+04, 6.14e+04, 1.08e+05, 8.75e+04, 1.07e+05,
03783    1.3e+05, 1.49e+05, 2.1e+05, 2.05e+05, 3.46e+05, 3.84e+05, 5.5e+05,
03784    4.45e+05, 2.97e+05, 2.33e+05, 1.78e+05, 6.01e+04, 2.92e+04, 5.96e+04,
03785    1.54e+05, 1.41e+05, 6.14e+04, 9.42e+05, 1.19e+04, 1.77e+04, 2.08e+04,
03786    7.01e+04, 1.06e+05, 7.94e+04, 1.08e+05, 6.13e+04, 8.38e+04, 3.83e+04,
03787    3.5e+04, 1.33e+05, 8.67e+04, 9.13e+04, 8.96e+04, 4.99e+04, 3.54e+03,
03788    6.06e+03, 3.05e+03, 4.88e+03, 3.78e+03, 1.83e+03, 2.37e+03, 1.67e+04,
03789    2.62e+04, 1.66e+04, 1.42e+04, 1.91e+04, 3.27e+04, 5.63e+04, 2.04e+04,
03790    5.55e+03, 211, 1.38e+03, 2.91e+03, 7.4e+03, 3.29e+03, 2.47, 0.0754,
03791    8.49e+04, 1.02e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03792   {1.13e+03, 3.74e+03, 204, 4.07, 669, 674, 518, 356, 835, 469, 1.12e+03, 133,
03793    163, 176, 258, 390, 599, 2.45e+03, 5.15e+03, 5.4e+03, 6.36e+03, 1.38e+04,
03794    3.38e+04, 1.2e+05, 1.62e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03795    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03796    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 432, 319,
03797    4.8e+04, 2.19e+05, 1.67e+05, 3.51e+04, 264, 0, 940, 2.39e+03, 1.65e+04,
03798    1.87e+04, 3.82e+04, 315, 29.4, 2.44e+04, 1.73e+04, 1.2e+03, 4.46e+03, 896,
03799    6.87e+03, 4.14e+04, 3.6e+04, 5.29e+04, 1.19e+05, 8.02e+04, 1.28e+05,
03800    2.03e+05, 1.38e+05, 2.23e+05, 3.42e+05, 4.61e+05, 2.32e+05, 1.22e+05,
03801    1.53e+05, 1.06e+05, 2.91e+04, 4.8e+04, 2.25e+05, 1.99e+05, 6.01e+04,
03802    1.4e+05, 6.25e+05, 2.28e+04, 1.03e+04, 2.48e+03, 8.02e+04, 1.62e+05,
03803    1.48e+05, 1.26e+05, 1.11e+05, 7.24e+04, 6.1e+04, 6.31e+04, 1.34e+05,
03804    9.73e+04, 1.46e+05, 8.12e+04, 3.62e+03, 7.37e+03, 5.07e+03, 3.22e+03,
03805    5.01e+03, 1.54e+03, 1.5e+03, 3.46e+03, 1.35e+04, 1.56e+04, 1.4e+04,
03806    1.56e+04, 3.07e+04, 4.53e+04, 2.88e+04, 1.29e+04, 1.23e+04, 421, 1.36e+03,
03807    2.66e+03, 1.08e+04, 2.42e+03, 0, 638, 1.3e+04, 5.21e+03, 0, 0, 0, 0, 0, 0,
03808    0, 0, 0, 0},
03809   {1.16e+03, 3.72e+03, 217, 4.36, 542, 597, 506, 278, 579, 156, 1.48e+03, 123,
03810    129, 154, 239, 361, 579, 3.34e+03, 4.2e+03, 4.7e+03, 5.67e+03, 1.39e+04,
03811    2.77e+04, 8.19e+04, 4.49e+04, 3.61e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03812    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03813    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03814    2.62e+03, 3.59e+05, 2.27e+05, 7.15e+04, 650, 163, 241, 8.16e+04, 1.98e+05,
03815    5.86e+04, 5.82e+03, 0, 0, 0, 0, 0, 3.18e+03, 2.78e+04, 2.57e+04, 4.71e+04,
03816    9.31e+04, 5.8e+04, 1.29e+05, 5.79e+04, 6.78e+04, 8.89e+04, 7.48e+04,
03817    1.78e+05, 1.29e+05, 1.74e+05, 3.11e+05, 3.31e+05, 3.43e+05, 2.22e+05,
03818    1.73e+05, 1.18e+05, 8.35e+04, 7.21e+04, 6.07e+04, 1.88e+05, 1.41e+05,
03819    2.63e+05, 1.1e+05, 1.73e+04, 1.14e+03, 847, 1.16e+05, 8.17e+04, 9.48e+04,
03820    1.26e+05, 1.2e+05, 4.72e+04, 7.68e+04, 4.5e+04, 6.36e+04, 1.28e+05,
03821    2.01e+05, 1.3e+04, 3.36e+03, 5.58e+03, 5.81e+03, 4.47e+03, 3.1e+03,
03822    1.1e+03, 1.02e+03, 2.76e+03, 2.38e+04, 1.04e+04, 9.5e+03, 1.85e+04,
03823    1.3e+04, 1.42e+04, 8.29e+03, 1.58e+04, 2.61e+04, 706, 1.84e+03, 2.26e+03,
03824    1.19e+04, 2.24e+03, 197, 8.88e+03, 2.34e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03825    0},
03826   {1.18e+03, 3.7e+03, 228, 4.31, 413, 553, 501, 156, 292, 270, 1.05e+03, 167,
03827    65.3, 186, 227, 662, 857, 4.02e+03, 3.61e+03, 4.04e+03, 5.07e+03, 1.1e+04,
03828    2.08e+04, 2.76e+04, 1.85e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03829    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03830    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 132, 6.08e+03,
03831    2.66e+05, 1.86e+05, 3.08e+03, 1.26e+03, 124, 390, 4.5e+04, 1.04e+05,
03832    1.26e+05, 6.29e+04, 1.23e+03, 0, 0, 0, 0, 443, 2.38e+04, 5.1e+04, 8.04e+03,
03833    4.42e+04, 6e+04, 1.38e+04, 1.16e+05, 8.26e+04, 5.6e+04, 5.78e+04, 1.74e+05,
03834    1.98e+05, 6.48e+04, 1.62e+05, 3.26e+05, 4.17e+05, 9.4e+05, 9.83e+05,
03835    4.52e+05, 2.43e+05, 4.42e+04, 4.98e+04, 6.76e+04, 3.14e+05, 2.36e+05,
03836    6.28e+05, 9.76e+04, 5.73e+03, 1.19e+04, 7.68e+03, 5.79e+04, 7.18e+04,
03837    1.13e+05, 8.6e+04, 1.1e+05, 2.95e+04, 4.09e+04, 4.46e+04, 1.01e+05,
03838    2.84e+05, 5.62e+04, 1.74e+03, 4.19e+03, 3.56e+03, 8.49e+03, 2.73e+03,
03839    4.06e+03, 1.59e+03, 2.64e+03, 3.98e+03, 1.5e+04, 8.51e+03, 8.16e+03,
03840    1.36e+04, 1.54e+04, 5.96e+03, 2.89e+03, 4.62e+03, 2.74e+04, 656, 1.67e+03,
03841    1.91e+03, 1.5e+04, 2.76e+03, 1.22e+03, 8.78e+03, 5.35e+04, 0, 0, 0, 0, 0,
03842    0, 0, 0, 0, 0, 0},
03843   {1.2e+03, 3.66e+03, 238, 4.06, 288, 558, 536, 41.6, 394, 265, 409, 406,
03844    57.1, 181, 381, 645, 1.4e+03, 3.91e+03, 2.55e+03, 3.66e+03, 3.92e+03,
03845    6.28e+03, 1.85e+04, 1e+04, 2.34e+05, 783, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03846    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03847    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03848    1.42e+04, 2.9e+05, 6.02e+04, 844, 4.68e+03, 44.2, 10.8, 6.87e+03, 7.86e+03,
03849    3.25e+04, 6.02e+04, 5.24e+03, 0, 0, 0, 0.00111, 1.51e+03, 9.04e+04,
03850    1.96e+04, 1.19e+04, 633, 572, 3.09e+03, 2.14e+04, 1.89e+05, 7.61e+04,
03851    7.39e+04, 1.15e+05, 2.91e+05, 7.89e+04, 4.15e+04, 2.55e+05, 3.63e+05,
03852    3.81e+05, 5.18e+05, 4.93e+05, 1.55e+06, 3.83e+05, 5.08e+04, 9.14e+04,
03853    1.53e+05, 2.78e+05, 2e+05, 9.38e+04, 5.16e+03, 2.81e+04, 3.07e+04,
03854    1.39e+04, 3.32e+04, 1.08e+05, 2.52e+04, 2.78e+04, 4.6e+04, 3.06e+04,
03855    4.97e+04, 6.05e+04, 6.3e+04, 1.18e+05, 4.08e+03, 7.81e+03, 6.54e+03,
03856    3.67e+03, 1.02e+04, 1.21e+04, 4.09e+03, 1.19e+03, 1.33e+03, 2.05e+03,
03857    6.69e+03, 1.02e+04, 5.4e+03, 9.22e+03, 4.96e+03, 3.63e+03, 2.5e+03,
03858    4.34e+03, 1.37e+04, 2.81e+03, 786, 1.31e+03, 1.33e+04, 6.25e+03, 4.84e+03,
03859    9.33e+03, 6.47e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03860   {1.22e+03, 3.63e+03, 243, 3.94, 175, 602, 417, 36.1, 603, 182, 316, 291,
03861    59.1, 177, 1.09e+03, 519, 2.06e+03, 3.84e+03, 2.11e+03, 3.44e+03, 2.99e+03,
03862    4.15e+03, 1.69e+04, 5.17e+04, 1.32e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03863    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03864    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.49e+03,
03865    1.57e+05, 1.23e+05, 751, 445, 80.2, 59.1, 110, 3.08e+03, 835, 0, 0, 0, 0,
03866    0, 0, 503, 4.48e+04, 8.77e+04, 124, 1.14e+04, 1.06e+03, 1.31e+03, 8.18e+03,
03867    2.79e+04, 2.16e+05, 9.1e+04, 1.24e+05, 3.61e+05, 2.68e+05, 6.04e+04,
03868    9.16e+04, 2.68e+04, 6.5e+04, 3.03e+05, 1.11e+05, 5.87e+03, 5.26e+05,
```

```
03869    8.82e+05, 5.82e+05, 3.05e+05, 1.31e+05, 1.07e+05, 1.19e+05, 1.69e+04, 958,
03870    1.89e+04, 5.11e+04, 4.11e+04, 5.59e+04, 1.12e+05, 1.4e+04, 1.63e+04,
03871    4.18e+04, 3.36e+04, 3.43e+04, 6.29e+04, 7.49e+04, 5.71e+04, 8.24e+03,
03872    2.22e+04, 1.21e+04, 6.58e+03, 6.44e+03, 4.43e+03, 6.12e+03, 754, 718,
03873    1.85e+03, 1.12e+04, 1.2e+04, 7.1e+03, 8.46e+03, 3.05e+03, 5.6e+03,
03874    3.08e+03, 4.77e+03, 6.25e+03, 9.87e+03, 988, 1.05e+03, 9.51e+03, 1.01e+04,
03875    4.97e+03, 5.63e+03, 6.61e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03876   {1.25e+03, 3.58e+03, 251, 3.9, 89, 664, 203, 103, 519, 176, 152, 388, 66.2,
03877    197, 1.25e+03, 797, 1.97e+03, 3.69e+03, 1.74e+03, 3.26e+03, 2.62e+03,
03878    3.95e+03, 1.66e+04, 2.87e+04, 1.76e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03879    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03880    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00881,
03881    8.45e+03, 5.13e+04, 523, 39.3, 551, 31.3, 1.68e+03, 60.4, 0, 0, 0, 0, 0, 0,
03882    1.21, 0.248, 186, 1.2e+03, 677, 1.83e+03, 1.33e+04, 1.52e+03, 1.14e+03,
03883    2.88e+03, 1.15e+05, 1.54e+05, 1.48e+05, 1.38e+05, 6.53e+04, 2.15e+05,
03884    7.69e+04, 9.96e+04, 1.22e+05, 7.01e+04, 1.82e+05, 7.97e+03, 4.58e+03,
03885    6.48e+03, 1.58e+05, 3.14e+05, 2.01e+05, 7.37e+04, 2.39e+04, 6.31e+04,
03886    7.21e+03, 4.37e+03, 1.87e+04, 3.44e+04, 3.86e+04, 2.81e+04, 1.32e+04,
03887    9.13e+03, 6.48e+03, 2.55e+04, 1.86e+04, 3.89e+04, 4.08e+04, 9.9e+04,
03888    3.94e+04, 1.72e+04, 1.78e+04, 2.39e+04, 1.16e+04, 7.14e+03, 4.65e+03,
03889    4.83e+03, 1.3e+03, 1.09e+03, 1.36e+03, 5.01e+03, 1.53e+04, 1.15e+04,
03890    7.13e+03, 3.79e+03, 6.59e+03, 1.31e+04, 9.47e+03, 5.29e+03, 1.29e+04, 703,
03891    4.01e+03, 1.1e+04, 7.97e+03, 6.87e+03, 3.49e+03, 2.54e+04, 0, 0, 0, 0, 0,
03892    0, 0, 0, 0, 0, 0},
03893   {1.29e+03, 3.54e+03, 258, 3.75, 34.9, 699, 43.8, 268, 282, 291, 51, 458,
03894    63.7, 404, 1.53e+03, 1.31e+03, 1.2e+03, 3.96e+03, 1.68e+03, 2.53e+03,
03895    2.79e+03, 3.94e+03, 1.96e+04, 6.14e+04, 5.59e+04, 0, 0, 0, 0, 0, 0, 0, 0,
03896    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03897    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 54.7, 20.5,
03898    0, 0, 5.2e+04, 1.13e+04, 1.61e+03, 19.2, 955, 2.13e+03, 0, 0, 0, 0, 89.6,
03899    0, 0, 0, 0, 0, 0, 0.13, 0.221, 236, 1.24e+03, 1.58e+03, 2.96e+03, 1.33e+05,
03900    2.76e+04, 3.37e+04, 9.91e+04, 1.65e+03, 2.8e+03, 6.83e+03, 4.75e+04,
03901    8.78e+04, 8.89e+04, 8.41e+04, 4.75e+04, 4.15e+04, 8.21e+04, 7.03e+04,
03902    6.37e+03, 3.87e+03, 5.33e+04, 1e+05, 1.02e+05, 8.31e+04, 4.93e+04,
03903    3.91e+04, 2.8e+04, 1.66e+05, 1.62e+04, 5.36e+04, 1.99e+04, 3.62e+04,
03904    1.48e+04, 4.27e+03, 3.79e+03, 1.83e+04, 1.9e+04, 2.35e+04, 1.84e+04,
03905    5.12e+04, 7.76e+04, 2.24e+04, 1.89e+04, 2.54e+04, 2.95e+04, 1.03e+04,
03906    2.59e+03, 4e+03, 4.73e+03, 1.12e+03, 1.46e+03, 3.6e+03, 1.82e+04, 6.12e+03,
03907    6.04e+03, 1.35e+04, 2.64e+03, 2.48e+04, 2.05e+04, 1.02e+04, 9.05e+03, 497,
03908    6.22e+03, 7.78e+03, 1.71e+04, 8.41e+03, 113, 0.0178, 0, 0, 0, 0, 0, 0, 0,
03909    0, 0, 0, 0},
03910   {1.33e+03, 3.49e+03, 261, 3.52, 10.4, 635, 16.3, 453, 125, 695, 36.2, 279,
03911    182, 518, 1.73e+03, 1.18e+03, 773, 3.62e+03, 1.9e+03, 1.93e+03, 3.33e+03,
03912    3.95e+03, 3.45e+04, 1.42e+05, 1.47e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03913    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03914    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.79e+05,
03915    1.81e+03, 0, 0, 375, 0, 0, 0, 26.5, 0, 0, 0, 0, 0, 0, 0, 0, 16.4, 3.93e+03,
03916    5.25e+04, 1.48e+04, 4.71e+04, 1.26e+05, 1.45e+05, 4.39e+04, 3.37, 0,
03917    0.0571, 1.38e+04, 2.71e+04, 3.45e+04, 5.74e+04, 4.45e+04, 3.71e+04,
03918    3.76e+04, 4.11e+04, 1.27e+05, 3.3e+04, 2.01e+04, 8.19e+04, 2.17e+05,
03919    2.04e+05, 9.34e+04, 3.03e+04, 2.3e+04, 6.96e+03, 1.84e+04, 4.03e+04,
03920    1.38e+05, 6.83e+04, 1.27e+04, 9.55e+03, 6.26e+03, 8.18e+03, 1.07e+04,
03921    6.22e+04, 6.9e+04, 2.68e+04, 1.66e+04, 7.42e+04, 8.92e+04, 2.08e+04,
03922    1.22e+04, 3.22e+04, 1.16e+04, 2.75e+03, 1.93e+03, 2.64e+03, 940, 1.47e+03,
03923    3.11e+03, 1.29e+04, 7.93e+03, 4.67e+03, 1.16e+04, 3.31e+03, 1.9e+04,
03924    1.1e+04, 1.24e+04, 6.94e+03, 912, 2.15e+03, 1.35e+04, 3.43e+04, 2.1e+03,
03925    15.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03926   {1.36e+03, 3.44e+03, 261, 2.94, 3.85, 519, 30, 517, 90, 785, 117, 147, 134,
03927    476, 1.4e+03, 1.29e+03, 759, 2.97e+03, 1.86e+03, 1.7e+03, 3.7e+03,
03928    4.48e+03, 4.62e+04, 2.12e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03929    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03930    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.53e+04, 2.36e+05,
03931    413, 0, 0, 1.53e+03, 0, 0, 0, 37.6, 0, 0, 1.09e+03, 0, 0, 0, 0, 0, 0,
03932    6.66e+04, 9.28e+04, 3.77e+04, 7.45e+04, 1.77e+05, 8.98e+04, 1.45, 0, 0, 0,
03933    1.36e+03, 4.08e+04, 1.49e+04, 3.82e+04, 4.82e+04, 4.71e+04, 3.1e+04,
03934    4.15e+04, 9.72e+04, 1.92e+05, 4.35e+04, 1.19e+05, 1.53e+05, 4.94e+05,
03935    3.88e+04, 1.56e+04, 1.08e+04, 2.04e+03, 3.67e+03, 9.45e+03, 1.31e+04,
03936    2.51e+04, 7.11e+03, 2.02e+04, 5.2e+03, 8.55e+03, 7.94e+03, 5.32e+04,
03937    6.18e+04, 5.5e+04, 2.76e+04, 5.2e+04, 6.37e+04, 1.47e+05, 2.62e+04,
03938    3.94e+04, 9.97e+03, 6.16e+03, 4.31e+03, 2.61e+03, 1.11e+03, 1.35e+03,
03939    2.93e+03, 5.25e+03, 5.75e+03, 9.63e+03, 2.09e+04, 4.01e+03, 1.07e+04,
03940    7.48e+03, 1.35e+04, 7.98e+03, 883, 2.06e+03, 2.71e+04, 4.75e+04, 879, 8.88,
03941    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03942   {1.39e+03, 3.37e+03, 260, 2.76, 3.49, 363, 39.8, 520, 97.5, 658, 212, 163,
03943    44, 537, 1.18e+03, 1.57e+03, 770, 2.35e+03, 1.35e+03, 1.82e+03, 3.8e+03,
03944    7.09e+03, 5.17e+04, 2.13e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03945    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03946    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.75e+03, 9.38e+04,
03947    0.00111, 0, 0, 0, 639, 141, 1.61e+03, 4.96e+03, 601, 0.00442, 0, 0, 0, 0,
03948    0, 0, 0, 1.04e+04, 2.59e+05, 8.31e+04, 7.81e+04, 5.74e+04, 151, 0,
03949    3.99e+03, 7.24e+04, 683, 1.51, 3.64e+03, 1.38e+04, 2.43e+04, 7.92e+04,
03950    5.18e+04, 3.11e+04, 6.77e+04, 6.5e+04, 8.8e+04, 1.41e+05, 2.4e+05,
03951    1.52e+05, 2.79e+05, 1.65e+05, 3.29e+04, 1.79e+04, 9.19e+03, 2.01e+03,
03952    1.73e+04, 5.05e+04, 3.43e+04, 6.43e+03, 6.71e+03, 6.19e+03, 2.15e+04,
03953    1.89e+04, 2.46e+04, 6.55e+04, 6.26e+04, 3.03e+04, 3.62e+04, 1.09e+05,
03954    6.52e+04, 1.99e+05, 1.34e+05, 8.96e+04, 7.43e+03, 3.73e+03, 2.07e+03, 947,
03955    770, 2.02e+03, 4.64e+03, 6.79e+03, 7.02e+03, 2.91e+04, 4.84e+03, 7.31e+03,
```

```
03956    1.86e+04, 1.56e+04, 8.11e+03, 323, 1.25e+03, 4.78e+04, 3.32e+04, 1.73e+03,
03957    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03958   {1.43e+03, 3.3e+03, 258, 3.14, 4.6, 210, 42.2, 578, 137, 550, 260, 244,
03959    61.3, 626, 1.03e+03, 1.71e+03, 586, 1.78e+03, 997, 1.94e+03, 3.96e+03,
03960    1.27e+04, 5.92e+04, 1.37e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03961    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03962    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.82e+04, 2.25e+05,
03963    41.8, 0, 0, 240, 702, 5.93e+03, 8.61e+03, 2.96e+04, 7.32e+03, 0, 0, 0, 0,
03964    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.26e+03, 5.66e+04, 2.4e+03, 1.14e+03,
03965    1.07e+04, 2.07e+04, 7.66e+04, 3.68e+04, 1.33e+05, 5.31e+04, 2.86e+04,
03966    4.24e+04, 1.65e+05, 2.21e+05, 2.6e+05, 8.98e+04, 9.36e+04, 6.2e+04,
03967    5.85e+04, 2.34e+04, 8.89e+03, 8.39e+03, 2.45e+04, 5.56e+04, 3.05e+04,
03968    5.72e+03, 1.33e+04, 5.76e+03, 1.17e+04, 9.48e+03, 1.59e+04, 4.72e+04,
03969    4.99e+04, 3.78e+04, 2.7e+04, 4.4e+04, 2.44e+05, 3.06e+05, 2.55e+05,
03970    3.8e+04, 5.1e+04, 5.81e+03, 1.37e+03, 1.99e+03, 1.23e+03, 1.7e+03, 3.2e+03,
03971    7.37e+03, 2.02e+04, 6.92e+03, 5.93e+03, 7.2e+03, 1.83e+04, 1.16e+04,
03972    6.26e+03, 202, 333, 5.7e+04, 3.89e+04, 445, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03973    0, 0, 0},
03974   {1.47e+03, 3.23e+03, 250, 4.24, 7.25, 100, 40, 601, 435, 563, 224, 314, 84,
03975    425, 905, 1.58e+03, 479, 1.69e+03, 934, 1.72e+03, 6.32e+03, 1.96e+04,
03976    5.58e+04, 6.84e+04, 6.78e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03977    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03978    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.03e+04, 1.65e+04, 276,
03979    0, 0, 1.75e+03, 6.87e+03, 2.25e+04, 1.8e+04, 6.67e+03, 1.09e+03, 0.0238, 0,
03980    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 103, 447, 35.8, 5.7e+04,
03981    3.16e+04, 3.72e+04, 9.47e+04, 1.49e+05, 6.25e+04, 7.98e+04, 4.31e+04,
03982    1.63e+05, 2e+05, 1.65e+05, 6.5e+04, 7.6e+04, 6.44e+04, 1.34e+05, 6.56e+04,
03983    9.35e+04, 3.67e+04, 1.17e+04, 1.07e+05, 2.55e+04, 6e+03, 5e+03, 5.21e+03,
03984    5.65e+03, 8.92e+03, 1.12e+04, 2.64e+04, 5.75e+04, 2.65e+04, 1.98e+04,
03985    3.17e+04, 1.74e+05, 2.45e+05, 3.03e+05, 6.57e+04, 5.88e+04, 6.82e+03,
03986    1.99e+03, 1.98e+03, 2.53e+03, 2.28e+03, 2.56e+03, 5.56e+03, 1.51e+04,
03987    3.75e+03, 5.43e+03, 5.11e+03, 9.38e+03, 1.79e+04, 1.58e+03, 347, 131,
03988    4.84e+04, 3.67e+04, 720, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
03989   {1.52e+03, 3.17e+03, 243, 6.73, 11.9, 47.6, 26.3, 375, 1.05e+03, 541, 302,
03990    379, 112, 236, 792, 1.52e+03, 305, 1.77e+03, 1.05e+03, 1.6e+03, 1.32e+04,
03991    2.29e+04, 7.02e+04, 3.62e+04, 1.95e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03992    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
03993    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.3e+05,
03994    6.08e+03, 0, 0, 0.141, 2.61e+03, 2.35e+03, 4.21e+04, 1.06e+04, 1.6e+04,
03995    2.18e+04, 1.18e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00999, 0, 0, 0,
03996    0, 0, 2.15e+04, 2.39e+04, 6.68e+04, 9.63e+04, 5.51e+04, 1.62e+04, 3.68e+04,
03997    3.07e+03, 1.11e+03, 3.02e+04, 6.04e+04, 3.87e+04, 1.25e+05, 1.17e+05,
03998    7.55e+04, 1.38e+05, 9.4e+04, 8.39e+04, 2.99e+04, 9.45e+04, 2.08e+04,
03999    7.26e+03, 5.56e+03, 5.34e+03, 5.86e+03, 7.91e+03, 1.3e+04, 1.5e+04,
04000    4.68e+04, 2.96e+04, 1.07e+04, 1e+04, 2.58e+04, 8.26e+04, 2.01e+05,
04001    9.63e+04, 4.22e+04, 1.51e+04, 4.07e+03, 3.83e+03, 1.52e+03, 2.35e+03,
04002    2.44e+03, 7.61e+03, 1.14e+04, 3.33e+03, 8.09e+03, 6.26e+03, 9.82e+03,
04003    4.02e+03, 3.06e+03, 350, 652, 1.39e+04, 2.03e+04, 19.5, 0, 0, 0, 0, 0, 0,
04004    0, 0, 0, 0, 0, 0, 0},
04005   {1.57e+03, 3.09e+03, 234, 9.03, 18.9, 32.6, 8.99, 418, 1.48e+03, 478, 225,
04006    483, 112, 164, 599, 1.38e+03, 213, 1.58e+03, 1.29e+03, 3.03e+03, 1.86e+04,
04007    2.96e+04, 8.39e+04, 7.36e+04, 2.17e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04008    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04009    0, 0, 79.3, 0.00444, 9.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04010    2.87e+05, 1.79e+03, 79.5, 0, 0, 954, 753, 4.72e+04, 6.05e+04, 6.94e+04,
04011    2.48e+04, 1.09e+04, 214, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04012    6.15e+03, 6.78e+03, 3.93e+04, 9.94e+04, 6.78e+04, 1.47e+04, 3.2e+03,
04013    4.78e+03, 593, 1.06e+04, 1.17e+04, 4.27e+03, 2.55e+04, 7.57e+04, 3.73e+04,
04014    4.96e+04, 5.98e+04, 1.36e+05, 5.24e+04, 3.78e+04, 1.74e+04, 3.47e+03,
04015    1.08e+04, 7.98e+03, 9.98e+03, 6.65e+03, 6.67e+03, 9.84e+03, 2.38e+04,
04016    2.6e+04, 2.25e+04, 5.8e+03, 2.45e+04, 4.78e+04, 1.66e+05, 1.1e+05,
04017    5.37e+04, 6.14e+04, 8.02e+03, 3.07e+03, 1.39e+03, 2.31e+03, 1.43e+03,
04018    7.03e+03, 7.71e+03, 4.7e+03, 6.43e+03, 5.21e+03, 3.3e+03, 2.24e+03,
04019    5.34e+03, 451, 1.19e+03, 2.06e+03, 2.65e+03, 0.233, 0, 0, 0, 0, 0, 0, 0, 0,
04020    0, 0, 0, 0, 0, 0},
04021   {1.61e+03, 3.02e+03, 223, 12.2, 24.8, 28.8, 16.3, 1.48e+03, 1.98e+03, 525,
04022    130, 667, 82.5, 109, 380, 1.27e+03, 204, 1.42e+03, 1.82e+03, 5.82e+03,
04023    2.07e+04, 2.19e+04, 1.02e+05, 8.81e+04, 7.09e+04, 0, 0, 0, 0, 0, 0, 0, 0,
04024    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04025    0, 0, 0, 0, 0, 5.54e+03, 4.12e+03, 436, 598, 881, 1.5e+03, 1.86e+03, 0, 0, 0,
04026    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.01e+03, 2.43e+05, 1.59e+03, 0, 0, 0, 667, 370,
04027    2.79e+04, 5.98e+04, 1.08e+05, 8.26e+04, 3.17e+04, 1.54e+04, 67.3, 0, 0, 0,
04028    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.46e+03, 1.54e+04, 4.73e+04,
04029    6.48e+04, 1.21e+05, 6.44e+04, 4.69e+04, 4.19e+04, 9.78e+03, 3.69e+03,
04030    1.37e+04, 7.68e+03, 1.48e+03, 877, 1.65e+05, 2.23e+05, 1.39e+05, 2.87e+05,
04031    9.11e+04, 4.19e+04, 9.06e+03, 3.09e+03, 3.01e+03, 1.17e+04, 6.03e+03,
04032    4.77e+03, 7.56e+03, 6.71e+03, 8.27e+03, 1.08e+04, 2.23e+04, 1.43e+04,
04033    1.65e+04, 4.76e+04, 2.12e+05, 1.42e+05, 5e+04, 3.97e+04, 2.42e+04,
04034    3.44e+03, 1.03e+03, 2.54e+03, 2.45e+03, 4.62e+03, 5.69e+03, 2.35e+03,
04035    6.65e+03, 6.82e+03, 2.31e+03, 1.23e+03, 4.02e+03, 412, 198, 48.3, 80.5, 0,
04036    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04037   {1.66e+03, 2.95e+03, 215, 17, 29.9, 35.8, 38.3, 1.45e+03, 2.02e+03, 748,
04038    57.4, 585, 156, 65.9, 165, 1.38e+03, 184, 1.56e+03, 2.49e+03, 6.98e+03,
04039    1.68e+04, 2.32e+04, 8.51e+04, 4.57e+04, 8.57, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04040    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.51e+03, 0,
04041    0, 1.07e+03, 8.27e+03, 1.09e+04, 1.41e+03, 1.78e+03, 3.16e+03, 3.74e+03,
04042    3.4e+03, 2.79e+03, 684, 5.24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 927,
```

```
04043    1.36e+05, 153, 0, 0, 1.11e+04, 1.8e+04, 5.4e+03, 7.19e+03, 6.29e+04,
04044    1.16e+05, 1.65e+05, 2.53e+05, 1.95e+05, 2.35e+04, 0, 0, 0, 0, 0, 0, 0,
04045    0, 0, 0, 0, 0, 0, 0, 0, 3.63e+04, 3.37e+04, 2.81e+04, 2.18e+04,
04046    1.91e+04, 9.42e+03, 8.71e+04, 1.79e+04, 2.66e+04, 1.68e+04, 109, 92.6, 168,
04047    99.7, 138, 251, 1.22e+05, 1.56e+05, 1.11e+05, 6.12e+03, 5.76e+03, 6.33e+03,
04048    1.69e+04, 2.51e+04, 6.46e+03, 3.26e+03, 3.22e+03, 3.54e+03, 5.32e+03,
04049    1.69e+04, 2.58e+04, 2.18e+04, 1.24e+05, 1.99e+05, 9.74e+04, 5.32e+04,
04050    2.98e+04, 4.54e+04, 6.66e+03, 1.44e+03, 2.81e+03, 1.87e+03, 3.57e+03,
04051    4.15e+03, 1.74e+03, 6.91e+03, 2.87e+03, 1.08e+03, 952, 716, 447, 413, 0, 0,
04052    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04053   {1.71e+03, 2.89e+03, 205, 23.2, 34.7, 59.9, 51.9, 1.15e+03, 1.8e+03,
04054    1.02e+03, 99.4, 128, 152, 43, 34, 1.37e+03, 170, 2.13e+03, 2.88e+03,
04055    6.7e+03, 1.46e+04, 2.96e+04, 5.62e+04, 4.04e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04056    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.32e+03,
04057    6.7e+03, 1.75e+04, 1.82e+04, 7.82e+03, 998, 1.38e+03, 1.33e+03, 785,
04058    1.66e+03, 2.59e+03, 3.34e+03, 8.88e+03, 1.11e+04, 3.51e+03, 0, 0, 0, 0, 0,
04059    0, 0, 0, 0, 0, 0, 1.17e+04, 1.3e+05, 0.592, 0, 0, 285, 4.9e+04, 2.51e+04,
04060    2.04e+04, 5.2e+03, 1.58e+05, 9.79e+04, 1.39e+05, 1.66e+05, 1.6e+05,
04061    2.6e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.12e+04,
04062    4.05e+04, 3.7e+04, 2.43e+04, 4.35e+04, 1.66e+04, 7.19e+03, 2.75e+04,
04063    8.98e+04, 2.17e+04, 124, 90.6, 77.8, 672, 108, 77.6, 224, 1.34e+05,
04064    1.13e+05, 1.24e+04, 3.28e+03, 1.1e+04, 9.8e+03, 1.76e+04, 5.06e+03,
04065    3.59e+03, 4.39e+03, 2.66e+03, 1.02e+04, 1.51e+04, 2.45e+04, 2.4e+04,
04066    2.11e+04, 9.23e+04, 1.43e+05, 6.66e+04, 4.63e+04, 6.01e+04, 7.2e+03,
04067    4.41e+03, 1.28e+03, 2e+03, 2.2e+03, 938, 1.34e+03, 5.04e+03, 2.29e+03,
04068    1.29e+03, 2.25e+03, 405, 1.11e+03, 251, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04069    0, 0, 0, 0},
04070   {1.77e+03, 2.8e+03, 193, 32.6, 42.5, 92.2, 51.9, 902, 1.33e+03, 686, 137,
04071    89.5, 71.8, 26.1, 6.18, 717, 147, 3.07e+03, 2.76e+03, 7.44e+03, 1.42e+04,
04072    3.57e+04, 4.63e+04, 6.65e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04073    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.8e+03, 1.61e+03,
04074    1.57e+03, 1.83e+03, 1.73e+03, 675, 1.2e+03, 2.31e+03, 2.58e+03, 1.65e+03,
04075    2.41e+03, 5.26e+03, 1.47e+04, 2.07e+04, 1.24e+04, 0, 0, 0, 0, 0, 0, 0, 0,
04076    0, 0, 0, 1.04e+05, 3.89e+04, 0, 0, 0, 0, 0, 0.0185, 5.23e+03, 2.11e+03,
04077    1.27e+04, 6.64e+04, 2.64e+04, 3.96e+04, 4.96e+04, 2.43e+05, 7.55e+03, 318,
04078    1.34e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 343, 6.19e+04, 8.37e+04,
04079    5.63e+04, 6.4e+04, 5.84e+04, 6.68e+03, 2.92e+04, 2.58e+04, 394, 552, 543,
04080    4.38e+03, 1.99e+04, 1.12e+04, 71.9, 51.5, 3.15e+04, 6.83e+04, 3.83e+04,
04081    1.97e+04, 1.89e+04, 7.1e+03, 6.22e+03, 6.31e+03, 1.38e+03, 9.38e+03,
04082    5.51e+03, 1.57e+04, 1.86e+04, 2.18e+04, 3.46e+04, 1.73e+04, 1.3e+05,
04083    1.92e+05, 8.61e+04, 4.44e+04, 2.5e+04, 2.05e+03, 2.42e+03, 1.32e+03, 717,
04084    1.71e+03, 661, 1.46e+03, 5.78e+03, 3.15e+03, 708, 2.04e+03, 567, 1.58e+03,
04085    156, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04086   {1.83e+03, 2.73e+03, 184, 44.5, 53.2, 114, 41.7, 707, 797, 414, 133, 80.1,
04087    127, 21.8, 20.3, 226, 302, 3.14e+03, 2.94e+03, 7.98e+03, 1.27e+04,
04088    3.47e+04, 6.02e+04, 4.93e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04089    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.56e+03, 4.56e+03, 955,
04090    1.18e+03, 1.62e+03, 2.6e+03, 2.33e+03, 776, 1.1e+03, 1.54e+03, 2.6e+03,
04091    7.99e+03, 2.39e+04, 2.63e+04, 7.33e+03, 5.65, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04092    3.6e+04, 1.1e+05, 0, 0, 0, 0, 0, 0, 0.011, 1.2, 9.17e+03, 1.72e+04, 570,
04093    1.52e+04, 6.16e+04, 5.57e+03, 18.8, 0, 5.21e+04, 1.11e+04, 0, 0, 0, 0, 0,
04094    0, 0, 0, 0, 0, 0, 0, 0, 0, 6.75e+03, 9.58e+04, 6.86e+04, 4.41e+04, 1.31e+05,
04095    4.47e+04, 5.16e+04, 1.28e+04, 1.73e+03, 1.05e+03, 165, 8.18e+03, 1.83e+04,
04096    2.2e+04, 50.1, 38.1, 3.56e+04, 5.37e+04, 4.73e+04, 7.34e+04, 5.52e+04,
04097    1.44e+04, 5.27e+03, 1.02e+04, 3.85e+03, 2.41e+03, 1.27e+04, 2.01e+04,
04098    1.91e+04, 2.12e+04, 3.66e+04, 2.19e+04, 8.86e+04, 2.35e+05, 1.09e+05,
04099    2.14e+04, 1.17e+04, 1.06e+03, 1.84e+03, 1.71e+03, 1.08e+03, 866, 911,
04100    1.32e+03, 3.7e+03, 5.17e+03, 3.01e+03, 1.43e+03, 479, 1.03e+03, 25.9, 0, 0,
04101    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04102   {1.89e+03, 2.64e+03, 174, 61.2, 65.4, 99.6, 22, 684, 490, 438, 205, 104,
04103    201, 55.1, 28.8, 49.3, 538, 2.47e+03, 3.68e+03, 6.38e+03, 9.45e+03,
04104    3.08e+04, 6.55e+04, 2.91e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04105    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 481, 1.14e+04, 961,
04106    1.84e+03, 1.25e+03, 1.36e+03, 1.2e+03, 1.37e+03, 1.9e+03, 1.6e+03,
04107    2.91e+03, 3.56e+03, 1.41e+04, 1.05e+04, 6.37e+03, 267, 0, 0, 0, 0, 0, 0, 0,
04108    0, 0, 1.26e+04, 2.64e+04, 2.25e+04, 4.69, 1.14, 69.9, 0.16, 3.88e+05,
04109    1.99e+04, 299, 0, 4.51e+03, 0, 0, 158, 1.72e+04, 296, 0, 0, 3.76, 2.39e+04,
04110    1.69e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.28e+04, 9.45e+04,
04111    8.21e+04, 1.06e+05, 8.34e+04, 6.75e+04, 1.99e+04, 926, 206, 94, 4.29e+03,
04112    9.23e+03, 2.27e+03, 41.1, 73.6, 3.19e+04, 3.12e+04, 5.48e+04, 2.25e+04,
04113    5.1e+04, 1.87e+04, 5.76e+03, 6.76e+03, 4.32e+03, 2.69e+03, 5.35e+03,
04114    2.02e+04, 1.95e+04, 1.95e+04, 3.45e+04, 2.74e+04, 7.31e+04, 1.75e+05,
04115    6.17e+04, 1.41e+04, 1.13e+04, 2.8e+03, 1.17e+03, 2.47e+03, 1.14e+03, 582,
04116    984, 1.11e+03, 2.05e+03, 3.14e+03, 4.51e+03, 2.77e+03, 439, 1.43e+03, 3.99,
04117    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04118   {1.95e+03, 2.56e+03, 164, 81.9, 76.8, 121, 58.4, 727, 391, 223, 318, 257,
04119    250, 102, 59.6, 13.8, 666, 2e+03, 3.83e+03, 4.73e+03, 8.73e+03, 2.73e+04,
04120    5.54e+04, 6.42e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04121    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.05e+04, 2.09e+03, 1.34e+03,
04122    1.51e+03, 974, 1.05e+03, 1.54e+03, 1.06e+03, 876, 1.08e+03, 721, 5.03e+03,
04123    3.65e+03, 7.34e+03, 3.8e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.17e+04, 7.79e+04,
04124    1.75e+04, 0, 1.3e+03, 1.42e+05, 9e+04, 5.17e+05, 2.33e+05, 3.8e+05,
04125    4.51e+04, 7.65e+04, 0, 0, 0, 156, 0, 0, 0, 0, 332, 680, 1.75e+03, 534,
04126    3.51e+03, 5.7e+04, 5.34e+04, 3.87e+03, 671, 0, 0, 0, 0, 1.95e+03, 78.3,
04127    1.78, 7.2e+03, 7.64e+04, 9.42e+04, 5.84e+04, 4.09e+04, 1.13e+04, 186, 49.5,
04128    79, 36.7, 2.22e+03, 4.56e+03, 1.18e+03, 0, 2e+04, 2.7e+04, 2.59e+04,
04129    4.09e+03, 1.73e+04, 3.08e+04, 1.78e+04, 2.27e+04, 2.95e+04, 6.86e+03,
```

```
04130    8.56e+03, 1.68e+04, 1.64e+04, 2.34e+04, 2.28e+04, 2.15e+04, 9.09e+04,
04131    1.14e+05, 5.79e+04, 8.5e+03, 1.11e+04, 3.74e+03, 1.5e+03, 3.21e+03,
04132    1.36e+03, 538, 582, 2.16e+03, 4.32e+03, 1.58e+03, 4.99e+03, 3.03e+03, 448,
04133    1.66e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04134   {2.01e+03, 2.47e+03, 156, 108, 82, 222, 80.3, 523, 292, 118, 163, 378, 428,
04135    147, 146, 17.7, 625, 1.5e+03, 3.79e+03, 4.08e+03, 8.49e+03, 2.26e+04,
04136    5.72e+04, 8.91e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04137    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.29e+03, 2.72e+03, 2.04e+03,
04138    2.01e+03, 1.42e+03, 1.22e+03, 995, 585, 328, 1.74e+03, 1.13e+03, 1.89e+03,
04139    2.67e+03, 3.11e+03, 6.8e+03, 7.7, 0, 0, 0, 0, 0, 0, 0, 4.8e+03,
04140    1.28e+05, 1.74e+04, 90, 406, 0, 1.6e+05, 2.26e+05, 2.72e+05, 3.04e+05,
04141    25.7, 2.6e+05, 0, 0, 0, 0.00111, 1.57e+03, 0, 0, 0, 0, 1.78, 2.56e+03,
04142    1.78e+05, 2.1e+04, 5.2e+04, 1.95e+05, 3.55e+05, 1.68e+05, 6.85e+03, 0, 0,
04143    2.55e+04, 6.67e+05, 8.93e+05, 2.43e+05, 0, 630, 2.72e+04, 6.15e+04,
04144    3.12e+04, 14.4, 23, 22.4, 2.45, 0, 4.23e+03, 8.16e+03, 2.78e+03, 601, 51.3,
04145    8.64e+03, 1.77e+04, 3.75e+03, 3.23e+03, 2.41e+04, 2.18e+04, 4.68e+04,
04146    2.68e+04, 2.07e+04, 1.04e+04, 1.47e+04, 2.38e+04, 3.04e+04, 3.51e+04,
04147    1.56e+04, 7.42e+04, 8.64e+04, 6.75e+04, 1.07e+04, 1.18e+04, 4.49e+03,
04148    1.58e+03, 3.52e+03, 747, 634, 626, 2.5e+03, 4.3e+03, 1.96e+03, 2.55e+03,
04149    2.97e+03, 624, 1.52e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04150    0},
04151   {2.08e+03, 2.39e+03, 147, 140, 77.7, 378, 91.4, 186, 171, 110, 145, 79.2,
04152    1.08e+03, 195, 325, 23.8, 469, 931, 4.06e+03, 3.63e+03, 7.01e+03, 2.02e+04,
04153    5.26e+04, 1.55e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04154    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.27e+03, 1.81e+03, 987,
04155    1.14e+03, 834, 1.74e+03, 1.75e+03, 729, 983, 826, 2.14e+03, 1.76e+03,
04156    3.5e+03, 730, 3e+03, 2.14e+03, 1.39e+03, 1.37e+03, 0, 0, 0, 0, 0, 391, 196,
04157    8.17e+04, 590, 1.35, 0, 1.75e+04, 8.51e+04, 1.05e+05, 6.63e+04, 2.17e+05,
04158    276, 1.78e+05, 0, 0, 0, 0, 171, 2.51e+04, 1.25e+04, 0, 885, 7.77e+04,
04159    9.89e+03, 5.03e+03, 1.52e+04, 1.09e+04, 8.09e+04, 8.48e+04, 5.74e+04, 910,
04160    820, 0, 0, 0, 1.25e+05, 8.93e+04, 0, 0, 4.64, 3.7e+03, 3.7e+03, 4.22, 11.3,
04161    0, 0, 0, 0, 4.13e+03, 106, 885, 1.15e+04, 602, 1.44e+04, 2.6e+03, 297,
04162    1.51e+03, 9.1e+03, 2.25e+04, 3.5e+04, 1.95e+04, 1.63e+04, 1.17e+04,
04163    2.05e+04, 1.57e+04, 5.2e+04, 2.28e+04, 3.48e+04, 3.87e+04, 4.52e+04,
04164    1.04e+04, 5.52e+03, 5.11e+03, 822, 4.09e+03, 823, 716, 460, 1.38e+03,
04165    2.73e+03, 2.45e+03, 4.45e+03, 4.29e+03, 1.56e+03, 4.61e+03, 0, 0, 0, 0, 0,
04166    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04167   {2.14e+03, 2.31e+03, 139, 179, 63.4, 550, 67.9, 67.8, 111, 102, 171, 64.7,
04168    1.32e+03, 154, 632, 19.2, 299, 776, 3.76e+03, 3.38e+03, 7.36e+03, 1.69e+04,
04169    4.74e+04, 1.74e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04170    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.13e+03, 2.21e+03, 2.51e+03,
04171    1.98e+03, 1.65e+03, 1.24e+03, 1.37e+03, 765, 396, 641, 1.72e+03, 1.71e+03,
04172    705, 675, 2.49e+03, 1.45e+03, 2.24e+03, 3.01e+03, 3.69, 0, 0, 0, 0,
04173    1.25e+03, 0.0299, 1.15e+04, 1.64e+04, 0, 2.05e+03, 1.34e+04, 9.25e+03, 0,
04174    818, 7.38e+04, 1.18e+04, 7.99e+04, 0, 0, 0, 0, 0, 38.9, 4.47e+04, 3.41e+04,
04175    6.54e+04, 4.1e+04, 9.13e+03, 2.92e+03, 1.98e+04, 0, 0, 0.0322, 0, 0, 0, 0,
04176    0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00222, 0, 0, 0, 0, 0, 38.2, 0, 74.4, 2.9e+04,
04177    2.74e+04, 791, 2.02e+03, 182, 72.4, 1.86e+03, 4.37e+03, 1.45e+04, 2.19e+04,
04178    1.69e+04, 2.46e+04, 1.9e+04, 1.78e+04, 1.99e+04, 3.29e+04, 3.88e+04,
04179    2.06e+04, 1.6e+04, 2.59e+04, 6.21e+03, 6.5e+03, 1.8e+03, 3.19e+03, 974,
04180    1.13e+03, 1.22e+03, 1.89e+03, 1.72e+03, 2.53e+03, 3.17e+03, 1.4e+04,
04181    4.91e+03, 1.11e+04, 0.568, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04182   {2.2e+03, 2.22e+03, 132, 220, 42.3, 698, 41.3, 109, 338, 270, 135, 116, 943,
04183    131, 1.01e+03, 20, 198, 879, 2.41e+03, 4.7e+03, 8.78e+03, 1.44e+04,
04184    5.82e+04, 1.54e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04185    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 555, 4.22e+03, 524, 1.83e+03,
04186    3.05e+03, 1.37e+03, 1.33e+03, 1.48e+03, 671, 358, 502, 1.12e+03, 586, 546,
04187    2.55e+03, 808, 945, 2.36e+03, 2.22e+03, 2.66, 0, 0, 0, 0, 1.11e+05,
04188    3.87e+04, 7.41e+03, 0, 0.00222, 23.8, 0, 0, 5.22e+03, 3.21, 3.32e+04,
04189    1.38e+05, 0, 0, 0, 0, 6.2e+04, 7.52e+04, 1.95e+05, 1.84e+03, 2.02e+04,
04190    7.2e+03, 7.41e+03, 1.93e+04, 1.22e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 523, 0,
04191    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.94e+03, 3.02e+04, 2.12e+04,
04192    2.8e+03, 905, 306, 43.6, 66.8, 2.27e+03, 6.74e+03, 9e+03, 2.18e+04,
04193    2.49e+04, 2.24e+04, 1.47e+04, 1.65e+04, 4.93e+04, 6.82e+04, 2.16e+04,
04194    6.4e+03, 8.54e+03, 5.02e+03, 2.55e+03, 2.34e+03, 802, 1.97e+03, 721,
04195    1.55e+03, 1.36e+03, 922, 1.92e+03, 6.34e+03, 8.54e+03, 5.11e+03, 11.1, 0,
04196    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04197   {2.26e+03, 2.13e+03, 127, 264, 25, 771, 78.1, 175, 398, 893, 85.3, 135, 670,
04198    234, 1.16e+03, 29.3, 108, 930, 1.64e+03, 5.92e+03, 6.52e+03, 1.48e+04,
04199    4.5e+04, 2.05e+05, 6.58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04200    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.06e+03, 249, 297, 1.65e+03,
04201    2.8e+03, 1.32e+03, 394, 449, 435, 264, 652, 461, 610, 2.65e+03, 2.63e+03,
04202    2.9e+03, 1.71e+04, 1.92e+04, 9.24e+03, 10.7, 0, 0, 0, 2.31e+04, 1.75e+05,
04203    5.67e+04, 0, 0, 0, 0, 0, 3.71e+04, 0, 0, 6.36e+04, 4.04e+03, 80.9, 670,
04204    28.3, 1.52e+05, 1.51e+05, 1.95e+05, 1.44e+05, 1.11e+04, 3.04e+04, 1.94e+04,
04205    151, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 25.3, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04206    1.21, 0, 0, 4.03e+03, 3.12e+03, 3.32e+04, 3.99e+04, 1.93e+04, 5.56e+03,
04207    1.27e+03, 1.55e+03, 226, 225, 2.06e+03, 2.38e+03, 2.79e+03, 9.31e+03,
04208    1.82e+04, 7.97e+03, 3.57e+04, 2.96e+04, 2.46e+04, 1.27e+04, 3.29e+04,
04209    1.37e+04, 3.06e+03, 7.8e+03, 1.53e+03, 1.81e+03, 941, 1.23e+03, 2.32e+03,
04210    5.96e+03, 6.52e+03, 5.77e+03, 2.52e+03, 8.61e+03, 9.1e+03, 593, 3.11, 0, 0,
04211    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04212   {2.33e+03, 2.05e+03, 121, 307, 21.9, 767, 120, 258, 467, 1.48e+03, 73.1,
04213    130, 560, 616, 888, 64.1, 40, 659, 1.62e+03, 7.15e+03, 5.06e+03, 1.76e+04,
04214    3.89e+04, 1e+05, 513, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04215    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11.6, 2.23e+03, 338, 139,
04216    2.77e+03, 1.22e+03, 1.28e+03, 850, 791, 961, 403, 897, 968, 800, 2.56e+03,
```

```
04217    1.12e+04, 1.01e+04, 8.23e+03, 1.39e+04, 2.37e+03, 0, 0, 0, 0, 1.8e+05,
04218    3.16e+04, 0, 0, 0, 1.95e+04, 1.7e+05, 4.54e+03, 0, 0, 0, 0, 70.5, 1.52e+03,
04219    0, 5.56e+04, 1.84e+05, 6.27e+04, 7.79e+04, 4.63e+03, 308, 0, 0, 0, 0, 0, 0,
04220    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.09e+04, 97.9, 243, 0, 61.9,
04221    8.21e+03, 2.24e+04, 7.76e+04, 7.32e+04, 3.24e+04, 6.6e+03, 4.26e+03, 645,
04222    689, 1.41e+03, 1.57e+03, 4.86e+03, 6.66e+03, 3.4e+03, 5.39e+03, 2.01e+03,
04223    4.66e+04, 1.14e+04, 1.9e+04, 4.35e+04, 1.85e+04, 3.28e+04, 5.55e+03,
04224    8.44e+03, 1.55e+03, 2.01e+03, 1.44e+03, 2.48e+04, 3.58e+04, 4.83e+04,
04225    2.84e+04, 1.43e+04, 7.27e+03, 9.23e+03, 1.77e+04, 29.1, 0.728, 0, 0, 0, 0,
04226    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04227   {2.4e+03, 1.96e+03, 121, 340, 37, 774, 158, 295, 894, 2.99e+03, 47.4, 134,
04228    554, 1.23e+03, 496, 145, 24.6, 456, 1.77e+03, 8.15e+03, 4.82e+03, 1.89e+04,
04229    4.94e+04, 1.64e+05, 0.204, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04230    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.63e+03, 402, 111,
04231    3.48e+03, 1.92e+03, 1.81e+03, 1.32e+03, 1.64e+03, 1.53e+03, 1.48e+03,
04232    1.01e+03, 1.48e+03, 683, 2.54e+03, 7.19e+03, 5.17e+03, 5.44e+03, 6.85e+03,
04233    5.74e+03, 0, 0, 0, 0, 2.96e+04, 2.18e+04, 0, 0, 0, 5.75e+04, 6.55e+04, 574,
04234    2.03e+03, 3.53e+03, 784, 0, 0, 569, 0.594, 0, 386, 448, 0, 0, 0, 0, 0, 0,
04235    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 31, 0, 0, 0, 0, 0, 0, 0, 1.8e+04, 897,
04236    1.66e+04, 2.42e+04, 7.52e+03, 2.74e+04, 6.93e+04, 2.38e+05, 1.32e+05,
04237    2.84e+04, 2.27e+04, 2.21e+04, 2.71e+03, 4.26e+03, 711, 4.07e+03, 5.24e+03,
04238    1.55e+04, 7.08e+03, 1.84e+03, 1.13e+03, 3.87e+04, 7.11e+03, 3.26e+04,
04239    6.16e+04, 5.18e+04, 1.16e+04, 5.77e+03, 1.19e+04, 1.28e+03, 1.64e+03,
04240    1.56e+04, 4.06e+04, 7.23e+04, 1.13e+05, 1.04e+05, 4.44e+04, 2.78e+04,
04241    9.46e+03, 1.1e+04, 8.55, 0.103, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04242    0, 0},
04243   {2.47e+03, 1.88e+03, 121, 358, 81.9, 793, 207, 277, 1.05e+03, 4.35e+03,
04244    52.1, 242, 388, 1.45e+03, 203, 579, 85.2, 526, 1.56e+03, 9.94e+03,
04245    4.72e+03, 2.2e+04, 4.45e+04, 1.73e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04246    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.32e+03,
04247    370, 156, 1.34e+03, 2.19e+03, 2.2e+03, 1.82e+03, 3.61e+03, 1.7e+03,
04248    1.32e+03, 588, 1.42e+03, 1.46e+03, 1.39e+03, 6.24e+03, 1.56e+04, 1.99e+04,
04249    9.94e+03, 1.33e+03, 0, 0, 0, 0, 0, 307, 0, 0, 0, 2.45e+03, 2.94e+04,
04250    2.55e+04, 2.58e+04, 6.28e+03, 5.08e+04, 2.75e+04, 759, 0, 0, 0, 0, 0, 0, 0,
04251    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 57.4, 920, 0, 0, 0, 0, 0, 0,
04252    0, 5.46e+04, 5.4e+04, 4.06e+04, 6.31e+04, 9.19e+04, 1.43e+05, 1.23e+05,
04253    9.76e+04, 1.68e+04, 7.95e+04, 5.02e+04, 1.49e+04, 1.56e+04, 5.25e+03,
04254    1.18e+04, 1.21e+03, 2.14e+03, 1.62e+03, 1.91e+03, 2.79e+04, 1.79e+04,
04255    3.63e+04, 8.64e+04, 4.24e+04, 1.03e+04, 4.87e+03, 1.22e+04, 1.24e+03,
04256    2.19e+03, 6.63e+04, 6.41e+04, 9.2e+04, 8.7e+04, 1.18e+05, 9.45e+04,
04257    3.61e+04, 1.86e+04, 1.16e+04, 13.6, 0.0341, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04258    0, 0, 0, 0, 0, 0},
04259   {2.55e+03, 1.79e+03, 123, 367, 145, 869, 257, 332, 901, 4.16e+03, 158, 428,
04260    236, 1.11e+03, 498, 506, 355, 782, 1.16e+03, 1.05e+04, 5.23e+03, 1.98e+04,
04261    7.01e+04, 1.2e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04262    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.36e+03, 373, 186, 2.15e+03,
04263    2.11e+03, 3.02e+03, 5.4e+03, 6.52e+03, 1.18e+03, 1.25e+03, 694, 793, 928,
04264    1.72e+03, 4.3e+03, 7.12e+03, 7.13e+03, 2.34e+03, 14.5, 0, 0, 0, 0, 0, 88.4,
04265    912, 0, 0, 95.8, 1.23e+05, 0.00111, 0, 1.37e+03, 1.03e+04, 3.51e+03, 732,
04266    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 34.6,
04267    1.26e+03, 0, 0, 0, 0, 343, 457, 9.36e+03, 3.4e+04, 8.84e+04, 9.01e+04, 0,
04268    5.71e+03, 3.5e+05, 1.25e+05, 5.31e+04, 4.19e+04, 4.9e+04, 5.25e+04,
04269    3.52e+04, 1.6e+04, 1.56e+04, 1.86e+03, 581, 1.2e+03, 4.65e+03, 1.18e+04,
04270    2.57e+04, 5.92e+04, 7.99e+04, 1.39e+05, 7.93e+03, 3.98e+03, 9.82e+03,
04271    2.63e+03, 2.22e+03, 9.1e+04, 4.96e+04, 8.07e+04, 5.29e+04, 6.77e+04,
04272    5.59e+04, 4.37e+04, 2.33e+04, 2.56e+03, 1.56, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04273    0, 0, 0, 0, 0, 0, 0},
04274   {2.63e+03, 1.7e+03, 128, 382, 211, 1.05e+03, 226, 568, 610, 2.99e+03, 250,
04275    289, 442, 811, 598, 431, 325, 446, 1.01e+03, 8.31e+03, 6.15e+03, 1.9e+04,
04276    4.97e+04, 1.6e+05, 1.48e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04277    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 438, 496, 450,
04278    2.54e+03, 1.73e+03, 3.64e+03, 5.71e+03, 6.64e+03, 2.53e+03, 3.05e+03,
04279    2.56e+03, 1.34e+03, 1.19e+03, 1.15e+03, 1.58e+03, 5.12e+03, 9.49e+03,
04280    8.28e+03, 6e+03, 81.7, 76.1, 20.1, 0, 0, 4.95e+03, 0, 0, 0, 601, 6.5e+04,
04281    4.04e+03, 0.269, 329, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04282    0, 0, 0, 0, 0, 0, 0, 0, 0, 152, 0, 8.4e+03, 1.48e+03, 7.92e+03, 2.17e+04,
04283    2.39e+03, 0, 16.6, 0, 0, 0, 0, 1.75e+04, 8.79e+04, 5.68e+04, 2.02e+04,
04284    1.64e+04, 2.91e+04, 2.44e+04, 2.83e+04, 1.62e+04, 6.27e+03, 1.37e+03, 598,
04285    2.7e+03, 4.44e+04, 6.21e+04, 1.13e+05, 6.5e+04, 1.1e+05, 7.16e+03,
04286    3.94e+03, 4.36e+03, 1.97e+03, 1.06e+03, 9.71e+04, 4.83e+04, 4.35e+04,
04287    1.96e+04, 4.77e+04, 3.01e+04, 3.42e+04, 2.93e+04, 0.0816, 0.00222, 0, 0, 0,
04288    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04289   {2.72e+03, 1.62e+03, 138, 411, 283, 1.31e+03, 171, 998, 403, 1.19e+03, 426,
04290    207, 400, 578, 755, 258, 360, 300, 913, 6.21e+03, 5.52e+03, 2.11e+04,
04291    3.24e+04, 2.03e+04, 8.2e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04292    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 116, 509, 2.33e+03,
04293    2.12e+03, 1.56e+03, 2.34e+03, 7.22e+03, 1.53e+03, 3.92e+03, 3.65e+03,
04294    3.69e+03, 1.1e+03, 1.6e+03, 1.02e+03, 1.41e+03, 2.48e+03, 5.27e+03,
04295    5.74e+03, 5.67e+03, 2.58e+03, 540, 130, 0, 0, 235, 21.1, 0, 0, 2.2e+03,
04296    1.82e+03, 0, 5.4e+03, 1.21e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04297    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.79e+04, 9.46e+03, 8.42e+04,
04298    1.13e+05, 1.97e+04, 0, 0, 0, 281, 0, 0, 0, 145, 8.57e+04, 3.52e+04,
04299    2.82e+04, 1.04e+04, 9.95e+03, 2.63e+04, 2.15e+04, 1.08e+04, 3.96e+04,
04300    2.33e+04, 2.02e+04, 3.96e+04, 6.3e+04, 1.19e+05, 6.98e+04, 5.85e+04,
04301    1.45e+04, 3.5e+03, 2.9e+03, 1.05e+03, 1.29e+03, 1.15e+05, 6.55e+04,
04302    3.93e+04, 1.77e+04, 1.44e+04, 2.04e+04, 1.25e+04, 6.27e+03, 0, 0, 0, 0, 0,
04303    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

```
04304   {2.83e+03, 1.54e+03, 151, 434, 371, 1.43e+03, 179, 1.22e+03, 319, 466, 526,
04305    313, 535, 944, 702, 179, 359, 250, 893, 4.81e+03, 3.94e+03, 1.84e+04,
04306    4.48e+04, 3.49e+04, 5.47e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04307    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.2e+03, 1.47e+03,
04308    1.14e+03, 849, 2.72e+03, 3.45e+03, 1.33e+04, 1.39e+03, 8.43e+03, 7.1e+03,
04309    3.22e+03, 1.66e+03, 1.25e+03, 734, 374, 932, 1.27e+03, 1.61e+03, 4.35e+03,
04310    2.1e+03, 34.6, 86.8, 0, 0, 244, 26.6, 0, 0, 1.52, 3.88e+04, 681, 1.2e+05,
04311    4.7e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04312    0, 0, 0, 0, 0, 0, 0, 0, 0, 749, 2.07e+04, 4.28e+04, 6.5e+04, 0, 0, 0, 0, 0,
04313    0, 0, 4.33e+03, 8.09e+03, 8.39e+03, 1.18e+04, 853, 4.35e+03, 6.98e+04,
04314    2.44e+04, 3.79e+04, 4.64e+04, 2.07e+04, 5.63e+04, 8.28e+04, 5.97e+04,
04315    6.43e+04, 4.21e+04, 2.13e+04, 3.18e+03, 4.31e+03, 1.19e+03, 1.52e+03,
04316    1.64e+05, 6.9e+04, 2.12e+04, 1.67e+04, 1.64e+04, 3.34e+04, 1.21e+03, 642,
04317    57.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04318   {2.92e+03, 1.47e+03, 167, 463, 460, 1.31e+03, 230, 1.6e+03, 196, 150, 501,
04319    423, 494, 635, 491, 231, 218, 265, 687, 2.92e+03, 2.27e+03, 1.06e+04,
04320    1.78e+04, 4.81e+04, 1.29e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04321    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.88e+03, 1.5e+03,
04322    419, 875, 1.55e+03, 3.84e+03, 2.54e+03, 1.82e+03, 1.53e+04, 4.16e+03,
04323    6.25e+03, 1.88e+03, 1.11e+03, 552, 238, 302, 737, 683, 4.11e+03, 1.96e+04,
04324    2.59e+03, 17.1, 0, 0, 0, 0, 434, 736, 8.3e+04, 4.58e+04, 8.79e+03,
04325    2.64e+05, 2.13e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04326    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.23e+04, 1.24e+05, 4.63e+04,
04327    989, 0, 0, 0, 0, 0, 0, 6.41e+04, 4.77e+04, 7.49e+03, 3.98e+03, 7.64e+03,
04328    4.33e+03, 4.76e+04, 2.56e+04, 4.67e+04, 7.03e+04, 4.62e+04, 4.32e+04,
04329    1.08e+05, 7.22e+04, 6.8e+04, 3.59e+04, 1.76e+04, 3.51e+03, 4.11e+03,
04330    1.47e+03, 2.09e+03, 2.46e+05, 6.91e+04, 3.12e+04, 1.49e+04, 1.46e+04,
04331    1.23e+04, 1.43e+04, 613, 210, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04332    0, 0, 0},
04333   {3.01e+03, 1.4e+03, 189, 502, 515, 1.2e+03, 326, 1.95e+03, 87.9, 102, 193,
04334    545, 254, 668, 755, 187, 325, 356, 1.14e+03, 3.16e+03, 1.09e+03, 5e+03,
04335    2.25e+04, 1.27e+05, 5.51e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04336    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 246, 2.41e+03, 430, 367,
04337    948, 1.07e+03, 1.91e+03, 3.77e+03, 2.02e+03, 1.13e+04, 6.38e+03, 1.31e+03,
04338    2.5e+03, 2.04e+03, 1.59e+03, 585, 335, 1.61e+03, 1.7e+03, 5.28e+03,
04339    8.48e+03, 3.04e+03, 0, 0, 0, 0, 9.39, 466, 0, 4.58e+04, 9.97e+04, 6.42e+04,
04340    6.6e+05, 0, 0, 0, 0, 0, 0, 0, 0.00444, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04341    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.97e+03, 1.33e+05, 6.75e+04,
04342    3.44e+03, 0, 0, 0, 0, 0, 0, 8.86e+04, 6.17e+04, 2.56e+04, 5.64e+03,
04343    1.21e+04, 98.8, 2.23e+04, 1.24e+05, 1.33e+05, 6.8e+04, 6.38e+04, 1.83e+05,
04344    9.71e+04, 6.95e+04, 1.84e+04, 2.97e+04, 1.74e+04, 3.6e+03, 2.25e+03,
04345    2.91e+03, 2.56e+03, 1.69e+05, 5.35e+04, 1.3e+04, 1.95e+04, 1.52e+04,
04346    1.25e+04, 4.1e+04, 3.91e+03, 0.04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04347    0, 0, 0, 0, 0, 0},
04348   {3.13e+03, 1.35e+03, 217, 545, 517, 1.23e+03, 369, 1.91e+03, 136, 174, 130,
04349    501, 283, 843, 621, 307, 256, 780, 880, 3.05e+03, 1.09e+03, 3.08e+03,
04350    2.64e+04, 8.89e+04, 3.06e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04351    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 67.3, 940, 3.08e+03, 3.45e+03,
04352    391, 479, 1.23e+03, 2.45e+03, 1.88e+03, 1.59e+03, 3.13e+03, 1.19e+04,
04353    1.11e+04, 2.27e+03, 3.03e+03, 3.03e+03, 305, 348, 1.01e+03, 3.4e+03,
04354    1.11e+03, 2.57e+03, 3.32e+03, 410, 0, 0, 0, 0, 0.0165, 241, 16.5, 1.49e+05,
04355    7.39e+04, 139, 531, 0, 0, 0, 0, 0, 0, 0, 0, 0, 68.2, 105, 0, 0, 0, 0, 0, 0, 0, 0,
04356    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.91e+04, 4.44e+04,
04357    9.61e+03, 0, 0, 0, 0, 0, 0, 1.86e+03, 6.12e+04, 3.69e+04, 3.78e+04,
04358    1.8e+04, 7.54e+03, 5.63e+03, 1.6e+05, 1.54e+05, 1.36e+05, 1.25e+05,
04359    1.94e+05, 9.43e+04, 5.69e+04, 2.24e+04, 2.3e+04, 2.03e+04, 7.46e+03,
04360    2.29e+03, 2.82e+03, 3.94e+03, 8e+04, 4.51e+04, 1.85e+04, 8.27e+04,
04361    2.84e+04, 1.68e+04, 2.16e+04, 4.13e+03, 8.53, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04362    0, 0, 0, 0, 0, 0, 0, 0, 0},
04363   {3.24e+03, 1.31e+03, 249, 605, 505, 1.31e+03, 342, 1.58e+03, 267, 113, 138,
04364    570, 322, 681, 306, 403, 534, 3.04e+03, 1.73e+03, 1.14e+03, 3.21e+03,
04365    1.17e+03, 3.47e+04, 9.57e+04, 1.95e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04366    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.13e+03, 4.55e+03,
04367    5.22e+03, 2.85e+03, 285, 1.35e+03, 1.14e+03, 2.25e+03, 1.99e+03, 591, 688,
04368    1.97e+03, 3.89e+03, 1.99e+03, 1.61e+03, 285, 89.8, 288, 4.57e+03, 3.1e+03,
04369    273, 632, 1.54e+03, 406, 0, 0, 0, 0, 0, 0, 20.8, 7.55e+05, 3.64e+05,
04370    2.14e+04, 5.01e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04371    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.21e+05, 4.41e+04,
04372    6.12e+04, 0, 0, 0, 0, 0, 0, 0, 7.49e+03, 7.44e+04, 4.52e+04, 3.87e+04,
04373    2.91e+04, 342, 8.86e+04, 6.57e+04, 3.17e+04, 4.53e+04, 3.12e+04, 6.94e+04,
04374    7.91e+04, 2.48e+04, 1.68e+04, 1.71e+04, 1.1e+04, 1.99e+04, 1.74e+04,
04375    1.23e+04, 4.16e+04, 1.91e+04, 3.8e+04, 6.5e+04, 5.41e+04, 1.62e+04,
04376    1.41e+04, 623, 18.7, 0, 407, 123, 473, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04377    0, 0},
04378   {3.33e+03, 1.28e+03, 287, 676, 506, 1.44e+03, 471, 1.23e+03, 297, 159, 234,
04379    632, 282, 583, 279, 169, 293, 3.8e+03, 1.04e+03, 170, 4.13e+03, 601,
04380    3.76e+04, 9.67e+04, 1.8e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04381    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.85e+03, 218, 6.19e+03,
04382    7.29e+03, 1.94e+03, 1.14e+03, 1.12e+03, 498, 325, 157, 249, 627, 825, 935,
04383    1e+03, 617, 586, 852, 732, 5.25e+03, 663, 0, 445, 97.3, 300, 0.36, 0, 0, 0,
04384    0, 0, 2.66e+03, 1.16e+06, 2.38e+05, 1.99e+04, 3.37, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04385    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04386    0, 2.26e+03, 3.45e+04, 1.71e+05, 4.39e+04, 66, 0, 0, 0, 0, 0, 0, 2.76e+04,
04387    1.1e+05, 7.56e+04, 6.12e+04, 6.4e+04, 6.35e+03, 1.25e+04, 2.08e+04,
04388    3.18e+04, 2.49e+04, 4.13e+03, 1.95e+05, 5.24e+04, 2.14e+04, 3.73e+04,
04389    5.47e+04, 4.9e+04, 6.28e+04, 5.07e+04, 3.47e+04, 1.68e+04, 3.44e+04,
04390    2.85e+04, 5.77e+04, 2.34e+04, 4.8e+04, 141, 11.1, 0, 0, 227, 127, 0, 0, 0,
```

```
04391    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04392   {3.45e+03, 1.28e+03, 327, 756, 529, 1.63e+03, 701, 1.19e+03, 794, 222, 556,
04393    697, 240, 591, 182, 147, 303, 1.37e+03, 909, 361, 3.76e+03, 2.52e+03,
04394    2.23e+04, 1.69e+05, 3.35e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04395    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.81e+03, 1.6e+03, 4.99e+03,
04396    3.34e+04, 2.41e+04, 7.94e+03, 2.54e+03, 311, 161, 90.8, 106, 211, 517,
04397    1.25e+03, 584, 215, 454, 1.36e+03, 5.53e+03, 5.99e+03, 33.8, 0, 0, 0, 0, 0,
04398    0, 0, 0.391, 0.0187, 0.18, 1.13e+04, 9.05e+05, 3.4e+05, 2.73e+04, 0, 0, 0,
04399    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04400    0, 0, 0, 0, 0, 0, 0, 0, 2.67e+05, 2.81e+05, 3.96e+05, 5.83e+03, 0, 0, 0, 0,
04401    0, 0, 0, 4.44e+04, 1.28e+05, 7.78e+04, 7.36e+04, 5.19e+04, 5.39e+04,
04402    3.78e+04, 3.61e+04, 6.02e+03, 8e+03, 2.66e+04, 1.44e+05, 2.63e+04,
04403    2.36e+04, 4.53e+04, 8.97e+04, 8.18e+04, 8.09e+04, 4.74e+04, 1.8e+04,
04404    5.22e+04, 8.5e+04, 8.52e+04, 6.55e+04, 6.45e+04, 495, 11.1, 0, 0, 2.46e+03,
04405    2.23e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04406   {3.57e+03, 1.3e+03, 368, 845, 582, 1.85e+03, 863, 1.09e+03, 1.1e+03, 143,
04407    581, 1.34e+03, 390, 575, 102, 125, 174, 1e+03, 1.02e+03, 392, 4.04e+03,
04408    1.19e+04, 1.32e+04, 2.12e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04409    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 231, 2.35e+04, 2.13e+04,
04410    1.6e+04, 2.51e+04, 5.06e+04, 3.36e+04, 750, 375, 402, 246, 189, 783, 507,
04411    3.37e+03, 4.52e+03, 2.46e+03, 5.56e+03, 588, 154, 0.981, 0, 0, 0, 0, 0, 0,
04412    0, 0.328, 10.5, 18.5, 1.5e+05, 6.93e+05, 2.13e+05, 3.74e+04, 0, 0, 0, 0, 0,
04413    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04414    0, 0, 0, 0, 0, 0, 1.44e+05, 2.54e+05, 2.55e+05, 2.65e+04, 0, 0, 0, 289, 0,
04415    0, 0, 0, 1.59e+04, 1.27e+05, 4.44e+04, 4.29e+04, 9.19e+04, 3.84e+04,
04416    2.55e+04, 2.26e+04, 0, 0, 8.34e+04, 3.59e+04, 1.33e+04, 4.06e+04, 6.28e+04,
04417    7.86e+04, 1.08e+05, 5.26e+04, 1.74e+04, 9.54e+04, 1.18e+05, 2.24e+05,
04418    4.55e+04, 5.22e+04, 233, 50.6, 0, 0, 3.02e+03, 2.72e+03, 0, 0, 0, 0, 0, 0,
04419    0, 0, 0, 0, 0, 0, 0, 0},
04420   {3.66e+03, 1.35e+03, 412, 936, 671, 2.03e+03, 859, 855, 1.13e+03, 472,
04421    1.04e+03, 1.91e+03, 603, 269, 273, 49.9, 209, 963, 1.76e+03, 1.09e+03, 924,
04422    1.73e+04, 2.21e+04, 2.52e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04423    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 125, 287, 785, 620, 390, 1.09e+04,
04424    1.15e+04, 492, 2.19e+03, 128, 72.2, 95, 251, 264, 307, 922, 1.98e+03,
04425    6.31e+03, 7.1e+03, 1.46e+03, 318, 0.373, 1.8, 0, 0, 0, 0, 0, 0, 0.00552,
04426    47.9, 80.4, 1.19e+03, 1.36e+06, 6.55e+05, 1.71e+04, 8.62e+03, 0, 0, 0, 0,
04427    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04428    0, 0, 0, 0, 0, 247, 39.5, 1.92e+03, 2.21e+03, 1.24e+05, 1.16e+05, 7.9e+04,
04429    3.22e+04, 2.33e+04, 3.02e+04, 3.26e+04, 0, 0, 0, 0, 2.21e+04, 6.28e+04,
04430    4.63e+04, 5.02e+04, 1.17e+04, 2.65e+04, 2.66e+04, 0, 0, 67.7, 9.13e+04,
04431    2.55e+04, 1.53e+04, 5.47e+04, 8.06e+04, 6.09e+04, 3.61e+04, 5.17e+04,
04432    8.6e+04, 1.85e+05, 1.36e+05, 3.7e+04, 5.92e+04, 340, 387, 100, 20.2, 322,
04433    941, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04434   {3.76e+03, 1.43e+03, 454, 1.02e+03, 794, 2.16e+03, 852, 716, 1.13e+03, 512,
04435    609, 2e+03, 659, 87.2, 416, 78, 339, 1.27e+03, 2.78e+03, 680, 2.29e+03,
04436    2.14e+04, 2.23e+04, 2.38e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04437    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.8e+03, 1.91e+03, 546, 574, 254,
04438    1.24e+03, 4.6e+03, 2.77e+03, 1.06e+03, 1e+03, 1.11e+03, 1.44e+03, 1.27e+03,
04439    547, 366, 964, 3.22e+03, 7.9e+03, 453, 264, 105, 47.7, 0.979, 0, 0, 0, 0,
04440    0, 0, 190, 587, 885, 7.37e+03, 6.81e+05, 1.24e+05, 4.47e+04, 3.72e+03, 0,
04441    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04442    23.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 52.7, 5.18e+04, 4.75e+04, 6.47e+04,
04443    7.94e+04, 4.29e+04, 1.6e+04, 7.65e+04, 8.66e+03, 3.29e+03, 0, 0, 0, 371,
04444    4.16e+03, 2.39e+04, 2.73e+04, 768, 0, 0, 0, 8.93e+03, 3.11e+04, 3.25e+04,
04445    6.37e+04, 1.89e+05, 1.61e+05, 3.09e+04, 9.9e+04, 1.23e+05, 9.29e+04,
04446    5.84e+04, 2.35e+04, 3.93e+04, 419, 2.65e+03, 550, 107, 19.6, 21.1, 0, 0, 0,
04447    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04448   {3.85e+03, 1.54e+03, 496, 1.1e+03, 937, 2.26e+03, 1.11e+03, 630, 1.32e+03,
04449    344, 378, 1.76e+03, 585, 107, 315, 104, 275, 3.37e+03, 1.56e+03, 349,
04450    3.61e+03, 3.75e+04, 8.68e+03, 1.79e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04451    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.87e+03, 1.02e+04, 591,
04452    272, 189, 166, 4.83e+03, 2.87e+03, 1.55e+03, 1.63e+03, 1.72e+03, 950, 625,
04453    1.25e+03, 532, 1.14e+03, 1.25e+03, 539, 696, 1.31e+03, 1.63e+03, 431, 214,
04454    504, 1.39e+03, 806, 477, 652, 18.4, 505, 939, 4.55e+03, 3.82e+05, 5.55e+05,
04455    1.26e+04, 7.45e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04456    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10.9,
04457    3.62e+04, 7.58e+04, 27.7, 364, 5.82e+04, 3.27e+04, 2.13e+04, 3.69e+03,
04458    2.27e+04, 2.07e+04, 4.52e+04, 5.71e+04, 2.55e+04, 3.44e+03, 1.71e+03, 128,
04459    0, 0, 0, 0, 3.51e+04, 5.02e+04, 8.91e+04, 8.54e+04, 4.44e+04, 4.38e+04,
04460    1.19e+05, 1.54e+05, 2.78e+04, 7.39e+04, 1.56e+04, 2.81e+04, 340, 138, 597,
04461    348, 5.13, 32.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04462   {3.93e+03, 1.69e+03, 535, 1.18e+03, 1.07e+03, 2.26e+03, 1.5e+03, 642,
04463    1.74e+03, 223, 267, 1.61e+03, 494, 184, 170, 85.8, 609, 4.24e+03, 911,
04464    1.07e+03, 6.8e+03, 4.72e+04, 4.53e+04, 1.59e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04465    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.33e+03, 6.34e+03,
04466    1.39e+04, 760, 80, 231, 168, 1.22e+03, 2.74e+03, 899, 191, 1.25e+03, 806,
04467    980, 1.17e+03, 950, 722, 663, 1.13e+03, 8.28e+03, 9.79e+03, 2.75e+03,
04468    1.25e+03, 1.28e+03, 2.39e+03, 1.09e+04, 7.45e+03, 2.25e+03, 630, 0,
04469    3.84e+03, 3.53e+03, 2.45e+05, 7.07e+05, 7.73e+05, 1.99e+04, 3.55e+04, 0, 0,
04470    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04471    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.59e+04, 1.4e+05,
04472    1.4e+05, 1.41e+04, 693, 1.83e+04, 2.71e+03, 4.92e+04, 6.11e+04, 6.64e+04,
04473    2.91e+03, 1.38e+03, 2.73e+03, 0, 0, 0, 0, 2.29e+03, 6.49e+04, 1.09e+05,
04474    1.05e+05, 2.6e+04, 7.43e+04, 1.24e+05, 1.75e+05, 7.71e+04, 9.47e+03,
04475    7.26e+03, 1.2e+04, 137, 87.1, 510, 102, 124, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04476    0, 0, 0, 0, 0},
04477   {4.01e+03, 1.87e+03, 568, 1.25e+03, 1.15e+03, 2.22e+03, 1.68e+03, 790,
```

```
04478    2.87e+03, 228, 222, 1.73e+03, 470, 257, 126, 133, 788, 3.72e+03, 1.43e+03,
04479    1.82e+03, 1.15e+04, 3.58e+04, 1.1e+05, 6.26e+04, 0, 0, 0, 0, 0, 0, 0, 0,
04480    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 321, 9.21e+03, 2.5e+04,
04481    2.35e+04, 955, 39.9, 45.9, 116, 505, 218, 901, 499, 553, 996, 1.22e+03,
04482    2.97e+03, 509, 524, 532, 2.63e+03, 3.7e+04, 1.41e+04, 1.17e+04, 6.89e+03,
04483    1.04e+04, 2.03e+03, 1.41e+03, 114, 0, 0, 0, 0.00111, 208, 2.58e+05,
04484    3.87e+05, 6.12e+05, 6.35e+03, 2.89, 0, 11.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04485    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04486    0, 0, 0, 0, 0, 0, 0, 0, 3.18e+04, 3.25e+04, 0, 147, 126, 0, 650, 3.09e+04,
04487    9.52e+03, 0, 0, 0, 0, 0, 0, 428, 8.98e+04, 9.92e+04, 4.66e+04, 3.69e+04,
04488    1.12e+05, 8.77e+04, 1.24e+05, 1.9e+05, 1.1e+03, 3.51e+03, 4.89e+03, 50.7,
04489    52.4, 34.8, 0.0555, 84.8, 57.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04490   {4.08e+03, 2.07e+03, 600, 1.32e+03, 1.16e+03, 2.21e+03, 1.69e+03, 999,
04491    4.05e+03, 307, 240, 1.96e+03, 473, 260, 122, 151, 699, 2.86e+03, 2.27e+03,
04492    1.7e+03, 1.46e+04, 4.01e+04, 1.49e+05, 137, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04493    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 575, 2.32e+04, 2.64e+04, 2.77, 0,
04494    1.04e+04, 2.35e+04, 112, 70, 66.6, 737, 1.01e+03, 3.43e+03, 153, 371, 790,
04495    737, 1.77e+03, 1.72e+03, 519, 472, 762, 4.98e+03, 1.38e+04, 5.64e+03,
04496    1.16e+04, 3.53e+04, 7.6e+04, 7.88e+03, 228, 0, 0, 0, 0, 0, 1.48e+03,
04497    3.4e+05, 5.54e+05, 1.96e+05, 2.69e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04498    0, 0, 0, 0, 77.6, 196, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04499    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.01e+03, 4.24e+04, 0, 0, 0, 0, 6.05, 0,
04500    0, 0, 0, 0, 0, 0, 0, 0, 1.94e+03, 7.04e+04, 1.21e+05, 3.15e+04, 6.94e+04,
04501    4.52e+04, 1.09e+05, 1.48e+05, 3.1e+05, 65.9, 7.84e+03, 1.95e+03, 35.9,
04502    38.3, 5.86, 0, 12.7, 72.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04503   {4.13e+03, 2.26e+03, 626, 1.39e+03, 1.15e+03, 2.21e+03, 1.58e+03, 1.34e+03,
04504    4.17e+03, 331, 556, 2.13e+03, 448, 197, 79.5, 66.7, 877, 2.63e+03,
04505    2.57e+03, 2.26e+03, 2.07e+04, 1.08e+05, 6.99e+04, 0, 0, 0, 0, 0, 0, 0, 0,
04506    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 574, 5.53e+04, 9.17e+04,
04507    5.13e+04, 0, 5.33e+03, 1.08e+05, 7.87e+04, 1.12e+03, 168, 3.05e+03,
04508    2.92e+03, 2.72e+03, 1.73e+03, 181, 107, 507, 832, 826, 2.08e+03, 6.23e+03,
04509    2.53e+03, 2.93e+03, 3.4e+03, 2.98e+03, 6.08e+04, 9.68e+04, 7.54e+04, 0, 0,
04510    0, 0, 0, 0, 0, 11.1, 9.82e+04, 4.58e+05, 6.23e+05, 2.73e+04, 0, 0, 0, 0, 0,
04511    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 0.00111, 0, 0, 0, 0, 0,
04512    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.22,
04513    1.17e+04, 435, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.27e+04, 4.73e+04,
04514    6.4e+04, 3.35e+04, 4.2e+04, 3.04e+04, 1.55e+05, 1.06e+05, 9.3e+04, 217,
04515    2.79e+03, 518, 147, 99.7, 0.138, 0, 59.2, 11.4, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04516    0, 0, 0, 0, 0, 0},
04517   {4.2e+03, 2.44e+03, 644, 1.47e+03, 1.12e+03, 2.26e+03, 1.39e+03, 1.65e+03,
04518    4.35e+03, 678, 2.27e+03, 2.39e+03, 418, 151, 70.8, 72.3, 1.31e+03, 2.4e+03,
04519    3.28e+03, 3.35e+03, 2.8e+04, 1.1e+05, 1.09e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04520    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.29e+03, 8e+04, 1.07e+05,
04521    2.84e+04, 0, 289, 6.61e+04, 1.45e+05, 2.86e+04, 4.3e+03, 1.75e+03,
04522    1.86e+03, 2.19e+03, 1.23e+03, 157, 65.3, 660, 3.64e+03, 4.91e+03, 6.64e+03,
04523    6.84e+03, 5.53e+03, 757, 3.94e+03, 2.23e+04, 7.01e+03, 0, 0, 0, 0, 0, 0, 0,
04524    0, 4.29e+03, 5.68e+05, 7.33e+05, 4.99e+05, 1.08e+06, 3.46e+03, 0, 0,
04525    2.82e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04526    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 422,
04527    1.07e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.97e+04, 3.3e+04,
04528    6.02e+04, 3.55e+04, 2.46e+04, 3.42e+04, 1.52e+05, 1.32e+05, 4.78e+03,
04529    3.28e+03, 2.98e+03, 193, 136, 119, 0, 0, 268, 24.8, 0, 0, 0, 0, 0, 0, 0, 0,
04530    0, 0, 0, 0, 0, 0},
04531   {4.24e+03, 2.62e+03, 654, 1.55e+03, 1.12e+03, 2.35e+03, 1.2e+03, 1.66e+03,
04532    5.82e+03, 1.21e+03, 6.08e+03, 2.78e+03, 400, 168, 89, 162, 1.37e+03,
04533    2.7e+03, 4.41e+03, 4.88e+03, 2.07e+04, 7.75e+04, 3.24e+03, 0, 0, 0, 0, 0,
04534    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.27e+03,
04535    7.58e+04, 4.84e+04, 1.47e+03, 0, 1.36e+04, 1.23e+05, 1.77e+05, 2.89e+04,
04536    9.8e+03, 9.65e+03, 1.68e+03, 670, 42.4, 140, 554, 2.75e+03, 4.94e+03,
04537    3.58e+04, 6.4e+03, 4.22e+03, 3.63e+03, 1.9e+04, 1.25e+04, 0, 0, 0, 0, 0, 0,
04538    0, 0, 0, 1.58e+05, 5.84e+05, 4.64e+04, 1e+04, 1.19e+05, 239, 0, 0, 0.0765,
04539    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04540    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.33e+04, 0,
04541    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9.99e+03, 4.13e+04, 5.65e+04, 4.49e+04,
04542    7.13e+04, 8.95e+04, 9.96e+04, 1e+05, 3.28e+04, 2.28e+04, 5.2e+03, 5.48e+03,
04543    143, 91.5, 0, 0, 290, 0, 0.00444, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04544   {4.27e+03, 2.8e+03, 652, 1.63e+03, 1.18e+03, 2.52e+03, 1.21e+03, 1.52e+03,
04545    9.84e+03, 684, 7.22e+03, 2.66e+03, 353, 246, 232, 408, 1.11e+03, 2.61e+03,
04546    4.96e+03, 7.21e+03, 1.68e+04, 5.98e+04, 3.8e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04547    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.92e+03,
04548    5.62e+04, 4.64e+04, 2.87e+04, 3.83e+04, 3.23e+04, 7.2e+03, 2.37e+04, 299,
04549    114, 1.12e+03, 1.54e+03, 4.96e+03, 1.57e+04, 1.59e+04, 3.21e+03, 1.35e+04,
04550    4.23e+04, 2.56e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.66e+05, 2.15e+04, 0, 0,
04551    3.46e+04, 142, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04552    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04553    0, 0, 0, 0, 4.2e+03, 4.14e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04554    4.16e+03, 5.94e+04, 2.7e+04, 1.07e+05, 4.89e+04, 5.81e+04, 8.61e+04,
04555    6.09e+04, 9.63e+03, 1.26e+04, 6.46e+03, 7.86e+03, 2.26e+03, 47.1, 0, 0,
04556    295, 0, 789, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04557   {4.29e+03, 2.95e+03, 648, 1.71e+03, 1.32e+03, 2.81e+03, 1.39e+03, 1.53e+03,
04558    1.6e+04, 1.68e+03, 6.7e+03, 2.71e+03, 343, 488, 425, 760, 578, 1.32e+03,
04559    5.61e+03, 1.06e+04, 2.21e+04, 5.37e+04, 3.15e+03, 0, 0, 0, 0, 0, 0, 0, 0,
04560    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 928,
04561    2.9e+04, 1.21e+05, 5.9e+04, 7.39e+04, 3.88e+04, 2.63e+04, 1.13e+04,
04562    3.85e+04, 3.09e+03, 1.07e+03, 773, 4.03e+03, 9.13e+03, 5.35e+03, 6.44e+03,
04563    168, 44.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.23e+03, 1.3e+05, 2.45e+04, 0, 0,
04564    7.21e+04, 3.47e+03, 0, 3.82e+03, 255, 854, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
04565    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04566    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.25e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04567    0, 1.11e+05, 4.96e+04, 1.06e+05, 4.1e+04, 3.16e+04, 3.59e+04, 6.16e+03,
04568    2.38e+03, 2.84e+03, 1.46e+03, 6.19e+03, 362, 5.16, 0, 0, 171, 0, 0, 0, 0,
04569    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04570   {4.31e+03, 3.08e+03, 629, 1.78e+03, 1.64e+03, 3.54e+03, 1.54e+03, 1.91e+03,
04571    2.54e+04, 3.89e+03, 8.59e+03, 6.05e+03, 474, 1.18e+03, 603, 487, 785, 728,
04572    6.48e+03, 1.31e+04, 2.87e+04, 7.33e+04, 6.16e+03, 0, 0, 0, 0, 0, 0, 0, 0,
04573    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04574    1.61e+04, 1.93e+04, 2.82e+04, 9.21e+04, 7.04e+04, 3.81e+04, 1.7e+04,
04575    3.91e+03, 3.7e+03, 6.23e+03, 9.27e+03, 3.45e+04, 3.53e+03, 53.7, 0, 0, 0,
04576    0, 0, 0, 0, 0, 0, 0, 580, 1.41e+04, 1.1e+04, 0.165, 0, 1.39e+05, 1.28e+05,
04577    0, 3.79e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04578    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04579    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.72e+04, 1.13e+05, 3.51e+04,
04580    8.87e+04, 7.8e+04, 4.41e+04, 3.01e+03, 77.9, 502, 1.04e+03, 341, 9.66e+03,
04581    98.6, 0.0593, 0, 0, 17.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04582   {4.31e+03, 3.2e+03, 604, 1.86e+03, 2.17e+03, 4.82e+03, 1.28e+03, 2.9e+03,
04583    3.56e+04, 1.33e+04, 1.94e+04, 7.77e+03, 736, 732, 1.36e+03, 1.56e+03,
04584    1.28e+03, 752, 6.29e+03, 1.26e+04, 2.64e+04, 1.13e+05, 0, 0, 0, 0, 0, 0, 0,
04585    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04586    0, 0, 0, 1.18e+03, 7.3e+04, 1.07e+05, 7.53e+04, 5.91e+04, 3.49e+04,
04587    2.51e+04, 1.75e+04, 1.21e+04, 1.11e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04588    27.7, 0, 0, 0, 0, 289, 1.6e+05, 4.28e+04, 2.96e+04, 0, 0, 0, 0, 0, 0, 0, 0,
04589    0, 0.00222, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04590    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.84e+03, 0, 0, 0, 0,
04591    0, 0, 0, 0, 0, 0, 0, 0, 4.82e+04, 1.11e+05, 3.37e+04, 3.86e+04, 5.92e+04,
04592    2.23e+04, 2.01e+04, 2e+04, 394, 155, 308, 1.49e+03, 26.4, 0, 0, 0, 0,
04593    0.0211, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04594   {4.3e+03, 3.28e+03, 579, 1.95e+03, 2.94e+03, 5.39e+03, 1.11e+03, 4.36e+03,
04595    3.68e+04, 2.32e+04, 4.16e+04, 3.82e+03, 1.71e+03, 423, 551, 2.98e+03,
04596    3.35e+03, 565, 7.92e+03, 1.01e+04, 2.76e+04, 1.33e+05, 1.08, 0, 0, 0, 0, 0,
04597    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04598    0, 0, 0, 0, 0, 2.32e+03, 1.47e+04, 6.19e+04, 1.69e+04, 4.19e+04, 2.36e+04,
04599    5.14e+03, 832, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 341, 565, 0, 56.1, 0, 0,
04600    0, 2.6e+04, 1.15e+05, 403, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04601    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04602    0, 0, 0, 0, 0, 0, 0, 0, 0, 33.1, 43.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.99e+04,
04603    6.94e+04, 4.25e+04, 4.94e+04, 2.89e+04, 2.29e+04, 2.27e+04, 1.39e+04,
04604    6.55e+03, 816, 621, 4.54e+03, 29.5, 0, 0, 0, 0, 30.3, 0, 0, 0, 0, 0, 0, 0, 0,
04605    0, 0, 0, 0, 0, 0, 0},
04606   {4.29e+03, 3.37e+03, 551, 2.04e+03, 3.81e+03, 5.09e+03, 1.87e+03, 6.72e+03,
04607    3.28e+04, 3.69e+04, 6.96e+04, 5.8e+03, 6.99e+03, 1.01e+03, 3.55e+03,
04608    4.12e+03, 4.36e+03, 331, 8.1e+03, 7.39e+03, 3.21e+04, 6.49e+04, 2.87e+03,
04609    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04610    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 142, 39.7, 5.75, 0, 0, 0, 0, 0, 0,
04611    0, 0, 0, 0, 0, 0, 0, 0, 198, 770, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04612    0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04613    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.21e+03,
04614    554, 0, 0, 0, 0, 0, 0, 0, 0, 2.27e+04, 7.53e+04, 5.8e+04, 3.16e+04,
04615    2.03e+04, 2.27e+04, 2.11e+04, 2.35e+04, 1.13e+04, 252, 64.3, 1.57e+03, 93,
04616    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04617   {4.26e+03, 3.44e+03, 518, 2.15e+03, 4.79e+03, 4.34e+03, 3.92e+03, 1.23e+04,
04618    2.35e+04, 6.4e+04, 7.4e+04, 1.43e+04, 1.28e+04, 2.62e+03, 4.71e+03,
04619    2.65e+03, 5.21e+03, 192, 6.06e+03, 7.38e+03, 4.04e+04, 1.06e+05, 0, 0, 0,
04620    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04621    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04622    0, 0, 0, 0, 0, 0, 0, 0, 0, 1.28e+03, 1.39e+05, 72.8, 0, 0, 0, 0, 0, 0, 0, 0,
04623    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04624    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.92e+03,
04625    1.95e+04, 0, 0, 0, 0, 0, 0, 0, 0, 1.54e+04, 1.43e+04, 9.71e+04, 5.15e+04,
04626    4.63e+04, 5.2e+04, 1.79e+04, 3.11e+04, 6.97e+03, 247, 107, 400, 209, 0, 0,
04627    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04628   {4.23e+03, 3.49e+03, 491, 2.24e+03, 6.05e+03, 3.67e+03, 4.44e+03, 3.65e+04,
04629    1.6e+04, 1.65e+05, 1.05e+05, 3.19e+04, 2.23e+04, 3.64e+03, 8.14e+03,
04630    5.72e+03, 4.2e+03, 408, 3.52e+03, 9.32e+03, 3.16e+04, 1.65e+05, 0, 0, 0, 0,
04631    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04632    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04633    0, 0, 0, 0, 0, 0, 0, 0, 1.34e+04, 4.34e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04634    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04635    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5.61e+03, 2.7e+04,
04636    4.15, 799, 4.96e+03, 8.09e+03, 2.5e+03, 270, 0, 0, 0, 1.12e+05, 6.06e+04,
04637    2.78e+04, 1.34e+04, 1.27e+04, 1.79e+04, 7.72e+03, 1.36e+03, 142, 128, 81,
04638    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04639   {4.16e+03, 3.51e+03, 460, 2.34e+03, 7.58e+03, 2.84e+03, 4.78e+03, 7.71e+04,
04640    4.58e+04, 3.63e+05, 3.08e+05, 5.99e+04, 3.03e+04, 5.46e+04, 2.02e+04,
04641    1.49e+04, 3.27e+03, 631, 2.09e+03, 1.23e+04, 3.95e+04, 1.26e+05, 0, 0, 0,
04642    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04643    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04644    0, 0, 0, 0, 0, 0, 395, 1.55e+04, 1.32e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04645    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04646    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.21e+04,
04647    4.77e+04, 6.33e+04, 1.25e+05, 1.18e+05, 6.52e+04, 5.71e+03, 695, 0, 0,
04648    2.32e+03, 1.42e+05, 3.59e+04, 3.71e+04, 9.24e+03, 1.91e+04, 1.53e+04,
04649    4.67e+03, 189, 74.4, 130, 0, 0, 0, 0, 0, 0.0143, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04650    0, 0, 0, 0},
04651   {4.11e+03, 3.53e+03, 430, 2.42e+03, 9.39e+03, 1.92e+03, 1.22e+04, 5.31e+04,
```

```
04652    6.55e+04, 2.96e+05, 4.49e+05, 7.98e+04, 2.31e+05, 1.24e+06, 2.14e+05,
04653    1.13e+04, 8.58e+03, 1.4e+03, 1.94e+03, 1.56e+04, 7.95e+04, 8.58e+04, 0, 0,
04654    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04655    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04656    0, 0, 0, 0, 0, 0, 0, 2.29e+03, 3.19e+03, 193, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04657    0, 0, 0, 883, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04658    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04659    8.18e+04, 9.87e+04, 1.3e+05, 1.25e+05, 1.49e+05, 4.7e+04, 3.37e+03, 0, 0,
04660    0, 1.77e+05, 4.82e+04, 3.35e+04, 2.2e+04, 1.72e+04, 1.2e+04, 7.92e+03, 270,
04661    119, 90.8, 0, 0, 0, 0, 0, 0.00222, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04662   {4.05e+03, 3.52e+03, 403, 2.5e+03, 1.16e+04, 1.53e+03, 2.03e+04, 9.43e+04,
04663    1.6e+05, 5.7e+04, 2.82e+05, 1.65e+05, 8.25e+05, 0, 5.32e+05, 6.68e+04,
04664    9.67e+03, 3.45e+03, 3.56e+03, 1.37e+04, 2.2e+05, 1.38e+04, 0, 0, 0, 0, 0,
04665    0, 0, 0, 0, 0, 0, 0, 0, 47.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04666    0, 0, 0, 0, 0, 0, 0, 0, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04667    0, 0, 0, 0, 0, 0, 0, 0, 156, 1.31e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04668    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04669    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 703,
04670    1.11e+05, 8.19e+04, 1.14e+05, 1.44e+05, 8.58e+04, 4.82e+03, 312, 0, 0,
04671    8.36e+04, 5.48e+04, 4.15e+04, 4.18e+04, 1.3e+04, 1.81e+04, 1.6e+04, 221,
04672    141, 37.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04673   {3.98e+03, 3.5e+03, 376, 2.55e+03, 1.48e+04, 1.8e+03, 3.23e+04, 2.65e+05,
04674    2.51e+05, 7.63e+04, 1.17e+05, 1.91e+05, 7.19e+05, 0, 0, 2.4e+05, 6.08e+03,
04675    4.88e+03, 3.83e+04, 2.02e+05, 2.25e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04676    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04677    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26.6, 0.00111,
04678    9.51e+04, 7.09e+03, 7.78e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04679    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04680    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.91e+03,
04681    7.91e+04, 1.05e+05, 3.35e+05, 1.67e+05, 8.23e+04, 1.96e+04, 0, 1.85e+04,
04682    2.36e+04, 7.27e+04, 3.88e+04, 3.9e+04, 3.36e+04, 1.37e+04, 4.56e+04, 72.8,
04683    88.9, 5.03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04684   {3.91e+03, 3.49e+03, 350, 2.59e+03, 1.88e+04, 2.7e+03, 4.82e+04, 6.06e+05,
04685    1.24e+05, 2.83e+04, 1.15e+04, 4.54e+05, 3.87e+05, 1.39e+05, 4.41e+03,
04686    3.81e+05, 3.67e+04, 5e+04, 9.47e+04, 6.12e+04, 4.47e+04, 0, 0, 0, 0, 0, 0,
04687    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04688    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04689    0, 0.00222, 0, 1.75e+04, 3.8e+04, 3.27e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04690    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04691    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04692    0, 0.444, 2.06e+05, 3.84e+05, 8.62e+04, 1.63e+05, 9.67e+04, 2.93e+04,
04693    4.58e+04, 7.95e+04, 5.8e+04, 2.23e+04, 2.01e+04, 6.49e+04, 1.05e+04,
04694    1.04e+04, 71.7, 9.9, 104, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04695    0, 0, 0},
04696   {3.81e+03, 3.49e+03, 329, 2.59e+03, 2.28e+04, 6.02e+03, 1.29e+05, 3.05e+05,
04697    2.2e+05, 0, 0, 1.19e+05, 4.17e+05, 1.28e+05, 0, 1.19e+05, 3.51e+05,
04698    1.13e+05, 7.5e+04, 1.09e+05, 8.92e+03, 0, 0, 0, 3.2e+03, 0, 0, 0, 0, 0, 0,
04699    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04700    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04701    1.86e+04, 2.7e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04702    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04703    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.75e+04,
04704    2.51e+04, 4.67e+04, 1.43e+04, 1.24e+05, 8.73e+04, 5.86e+03, 4.72e+04,
04705    7.88e+04, 2.09e+04, 3.83e+04, 3.06e+04, 1.2e+04, 1.72e+03, 2.36e+04, 624,
04706    7.89, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04707   {3.71e+03, 3.49e+03, 307, 2.54e+03, 2.43e+04, 2.07e+04, 6.54e+04, 3.88e+05,
04708    1.88e+05, 0, 0, 1.97e+05, 5.46e+05, 4.06e+04, 0, 3e+04, 1.26e+06, 9.55e+04,
04709    1.16e+05, 3.18e+05, 4.19e+03, 0, 0, 0.00444, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04710    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04711    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04712    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.28, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04713    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04714    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.27e+04, 5.8e+03, 1.41e+04,
04715    1.68e+04, 3.36e+04, 2.13e+03, 2.03e+04, 6.88e+04, 2.07e+04, 3.07e+04,
04716    7.23e+04, 1.62e+04, 1.21e+04, 2.18e+04, 9.07e+03, 0, 0, 0, 0, 0, 0, 0, 0,
04717    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04718   {3.62e+03, 3.52e+03, 290, 2.45e+03, 2.34e+04, 7.91e+04, 5.86e+04, 3.01e+05,
04719    7.08e+04, 0, 0, 2.07e+05, 2.33e+05, 1.69e+03, 0, 0, 1.08e+06, 1.53e+05,
04720    4.6e+04, 3.04e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04721    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04722    0, 0, 0, 0, 0, 3.82e+03, 2.65e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04723    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04724    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04725    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.07e+03, 1.22e+04, 3.31e+03,
04726    2.88e+04, 5.86e+04, 2.74e+04, 2.74e+04, 4.27e+04, 1.97e+04, 2.34e+04,
04727    2.54e+04, 4.23e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04728    0, 0},
04729   {3.51e+03, 3.54e+03, 277, 2.33e+03, 2.06e+04, 1.24e+05, 1.89e+05, 6.27e+05,
04730    6.05e+04, 0, 0, 1.06e+04, 1.82e+05, 0, 0, 0, 1.38e+05, 4.08e+05, 2.03e+05,
04731    2.08e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04732    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04733    0, 945, 6.32e+04, 5.76e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04734    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04735    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04736    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.54e+03, 1.68e+04, 3.6e+04, 1.8e+04,
04737    3.99e+04, 3.06e+04, 3.64e+04, 4.55e+04, 1.55e+04, 2.12e+04, 7.43e+04, 297,
04738    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

```
04739   {3.4e+03, 3.58e+03, 266, 2.18e+03, 1.88e+04, 2.32e+05, 4.76e+05, 1.07e+06,
04740    0.111, 0, 0, 4.69e+03, 2.27e+04, 109, 0, 0, 6.58e+03, 5.04e+05, 1.57e+05,
04741    3.1e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 494, 0,
04742    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04743    0, 0, 7.31e+04, 1.44e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0217, 0, 0, 0, 0, 0,
04744    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04745    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04746    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.49e+03, 0, 0, 0, 0, 4.67e+03, 3.35e+04,
04747    2.04e+04, 3.32e+04, 3.51e+04, 4.8e+04, 4.29e+04, 4.43e+04, 5.69e+04,
04748    6.6e+04, 137, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04749   {3.29e+03, 3.62e+03, 259, 2.04e+03, 2.07e+04, 2.25e+05, 9.06e+05, 5.98e+05,
04750    0, 0, 0, 1.41e+04, 2.86e+05, 1.97e+05, 0, 0, 0, 1.02e+06, 9.27e+04,
04751    4.18e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04752    0, 0, 1.45e+05, 1.18e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04753    0, 0, 0, 0, 0, 0, 10.4, 2.64e+04, 72.3, 0, 0, 0, 29.9, 7.11e+03, 5.3e+04,
04754    2.36e+03, 0, 1.05e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04755    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04756    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04757    1.85e+03, 0, 0, 0, 0, 2.69e+04, 3.83e+04, 1.99e+04, 1.82e+04, 3.56e+04,
04758    3.63e+04, 2.86e+04, 4.38e+04, 4.73e+04, 6.85e+04, 21.5, 0, 0, 0, 0, 0, 0,
04759    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04760   {3.18e+03, 3.66e+03, 258, 1.91e+03, 3.34e+04, 1.15e+05, 5.63e+05, 5.52e+04,
04761    0, 0, 0, 0, 2.09e+05, 1.14e+05, 0, 0, 0, 7.53e+05, 5.01e+05, 3.74e+05, 0,
04762    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6.64e+03,
04763    7.43e+04, 2.15e+05, 1e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04764    0, 0, 0, 0, 0.00111, 30.1, 7.44, 0, 0, 1.92e+03, 1.43e+03, 1.67e+04,
04765    8.56e+03, 1.42e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04766    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04767    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04768    463, 0, 0, 0, 0, 344, 4.23e+04, 2.7e+04, 8.64e+03, 5.14e+04, 2.89e+04,
04769    4.18e+04, 5.18e+04, 4.68e+04, 2.55e+04, 167, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04770    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04771   {3.07e+03, 3.72e+03, 257, 1.85e+03, 4.51e+04, 1.23e+05, 3.44e+05, 5.01e+04,
04772    0, 0, 0, 0, 5.44e+04, 0, 0, 0, 0, 4.78e+05, 6.57e+05, 1.29e+05, 0, 0, 0, 0,
04773    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 56.5, 0, 0, 0, 0, 0, 0, 2.34e+03,
04774    5.81e+04, 2.06e+05, 3.35e+05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04775    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.85e+03, 7.6e+03, 0.0397, 0, 0, 0, 0, 0, 0, 0, 0,
04776    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04777    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04778    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.76e+04,
04779    3.73e+04, 1.72e+04, 1.82e+04, 3.79e+04, 3.12e+04, 1.56e+04, 8.2e+04,
04780    2.6e+04, 74.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04781   {2.97e+03, 3.78e+03, 261, 1.92e+03, 4.21e+04, 3.43e+05, 8.42e+04, 2.2e+04,
04782    0, 0, 0, 0, 0, 0, 0, 0, 0, 21, 5.48e+05, 2.99e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04783    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 223, 8.26e+04, 1.08e+05,
04784    3e+05, 9.93e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04785    0, 0, 0, 1.19e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04786    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04787    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04788    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.57e+04, 6.76e+04, 7.99e+04, 5.26e+04,
04789    1.26e+04, 4.46e+04, 1.66e+04, 3.15e+04, 9.64e+04, 3.94e+03, 1.72e+03, 0, 0,
04790    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04791   {2.87e+03, 3.84e+03, 272, 2.14e+03, 3.66e+04, 2.69e+05, 3.6e+04, 1.86e+03,
04792    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.93e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04793    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.33e+03, 5.49e+04, 2.61e+05,
04794    4.07e+05, 1.35e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04795    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04796    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04797    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04798    0, 0, 0, 0, 0, 0, 0, 0, 0, 2.1e+03, 6.04e+04, 8.58e+04, 3.47e+04, 5.27e+03,
04799    5.78e+03, 3.93e+04, 2.68e+04, 4.73e+04, 8.61e+03, 3.56e+04, 0, 0, 0, 0, 0,
04800    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04801   {2.76e+03, 3.92e+03, 288, 2.44e+03, 2.71e+04, 3.67e+05, 2.46e+05, 0, 0, 0,
04802    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04803    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.54e+04, 1.71e+05, 1.56e+05, 1.18e+05, 0,
04804    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04805    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04806    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04807    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04808    9.77e+04, 9.84e+04, 1.45e+04, 2.13e+04, 1.87e+04, 4.14e+04, 1.05e+04,
04809    3.37e+04, 2.27e+04, 5.64e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04810    0, 0, 0, 0, 0},
04811   {2.66e+03, 3.99e+03, 310, 2.57e+03, 1.71e+04, 2.32e+05, 7.55e+05, 0, 0, 0,
04812    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04813    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8.12e+03, 5.62e+04, 1.48e+05, 1.33e+05, 0,
04814    0, 0, 0, 508, 3.28e+03, 111, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04815    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00332, 0, 0, 0,
04816    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04817    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 36,
04818    1.07e+04, 0, 0, 0, 0, 0, 0, 0, 7.67e+03, 5.08e+04, 1.8e+04, 3.17e+04,
04819    1.07e+04, 2.34e+04, 2.27e+04, 2.12e+04, 2.29e+04, 6.13e+03, 0, 0, 0, 0, 0,
04820    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04821   {2.56e+03, 4.08e+03, 339, 2.52e+03, 1.02e+04, 9.32e+04, 8.13e+05, 0, 0, 0,
04822    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04823    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.75e+05, 2.58e+04, 1.26e+03,
04824    2.98e+04, 0, 114, 1.22e+04, 5.74e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04825    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00111,
```

```
04826    0.00115, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04827    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04828    0, 0, 0, 0, 112, 0.00111, 0, 0, 0, 0, 0, 0, 0, 0, 3.71e+04, 2.71e+04,
04829    2.81e+04, 5.9e+03, 1.47e+04, 1.06e+04, 3.04e+04, 2.37e+04, 3.12e+03, 0, 0,
04830    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04831   {2.47e+03, 4.17e+03, 376, 2.41e+03, 8.45e+03, 1.07e+05, 1.21e+06, 0, 0, 0,
04832    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04833    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.13e+03, 5.74e+04, 2.5e+04,
04834    4.05e+04, 1.88e+04, 5.05e+03, 1.22e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04835    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00111,
04836    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04837    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04838    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.31e+04, 2.97e+04, 2.63e+04, 9.57e+03,
04839    4.5e+04, 3.24e+04, 2.77e+04, 5.04e+04, 7.47e+03, 0, 0, 0, 0, 0, 0, 0, 0,
04840    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04841   {2.37e+03, 4.27e+03, 423, 2.28e+03, 8.44e+03, 2.9e+05, 1.03e+06, 0, 0, 0, 0,
04842    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04843    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.16e+04, 8.12e+04, 9.27e+04,
04844    3.67e+04, 1.08e+04, 78.6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04845    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04846    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04847    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 216, 0, 0,
04848    0, 0, 0, 0, 0, 0, 0, 3.16e+03, 5.83e+04, 8.1e+03, 3.76e+03, 1.69e+04,
04849    3.9e+04, 2.17e+04, 7.48e+04, 6.82e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04850    0, 0, 0, 0, 0, 0, 0, 0},
04851   {2.27e+03, 4.36e+03, 484, 2.23e+03, 8.46e+03, 4.88e+05, 3.87e+05, 0, 0, 0,
04852    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04853    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.57e+03, 9.47e+04, 5.31e+04,
04854    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 881, 4.27, 0, 0,
04855    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04856    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04857    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 160, 0, 0, 0, 0, 0, 0, 0,
04858    0, 0, 3.27, 4.62e+04, 656, 1.03e+03, 2.49e+04, 4.76e+04, 4.35e+04,
04859    7.91e+04, 5.3e+03, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04860    0},
04861   {2.17e+03, 4.45e+03, 562, 2.23e+03, 8.39e+03, 6.52e+05, 1.69e+05, 0, 0, 0,
04862    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04863    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.01e+04, 8.21e+04, 0, 0,
04864    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 299, 4.46e+04, 2.39e+03, 0,
04865    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04866    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04867    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 249, 0, 0, 0, 0, 0, 0,
04868    0, 0, 0, 123, 2.85e+04, 952, 1.17e+04, 2.34e+04, 4.44e+04, 5.62e+04,
04869    7.28e+04, 86, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04870   {2.07e+03, 4.53e+03, 659, 2.26e+03, 9.27e+03, 4.83e+05, 7.38e+04, 0, 0, 0,
04871    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04872    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 19.2, 0, 0, 0, 0, 0, 0,
04873    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 646, 1.02e+04, 526, 0, 0, 0, 0, 0,
04874    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04875    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04876    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22.8, 951, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04877    2.54e+03, 2.05e+04, 94.3, 1.7e+04, 1.91e+04, 4.2e+04, 6.17e+04, 7.83e+04,
04878    0, 1.22e+04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
04879   {1.99e+03, 4.59e+03, 760, 2.29e+03, 1.27e+04, 3.52e+05, 4.25e+04, 0, 0, 0,
04880    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04881    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04882    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00208, 0, 8.18e+03, 3.06e+03, 0, 0, 0,
04883    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04884    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
04885    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.93e+03, 0, 0, 0, 0, 0, 0, 0, 0,
04886    0, 0, 1.07e+03, 0, 1.19e+03, 3.53e+04, 4.31e+04, 7.14e+04, 4.5e+04, 0,
04887    2.11e+04, 2.41, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
04888 };
```

5.33 trac.c File Reference

Lagrangian particle dispersion model.

Functions

- void init_simtime (ctl_t *ctl, atm_t *atm)

  Set simulation time interval.
- void module_advection (met_t *met0, met_t *met1, atm_t *atm, int ip, double dt)

  Calculate advection of air parcels.
- void module_decay (ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, int ip, double dt)

Calculate exponential decay of particle mass.

- void module_diffusion_meso (ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, int ip, double dt, gsl_rng *rng)

  Calculate mesoscale diffusion.

- void module_diffusion_turb (ctl_t *ctl, atm_t *atm, int ip, double dt, gsl_rng *rng)

  Calculate turbulent diffusion.

- void module_isosurf (ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, int ip)

  Force air parcels to stay on isosurface.

- void module_meteo (ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, int ip)

  Interpolate meteorological data for air parcel positions.

- void module_position (met_t *met0, met_t *met1, atm_t *atm, int ip)

  Check position of air parcels.

- void module_sedi (ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, int ip, double dt)

  Calculate sedimentation of air parcels.

- void write_output (const char *dirname, ctl_t *ctl, met_t *met0, met_t *met1, atm_t *atm, double t)

  Write simulation output.

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

5.33.1 Detailed Description

Lagrangian particle dispersion model.

Definition in file trac.c.

5.33.2 Function Documentation

5.33.2.1 void init_simtime (ctl_t * *ctl*, atm_t * *atm*)

Set simulation time interval.

Definition at line 399 of file trac.c.

```
00401               {
00402
00403   /* Set inital and final time... */
00404   if (ctl->direction == 1) {
00405     if (ctl->t_start < -1e99)
00406       ctl->t_start = gsl_stats_min(atm->time, 1, (size_t) atm->np);
00407     if (ctl->t_stop < -1e99)
00408       ctl->t_stop = gsl_stats_max(atm->time, 1, (size_t) atm->np);
00409   } else if (ctl->direction == -1) {
00410     if (ctl->t_stop < -1e99)
00411       ctl->t_stop = gsl_stats_min(atm->time, 1, (size_t) atm->np);
00412     if (ctl->t_start < -1e99)
00413       ctl->t_start = gsl_stats_max(atm->time, 1, (size_t) atm->np);
00414   }
00415
00416   /* Check time... */
00417   if (ctl->direction * (ctl->t_stop - ctl->t_start) <= 0)
00418     ERRMSG("Nothing to do!");
00419 }
```

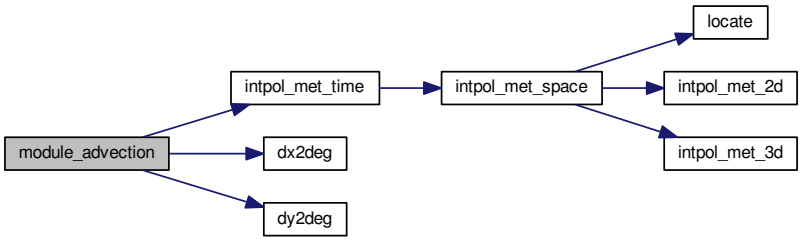
5.33.2.2 void module_advection (met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*, double *dt*)

Calculate advection of air parcels.

Definition at line 423 of file trac.c.

```
00428             {
00429
00430   double v[3], xm[3];
00431
00432   /* Interpolate meteorological data... */
00433   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00434                   atm->lon[ip], atm->lat[ip], NULL, NULL,
00435                   &v[0], &v[1], &v[2], NULL, NULL);
00436
00437   /* Get position of the mid point... */
00438   xm[0] = atm->lon[ip] + dx2deg(0.5 * dt * v[0] / 1000., atm->lat[ip]);
00439   xm[1] = atm->lat[ip] + dy2deg(0.5 * dt * v[1] / 1000.);
00440   xm[2] = atm->p[ip] + 0.5 * dt * v[2];
00441
00442   /* Interpolate meteorological data for mid point... */
00443   intpol_met_time(met0, met1, atm->time[ip] + 0.5 * dt,
00444                   xm[2], xm[0], xm[1], NULL, NULL,
00445                   &v[0], &v[1], &v[2], NULL, NULL);
00446
00447   /* Save new position... */
00448   atm->time[ip] += dt;
00449   atm->lon[ip] += dx2deg(dt * v[0] / 1000., xm[1]);
00450   atm->lat[ip] += dy2deg(dt * v[1] / 1000.);
00451   atm->p[ip] += dt * v[2];
00452 }
```

Here is the call graph for this function:



5.33.2.3 void module_decay (ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*, double *dt*)

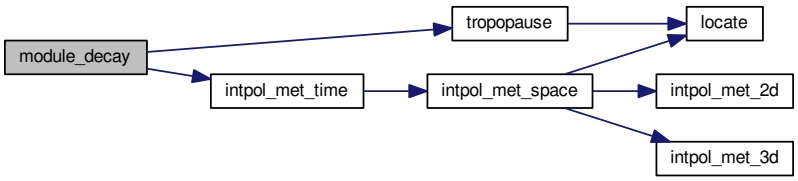
Calculate exponential decay of particle mass.

Definition at line 456 of file trac.c.

```
00462             {
00463
00464   double ps, pt, tdec;
00465
00466   /* Check lifetime values... */
00467   if ((ctl->tdec_trop <= 0 && ctl->tdec_strat <= 0) || ctl->
      qnt_m < 0)
00468     return;
00469
00470   /* Set constant lifetime... */
00471   if (ctl->tdec_trop == ctl->tdec_strat)
00472     tdec = ctl->tdec_trop;
00473
00474   /* Set altitude-dependent lifetime... */
```

```
00475   else {
00476
00477     /* Get surface pressure... */
00478     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00479                     atm->lon[ip], atm->lat[ip], &ps, NULL,
00480                     NULL, NULL, NULL, NULL, NULL);
00481
00482     /* Get tropopause pressure... */
00483     pt = tropopause(atm->time[ip], atm->lat[ip]);
00484
00485     /* Set lifetime... */
00486     if (atm->p[ip] <= pt)
00487       tdec = ctl->tdec_strat;
00488     else
00489       tdec = LIN(ps, ctl->tdec_trop, pt, ctl->tdec_strat, atm->
      p[ip]);
00490   }
00491
00492   /* Calculate exponential decay... */
00493   atm->q[ctl->qnt_m][ip] *= exp(-dt / tdec);
00494 }
```

Here is the call graph for this function:



5.33.2.4 void module_diffusion_meso (ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*, double *dt*, gsl_rng * *rng*)

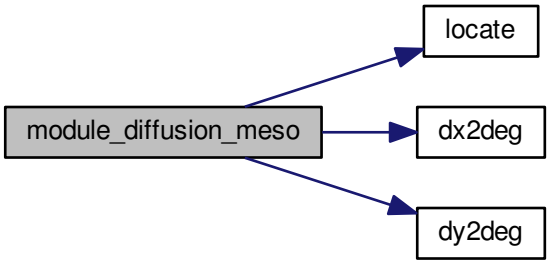
Calculate mesoscale diffusion.

Definition at line 498 of file trac.c.

```
00505                 {
00506
00507   double r, rs, u[16], v[16], w[16], usig, vsig, wsig;
00508
00509   int ix, iy, iz;
00510
00511   /* Calculate mesoscale velocity fluctuations... */
00512   if (ctl->turb_meso > 0) {
00513
00514     /* Get indices... */
00515     ix = locate(met0->lon, met0->nx, atm->lon[ip]);
00516     iy = locate(met0->lat, met0->ny, atm->lat[ip]);
00517     iz = locate(met0->p, met0->np, atm->p[ip]);
00518
00519     /* Collect local wind data... */
00520     u[0] = met0->u[ix][iy][iz];
00521     u[1] = met0->u[ix + 1][iy][iz];
00522     u[2] = met0->u[ix][iy + 1][iz];
00523     u[3] = met0->u[ix + 1][iy + 1][iz];
00524     u[4] = met0->u[ix][iy][iz + 1];
00525     u[5] = met0->u[ix + 1][iy][iz + 1];
00526     u[6] = met0->u[ix][iy + 1][iz + 1];
00527     u[7] = met0->u[ix + 1][iy + 1][iz + 1];
00528
00529     v[0] = met0->v[ix][iy][iz];
00530     v[1] = met0->v[ix + 1][iy][iz];
00531     v[2] = met0->v[ix][iy + 1][iz];
00532     v[3] = met0->v[ix + 1][iy + 1][iz];
00533     v[4] = met0->v[ix][iy][iz + 1];
```

```
00534     v[5] = met0->v[ix + 1][iy][iz + 1];
00535     v[6] = met0->v[ix][iy + 1][iz + 1];
00536     v[7] = met0->v[ix + 1][iy + 1][iz + 1];
00537
00538     w[0] = met0->w[ix][iy][iz];
00539     w[1] = met0->w[ix + 1][iy][iz];
00540     w[2] = met0->w[ix][iy + 1][iz];
00541     w[3] = met0->w[ix + 1][iy + 1][iz];
00542     w[4] = met0->w[ix][iy][iz + 1];
00543     w[5] = met0->w[ix + 1][iy][iz + 1];
00544     w[6] = met0->w[ix][iy + 1][iz + 1];
00545     w[7] = met0->w[ix + 1][iy + 1][iz + 1];
00546
00547     /* Get indices... */
00548     ix = locate(met1->lon, met1->nx, atm->lon[ip]);
00549     iy = locate(met1->lat, met1->ny, atm->lat[ip]);
00550     iz = locate(met1->p, met1->np, atm->p[ip]);
00551
00552     /* Collect local wind data... */
00553     u[8] = met1->u[ix][iy][iz];
00554     u[9] = met1->u[ix + 1][iy][iz];
00555     u[10] = met1->u[ix][iy + 1][iz];
00556     u[11] = met1->u[ix + 1][iy + 1][iz];
00557     u[12] = met1->u[ix][iy][iz + 1];
00558     u[13] = met1->u[ix + 1][iy][iz + 1];
00559     u[14] = met1->u[ix][iy + 1][iz + 1];
00560     u[15] = met1->u[ix + 1][iy + 1][iz + 1];
00561
00562     v[8] = met1->v[ix][iy][iz];
00563     v[9] = met1->v[ix + 1][iy][iz];
00564     v[10] = met1->v[ix][iy + 1][iz];
00565     v[11] = met1->v[ix + 1][iy + 1][iz];
00566     v[12] = met1->v[ix][iy][iz + 1];
00567     v[13] = met1->v[ix + 1][iy][iz + 1];
00568     v[14] = met1->v[ix][iy + 1][iz + 1];
00569     v[15] = met1->v[ix + 1][iy + 1][iz + 1];
00570
00571     w[8] = met1->w[ix][iy][iz];
00572     w[9] = met1->w[ix + 1][iy][iz];
00573     w[10] = met1->w[ix][iy + 1][iz];
00574     w[11] = met1->w[ix + 1][iy + 1][iz];
00575     w[12] = met1->w[ix][iy][iz + 1];
00576     w[13] = met1->w[ix + 1][iy][iz + 1];
00577     w[14] = met1->w[ix][iy + 1][iz + 1];
00578     w[15] = met1->w[ix + 1][iy + 1][iz + 1];
00579
00580     /* Get standard deviations of local wind data... */
00581     usig = gsl_stats_sd(u, 1, 16);
00582     vsig = gsl_stats_sd(v, 1, 16);
00583     wsig = gsl_stats_sd(w, 1, 16);
00584
00585     /* Set temporal correlations for mesoscale fluctuations... */
00586     r = 1 - 2 * fabs(dt) / ctl->dt_met;
00587     rs = sqrt(1 - r * r);
00588
00589     /* Calculate mesoscale wind fluctuations... */
00590     atm->up[ip] =
00591       r * atm->up[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00592                                                        ctl->turb_meso * usig);
00593     atm->vp[ip] =
00594       r * atm->vp[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00595                                                        ctl->turb_meso * vsig);
00596     atm->wp[ip] =
00597       r * atm->wp[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00598                                                        ctl->turb_meso * wsig);
00599
00600     /* Calculate air parcel displacement... */
00601     atm->lon[ip] += dx2deg(atm->up[ip] * dt / 1000., atm->lat[ip]);
00602     atm->lat[ip] += dy2deg(atm->vp[ip] * dt / 1000.);
00603     atm->p[ip] += atm->wp[ip] * dt;
00604   }
00605 }
```

Here is the call graph for this function:



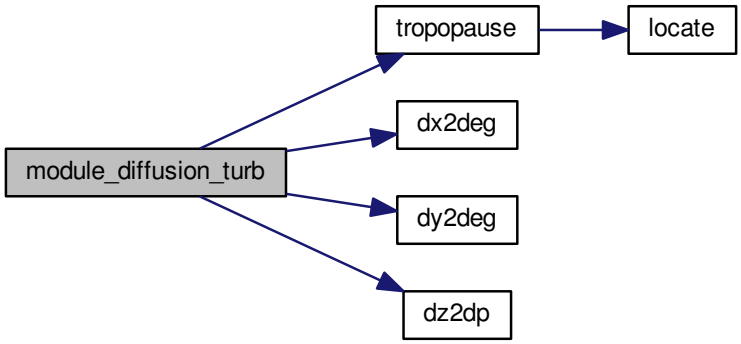
5.33.2.5 void module_diffusion_turb (ctl_t * *ctl*, atm_t * *atm*, int *ip*, double *dt*, gsl_rng * *rng*)

Calculate turbulent diffusion.

Definition at line 609 of file trac.c.

```
00614                 {
00615
00616   double dx, dz, pt, p0, p1, w;
00617
00618   /* Get tropopause pressure... */
00619   pt = tropopause(atm->time[ip], atm->lat[ip]);
00620
00621   /* Get weighting factor... */
00622   p1 = pt * 0.866877899;
00623   p0 = pt / 0.866877899;
00624   if (atm->p[ip] > p0)
00625     w = 1;
00626   else if (atm->p[ip] < p1)
00627     w = 0;
00628   else
00629     w = LIN(p0, 1.0, p1, 0.0, atm->p[ip]);
00630
00631   /* Set diffusivitiy... */
00632   dx = w * ctl->turb_dx_trop + (1 - w) * ctl->turb_dx_strat;
00633   dz = w * ctl->turb_dz_trop + (1 - w) * ctl->turb_dz_strat;
00634
00635   /* Horizontal turbulent diffusion... */
00636   if (dx > 0) {
00637     atm->lon[ip]
00638       += dx2deg(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dx * fabs(dt)))
00639                 / 1000., atm->lat[ip]);
00640     atm->lat[ip]
00641       += dy2deg(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dx * fabs(dt)))
00642                 / 1000.);
00643   }
00644
00645   /* Vertical turbulent diffusion... */
00646   if (dz > 0)
00647     atm->p[ip]
00648       += dz2dp(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dz * fabs(dt)))
00649                / 1000., atm->p[ip]);
00650 }
```

Here is the call graph for this function:



5.33.2.6 void module_isosurf (ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*)

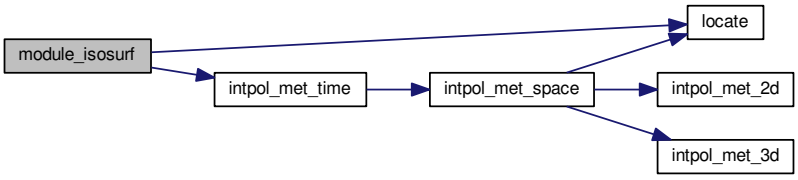
Force air parcels to stay on isosurface.

Definition at line 654 of file trac.c.

```
00659           {
00660
00661   static double *iso, *ps, t, *ts;
00662
00663   static int idx, ip2, n, nb = 100000;
00664
00665   FILE *in;
00666
00667   char line[LEN];
00668
00669   /* Check control parameter... */
00670   if (ctl->isosurf < 1 || ctl->isosurf > 4)
00671     return;
00672
00673   /* Initialize... */
00674   if (ip < 0) {
00675
00676     /* Allocate... */
00677     ALLOC(iso, double,
00678           NP);
00679     ALLOC(ps, double,
00680           nb);
00681     ALLOC(ts, double,
00682           nb);
00683
00684     /* Save pressure... */
00685     if (ctl->isosurf == 1)
00686       for (ip2 = 0; ip2 < atm->np; ip2++)
00687         iso[ip2] = atm->p[ip2];
00688
00689     /* Save density... */
00690     else if (ctl->isosurf == 2)
00691       for (ip2 = 0; ip2 < atm->np; ip2++) {
00692         intpol_met_time(met0, met1, atm->time[ip2], atm->p[ip2],
00693                         atm->lon[ip2], atm->lat[ip2], NULL, &t, NULL, NULL,
00694                         NULL, NULL, NULL);
00695         iso[ip2] = atm->p[ip2] / t;
00696       }
00697
00698     /* Save potential temperature... */
00699     else if (ctl->isosurf == 3)
00700       for (ip2 = 0; ip2 < atm->np; ip2++) {
00701         intpol_met_time(met0, met1, atm->time[ip2], atm->p[ip2],
00702                         atm->lon[ip2], atm->lat[ip2], NULL, &t, NULL, NULL,
```

```
00703                         NULL, NULL, NULL);
00704         iso[ip2] = t * pow(P0 / atm->p[ip2], 0.286);
00705       }
00706
00707     /* Read balloon pressure data... */
00708     else if (ctl->isosurf == 4) {
00709
00710       /* Write info... */
00711       printf("Read balloon pressure data: %s\n", ctl->balloon);
00712
00713       /* Open file... */
00714       if (!(in = fopen(ctl->balloon, "r")))
00715         ERRMSG("Cannot open file!");
00716
00717       /* Read pressure time series... */
00718       while (fgets(line, LEN, in))
00719         if (sscanf(line, "%lg %lg", &ts[n], &ps[n]) == 2)
00720           if ((++n) > 100000)
00721             ERRMSG("Too many data points!");
00722
00723       /* Check number of points... */
00724       if (n < 1)
00725         ERRMSG("Could not read any data!");
00726
00727       /* Close file... */
00728       fclose(in);
00729     }
00730
00731     /* Leave initialization... */
00732     return;
00733   }
00734
00735   /* Restore pressure... */
00736   if (ctl->isosurf == 1)
00737     atm->p[ip] = iso[ip];
00738
00739   /* Restore density... */
00740   else if (ctl->isosurf == 2) {
00741     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00742                     atm->lat[ip], NULL, &t, NULL, NULL, NULL, NULL, NULL);
00743     atm->p[ip] = iso[ip] * t;
00744   }
00745
00746   /* Restore potential temperature... */
00747   else if (ctl->isosurf == 3) {
00748     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00749                     atm->lat[ip], NULL, &t, NULL, NULL, NULL, NULL, NULL);
00750     atm->p[ip] = P0 * pow(iso[ip] / t, -1. / 0.286);
00751   }
00752
00753   /* Interpolate pressure... */
00754   else if (ctl->isosurf == 4) {
00755     if (atm->time[ip] <= ts[0])
00756       atm->p[ip] = ps[0];
00757     else if (atm->time[ip] >= ts[n - 1])
00758       atm->p[ip] = ps[n - 1];
00759     else {
00760       idx = locate(ts, n, atm->time[ip]);
00761       atm->p[ip] = LIN(ts[idx], ps[idx],
00762                        ts[idx + 1], ps[idx + 1], atm->time[ip]);
00763     }
00764   }
00765 }
```

Here is the call graph for this function:



5.33.2.7 void module_meteo (ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*)

Interpolate meteorological data for air parcel positions.

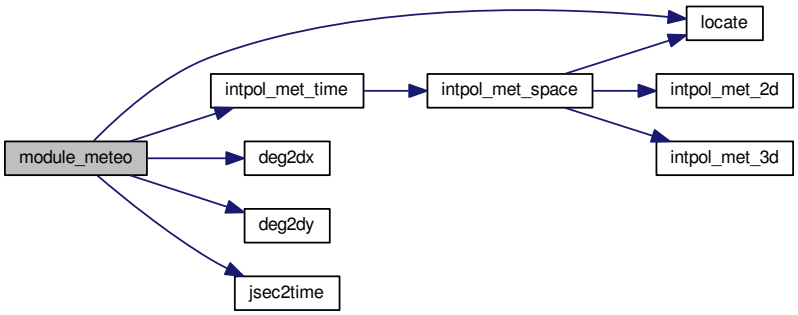
Definition at line 769 of file trac.c.

```
00774           {
00775
00776 #include "topo.c"
00777
00778   static FILE *in;
00779
00780   static char filename[LEN], line[LEN];
00781
00782   static double lon[GX], lat[GY], var[GX][GY],
00783     rdum, rlat, rlat_old = -999, rlon, rvar;
00784
00785   static int year_old, mon_old, day_old, nlon, nlat;
00786
00787   double b, c, ps, p1, p_hno3, p_h2o, t, t1, term1, term2,
00788     u, u1, v, v1, w, x1, x2, h2o, o3, grad, vort, var0, var1;
00789
00790   int day, mon, year, idum, ilat, ilon;
00791
00792   /* Interpolate meteorological data... */
00793   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00794                   atm->lat[ip], &ps, &t, &u, &v, &w, &h2o, &o3);
00795
00796   /* Set surface pressure... */
00797   if (ctl->qnt_ps >= 0)
00798     atm->q[ctl->qnt_ps][ip] = ps;
00799
00800   /* Set pressure... */
00801   if (ctl->qnt_p >= 0)
00802     atm->q[ctl->qnt_p][ip] = atm->p[ip];
00803
00804   /* Set temperature... */
00805   if (ctl->qnt_t >= 0)
00806     atm->q[ctl->qnt_t][ip] = t;
00807
00808   /* Set zonal wind... */
00809   if (ctl->qnt_u >= 0)
00810     atm->q[ctl->qnt_u][ip] = u;
00811
00812   /* Set meridional wind... */
00813   if (ctl->qnt_v >= 0)
00814     atm->q[ctl->qnt_v][ip] = v;
00815
00816   /* Set vertical velocity... */
00817   if (ctl->qnt_w >= 0)
00818     atm->q[ctl->qnt_w][ip] = w;
00819
00820   /* Set water vapor vmr... */
00821   if (ctl->qnt_h2o >= 0)
00822     atm->q[ctl->qnt_h2o][ip] = h2o;
00823
00824   /* Set ozone vmr... */
00825   if (ctl->qnt_o3 >= 0)
00826     atm->q[ctl->qnt_o3][ip] = o3;
00827
00828   /* Calculate potential temperature... */
00829   if (ctl->qnt_theta >= 0)
00830     atm->q[ctl->qnt_theta][ip] = t * pow(P0 / atm->p[ip], 0.286);
00831
00832   /* Calculate potential vorticity... */
00833   if (ctl->qnt_pv >= 0) {
00834
00835     /* Absolute vorticity... */
00836     vort = 2 * 7.2921e-5 * sin(atm->lat[ip] * M_PI / 180.);
00837     if (fabs(atm->lat[ip]) < 89.) {
00838       intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00839                       (atm->lon[ip] >=
00840                        0 ? atm->lon[ip] - 1. : atm->lon[ip] + 1.),
00841                       atm->lat[ip], NULL, NULL, NULL, &v1, NULL, NULL, NULL);
00842       vort += (v1 - v) / 1000.
00843         / ((atm->lon[ip] >= 0 ? -1 : 1) * deg2dx(1., atm->lat[ip]));
00844     }
00845     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00846                     (atm->lat[ip] >=
00847                      0 ? atm->lat[ip] - 1. : atm->lat[ip] + 1.), NULL, NULL,
00848                     &u1, NULL, NULL, NULL, NULL);
```

```
00849     vort += (u1 - u) / 1000. / ((atm->lat[ip] >= 0 ? -1 : 1) * deg2dy(1.));
00850
00851     /* Potential temperature gradient... */
00852     p1 = 0.85 * atm->p[ip];
00853     intpol_met_time(met0, met1, atm->time[ip], p1, atm->lon[ip],
00854                     atm->lat[ip], NULL, &t1, NULL, NULL, NULL, NULL, NULL);
00855     grad = (t1 * pow(P0 / p1, 0.286) - t * pow(P0 / atm->p[ip], 0.286))
00856       / (100. * (p1 - atm->p[ip]));
00857
00858     /* Calculate PV... */
00859     atm->q[ctl->qnt_pv][ip] = -1e6 * G0 * vort * grad;
00860   }
00861
00862   /* Calculate T_ice (Marti and Mauersberger, 1993)... */
00863   if (ctl->qnt_tice >= 0 || ctl->qnt_tsts >= 0)
00864     atm->q[ctl->qnt_tice][ip] = -2663.5
00865       / (log10(ctl->psc_h2o * atm->p[ip] * 100.) - 12.537);
00866
00867   /* Calculate T_NAT (Hanson and Mauersberger, 1988)... */
00868   if (ctl->qnt_tnat >= 0 || ctl->qnt_tsts >= 0) {
00869     p_hno3 = ctl->psc_hno3 * atm->p[ip] / 1.333224;
00870     p_h2o = ctl->psc_h2o * atm->p[ip] / 1.333224;
00871     term1 = 38.9855 - log10(p_hno3) - 2.7836 * log10(p_h2o);
00872     term2 = 0.009179 - 0.00088 * log10(p_h2o);
00873     b = term1 / term2;
00874     c = -11397.0 / term2;
00875     x1 = (-b + sqrt(b * b - 4. * c)) / 2.;
00876     x2 = (-b - sqrt(b * b - 4. * c)) / 2.;
00877     if (x1 > 0)
00878       atm->q[ctl->qnt_tnat][ip] = x1;
00879     if (x2 > 0)
00880       atm->q[ctl->qnt_tnat][ip] = x2;
00881   }
00882
00883   /* Calculate T_STS (mean of T_ice and T_NAT)... */
00884   if (ctl->qnt_tsts >= 0) {
00885     if (ctl->qnt_tice < 0 || ctl->qnt_tnat < 0)
00886       ERRMSG("Need T_ice and T_NAT to calculate T_STS!");
00887     atm->q[ctl->qnt_tsts][ip] = 0.5 * (atm->q[ctl->qnt_tice][ip]
00888                                        + atm->q[ctl->qnt_tnat][ip]);
00889   }
00890
00891   /* Interpolate low-level (750 hPa) wind... */
00892   if (ctl->qnt_gw_wind >= 0)
00893     intpol_met_time(met0, met1, atm->time[ip], 750., atm->lon[ip],
00894                     atm->lat[ip], NULL, NULL, &atm->q[ctl->qnt_gw_wind][ip],
00895                     NULL, NULL, NULL, NULL);
00896
00897   /* Interpolate terrain height variance... */
00898   if (ctl->qnt_gw_sso >= 0) {
00899     ilat = locate(topo_lat, TOPO_NLAT, atm->lat[ip]);
00900     ilon = locate(topo_lon, TOPO_NLON, atm->lon[ip]);
00901     var0 = LIN(topo_lat[ilat], topo_zvar[ilon][ilat],
00902                topo_lat[ilat + 1], topo_zvar[ilon][ilat + 1], atm->lat[ip]);
00903     var1 = LIN(topo_lat[ilat], topo_zvar[ilon + 1][ilat],
00904                topo_lat[ilat + 1], topo_zvar[ilon + 1][ilat + 1],
00905                atm->lat[ip]);
00906     atm->q[ctl->qnt_gw_sso][ip]
00907       = LIN(topo_lon[ilon], var0, topo_lon[ilon + 1], var1, atm->lon[ip]);
00908   }
00909
00910   /* Read variance data for current day... */
00911   if (ip == 0 && ctl->qnt_gw_var >= 0) {
00912     jsec2time(atm->time[ip], &year, &mon, &day, &idum, &idum, &idum, &rdum);
00913     if (year != year_old || mon != mon_old || day != day_old) {
00914       year_old = year;
00915       mon_old = mon;
00916       day_old = day;
00917       nlon = nlat = -1;
00918       sprintf(filename, "%s_%d_%02d_%02d.tab",
00919               ctl->gw_basename, year, mon, day);
00920       if ((in = fopen(filename, "r"))) {
00921         printf("Read gravity wave data: %s\n", filename);
00922         while (fgets(line, LEN, in)) {
00923           if (sscanf(line, "%lg %lg %lg", &rlon, &rlat, &rvar) != 3)
00924             continue;
00925           if (rlat != rlat_old) {
00926             rlat_old = rlat;
00927             if ((++nlat) > GY)
00928               ERRMSG("Too many latitudes!");
00929             nlon = -1;
00930           }
00931           if ((++nlon) > GX)
00932             ERRMSG("Too many longitudes!");
00933           lon[nlon] = rlon;
00934           lat[nlat] = rlat;
00935           var[nlon][nlat] = GSL_MAX(0, rvar);
```

```
00936         }
00937         fclose(in);
00938         nlat++;
00939         nlon++;
00940       } else
00941         printf("Missing gravity wave data: %s\n", filename);
00942     }
00943   }
00944
00945   /* Interpolate variance data... */
00946   if (ctl->qnt_gw_var >= 0) {
00947     if (nlat >= 2 && nlon >= 2) {
00948       ilat = locate(lat, nlat, atm->lat[ip]);
00949       ilon = locate(lon, nlon, atm->lon[ip]);
00950       var0 = LIN(lat[ilat], var[ilon][ilat],
00951                  lat[ilat + 1], var[ilon][ilat + 1], atm->lat[ip]);
00952       var1 = LIN(lat[ilat], var[ilon + 1][ilat],
00953                  lat[ilat + 1], var[ilon + 1][ilat + 1], atm->lat[ip]);
00954       atm->q[ctl->qnt_gw_var][ip]
00955         = LIN(lon[ilon], var0, lon[ilon + 1], var1, atm->lon[ip]);
00956     } else
00957       atm->q[ctl->qnt_gw_var][ip] = GSL_NAN;
00958   }
00959 }
```

Here is the call graph for this function:



5.33.2.8 void module_position (met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*)

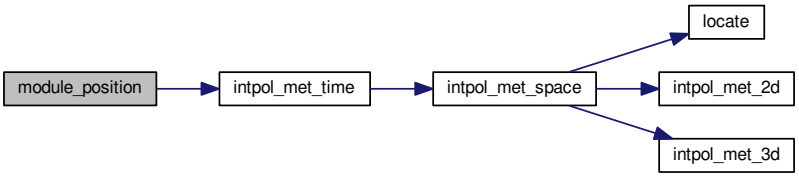
Check position of air parcels.

Definition at line 963 of file trac.c.

```
00967           {
00968
00969   double ps;
00970
00971   /* Calculate modulo... */
00972   atm->lon[ip] = fmod(atm->lon[ip], 360);
00973   atm->lat[ip] = fmod(atm->lat[ip], 360);
00974
00975   /* Check latitude... */
00976   while (atm->lat[ip] < -90 || atm->lat[ip] > 90) {
00977     if (atm->lat[ip] > 90) {
00978       atm->lat[ip] = 180 - atm->lat[ip];
00979       atm->lon[ip] += 180;
00980     }
00981     if (atm->lat[ip] < -90) {
00982       atm->lat[ip] = -180 - atm->lat[ip];
00983       atm->lon[ip] += 180;
00984     }
00985   }
00986
00987   /* Check longitude... */
```

```
00988   while (atm->lon[ip] < -180)
00989     atm->lon[ip] += 360;
00990   while (atm->lon[ip] >= 180)
00991     atm->lon[ip] -= 360;
00992
00993   /* Get surface pressure... */
00994   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00995                   atm->lon[ip], atm->lat[ip], &ps, NULL,
00996                   NULL, NULL, NULL, NULL, NULL);
00997
00998   /* Check pressure... */
00999   if (atm->p[ip] > ps)
01000     atm->p[ip] = ps;
01001   else if (atm->p[ip] < met0->p[met0->np - 1])
01002     atm->p[ip] = met0->p[met0->np - 1];
01003 }
```

Here is the call graph for this function:



5.33.2.9 void module_sedi (ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, int *ip*, double *dt*)

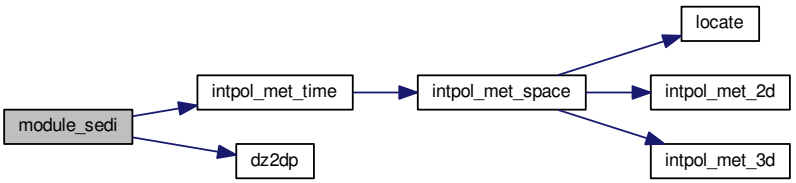
Calculate sedimentation of air parcels.

Definition at line 1007 of file trac.c.

```
01013             {
01014
01015   /* Coefficients for Cunningham slip-flow correction (Kasten, 1968): */
01016   const double A = 1.249, B = 0.42, C = 0.87;
01017
01018   /* Specific gas constant for dry air [J/(kg K)]: */
01019   const double R = 287.058;
01020
01021   /* Average mass of an air molecule [kg/molec]: */
01022   const double m = 4.8096e-26;
01023
01024   double G, K, eta, lambda, p, r_p, rho, rho_p, T, v, v_p;
01025
01026   /* Check if parameters are available... */
01027   if (ctl->qnt_r < 0 || ctl->qnt_rho < 0)
01028     return;
01029
01030   /* Convert units... */
01031   p = 100 * atm->p[ip];
01032   r_p = 1e-6 * atm->q[ctl->qnt_r][ip];
01033   rho_p = atm->q[ctl->qnt_rho][ip];
01034
01035   /* Get temperature... */
01036   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
01037                   atm->lat[ip], NULL, &T, NULL, NULL, NULL, NULL, NULL);
01038
01039   /* Density of dry air... */
01040   rho = p / (R * T);
01041
01042   /* Dynamic viscosity of air... */
01043   eta = 1.8325e-5 * (416.16 / (T + 120.)) * pow(T / 296.16, 1.5);
01044
01045   /* Thermal velocity of an air molecule... */
01046   v = sqrt(8 * GSL_CONST_MKSA_BOLTZMANN * T / (M_PI * m));
```

```
01047
01048   /* Mean free path of an air molecule... */
01049   lambda = 2 * eta / (rho * v);
01050
01051   /* Knudsen number for air... */
01052   K = lambda / r_p;
01053
01054   /* Cunningham slip-flow correction... */
01055   G = 1 + K * (A + B * exp(-C / K));
01056
01057   /* Sedimentation (fall) velocity... */
01058   v_p =
01059     2. * gsl_pow_2(r_p) * (rho_p -
01060                            rho) * GSL_CONST_MKSA_GRAV_ACCEL / (9. * eta) * G;
01061
01062   /* Calculate pressure change... */
01063   atm->p[ip] += dz2dp(v_p * dt / 1000., atm->p[ip]);
01064 }
```

Here is the call graph for this function:



5.33.2.10 void write_output (const char * *dirname*, ctl_t * *ctl*, met_t * *met0*, met_t * *met1*, atm_t * *atm*, double *t*)

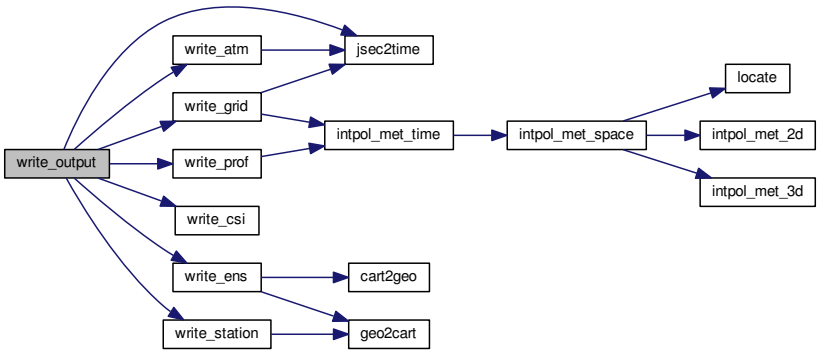
Write simulation output.

Definition at line 1068 of file trac.c.

```
01074           {
01075
01076   char filename[LEN];
01077
01078   double r;
01079
01080   int year, mon, day, hour, min, sec;
01081
01082   /* Get time... */
01083   jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01084
01085   /* Write atmospheric data... */
01086   if (ctl->atm_basename[0] != '-' && fmod(t, ctl->atm_dt_out) == 0) {
01087     sprintf(filename, "%s/%s_%04d_%02d_%02d_%02d_%02d.tab",
01088             dirname, ctl->atm_basename, year, mon, day, hour, min);
01089     write_atm(filename, ctl, atm, t);
01090   }
01091
01092   /* Write CSI data... */
01093   if (ctl->csi_basename[0] != '-') {
01094     sprintf(filename, "%s/%s.tab", dirname, ctl->csi_basename);
01095     write_csi(filename, ctl, atm, t);
01096   }
01097
01098   /* Write ensemble data... */
01099   if (ctl->ens_basename[0] != '-') {
01100     sprintf(filename, "%s/%s.tab", dirname, ctl->ens_basename);
01101     write_ens(filename, ctl, atm, t);
01102   }
01103
01104   /* Write gridded data... */
01105   if (ctl->grid_basename[0] != '-' && fmod(t, ctl->grid_dt_out) == 0) {
01106     sprintf(filename, "%s/%s_%04d_%02d_%02d_%02d_%02d.tab",
```

```
01107             dirname, ctl->grid_basename, year, mon, day, hour, min);
01108     write_grid(filename, ctl, met0, met1, atm, t);
01109   }
01110
01111   /* Write profile data... */
01112   if (ctl->prof_basename[0] != '-') {
01113     sprintf(filename, "%s/%s.tab", dirname, ctl->prof_basename);
01114     write_prof(filename, ctl, met0, met1, atm, t);
01115   }
01116
01117   /* Write station data... */
01118   if (ctl->stat_basename[0] != '-') {
01119     sprintf(filename, "%s/%s.tab", dirname, ctl->stat_basename);
01120     write_station(filename, ctl, atm, t);
01121   }
01122 }
```

Here is the call graph for this function:



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

Definition at line 160 of file trac.c.

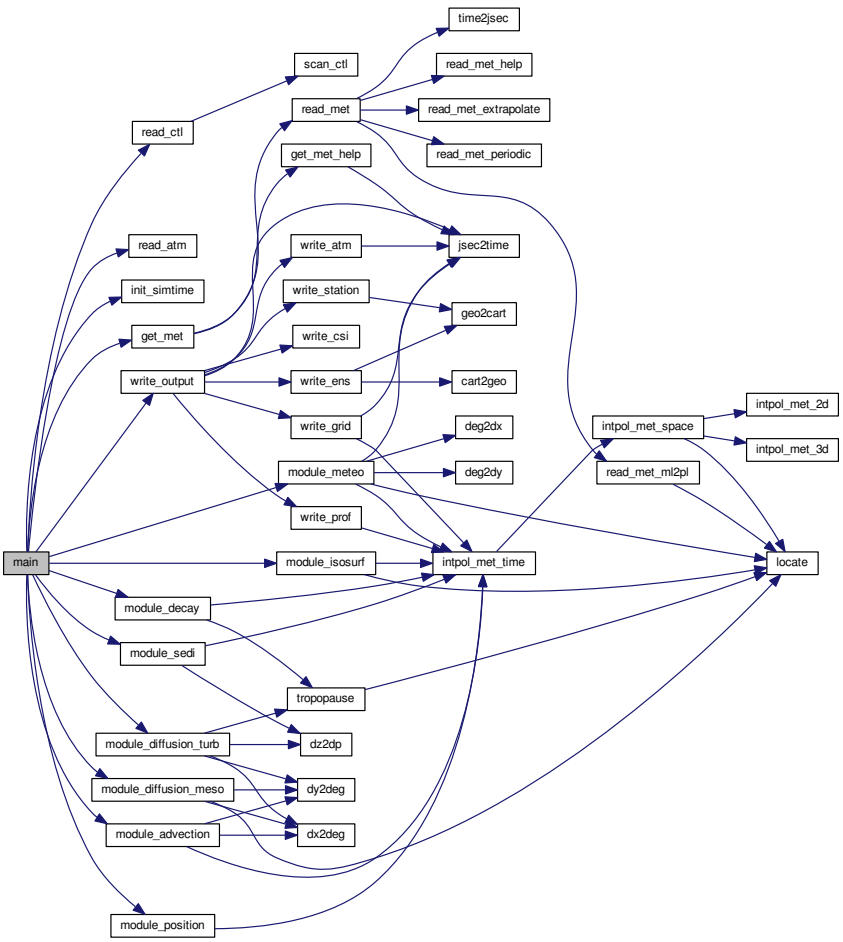
```
00162                {
00163
00164   ctl_t ctl;
00165
00166   atm_t *atm;
00167
00168   met_t *met0, *met1;
00169
00170   gsl_rng *rng[NTHREADS];
00171
00172   FILE *dirlist;
00173
00174   char dirname[LEN], filename[LEN];
00175
00176   double *dt, t, t0;
00177
00178   int i, ip, ntask = 0, rank = 0, size = 1;
00179
00180 #ifdef MPI
00181   /* Initialize MPI... */
00182   MPI_Init(&argc, &argv);
00183   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
00184   MPI_Comm_size(MPI_COMM_WORLD, &size);
00185 #endif
00186
00187   /* Check arguments... */
00188   if (argc < 5)
00189     ERRMSG("Give parameters: <dirlist> <ctl> <atm_in> <metbase>");
00190
00191   /* Open directory list... */
00192   if (!(dirlist = fopen(argv[1], "r")))
```

```
00193     ERRMSG("Cannot open directory list!");
00194
00195   /* Loop over directories... */
00196   while (fscanf(dirlist, "%s", dirname) != EOF) {
00197
00198     /* MPI parallelization... */
00199     if ((++ntask) % size != rank)
00200       continue;
00201
00202     /* ------------------------------------------------------------
00203        Initialize model run...
00204        ------------------------------------------------------------ */
00205
00206     /* Set timers... */
00207     START_TIMER(TIMER_TOTAL);
00208     START_TIMER(TIMER_INIT);
00209
00210     /* Allocate... */
00211     ALLOC(atm, atm_t, 1);
00212     ALLOC(met0, met_t, 1);
00213     ALLOC(met1, met_t, 1);
00214     ALLOC(dt, double,
00215           NP);
00216
00217     /* Read control parameters... */
00218     sprintf(filename, "%s/%s", dirname, argv[2]);
00219     read_ctl(filename, argc, argv, &ctl);
00220
00221     /* Initialize random number generators... */
00222     gsl_rng_env_setup();
00223     for (i = 0; i < NTHREADS; i++)
00224       rng[i] = gsl_rng_alloc(gsl_rng_default);
00225
00226     /* Read atmospheric data... */
00227     sprintf(filename, "%s/%s", dirname, argv[3]);
00228     read_atm(filename, &ctl, atm);
00229
00230     /* Get simulation time interval... */
00231     init_simtime(&ctl, atm);
00232
00233     /* Get rounded start time... */
00234     if (ctl.direction == 1)
00235       t0 = floor(ctl.t_start / ctl.dt_mod) * ctl.dt_mod;
00236     else
00237       t0 = ceil(ctl.t_start / ctl.dt_mod) * ctl.dt_mod;
00238
00239     /* Set timers... */
00240     STOP_TIMER(TIMER_INIT);
00241
00242     /* ------------------------------------------------------------
00243        Loop over timesteps...
00244        ------------------------------------------------------------ */
00245
00246     /* Loop over timesteps... */
00247     for (t = t0; ctl.direction * (t - ctl.t_stop) < ctl.dt_mod;
00248          t += ctl.direction * ctl.dt_mod) {
00249
00250       /* Adjust length of final time step... */
00251       if (ctl.direction * (t - ctl.t_stop) > 0)
00252         t = ctl.t_stop;
00253
00254       /* Set time steps for air parcels... */
00255       for (ip = 0; ip < atm->np; ip++)
00256         if ((ctl.direction * (atm->time[ip] - ctl.t_start) >= 0
00257              && ctl.direction * (atm->time[ip] - ctl.t_stop) <= 0
00258              && ctl.direction * (atm->time[ip] - t) < 0))
00259           dt[ip] = t - atm->time[ip];
00260         else
00261           dt[ip] = GSL_NAN;
00262
00263       /* Get meteorological data... */
00264       START_TIMER(TIMER_INPUT);
00265       get_met(&ctl, argv[4], t, met0, met1);
00266       STOP_TIMER(TIMER_INPUT);
00267
00268       /* Initialize isosurface... */
00269       START_TIMER(TIMER_ISOSURF);
00270       if (t == t0)
00271         module_isosurf(&ctl, met0, met1, atm, -1);
00272       STOP_TIMER(TIMER_ISOSURF);
00273
00274       /* Advection... */
00275       START_TIMER(TIMER_ADVECT);
00276 #pragma omp parallel for default(shared) private(ip)
00277       for (ip = 0; ip < atm->np; ip++)
00278         if (gsl_finite(dt[ip]))
00279           module_advection(met0, met1, atm, ip, dt[ip]);
```

```
00280       STOP_TIMER(TIMER_ADVECT);
00281
00282       /* Turbulent diffusion... */
00283       START_TIMER(TIMER_DIFFTURB);
00284 #pragma omp parallel for default(shared) private(ip)
00285       for (ip = 0; ip < atm->np; ip++)
00286         if (gsl_finite(dt[ip]))
00287           module_diffusion_turb(&ctl, atm, ip, dt[ip],
00288                                 rng[omp_get_thread_num()]);
00289       STOP_TIMER(TIMER_DIFFTURB);
00290
00291       /* Mesoscale diffusion... */
00292       START_TIMER(TIMER_DIFFMESO);
00293 #pragma omp parallel for default(shared) private(ip)
00294       for (ip = 0; ip < atm->np; ip++)
00295         if (gsl_finite(dt[ip]))
00296           module_diffusion_meso(&ctl, met0, met1, atm, ip, dt[ip],
00297                                 rng[omp_get_thread_num()]);
00298       STOP_TIMER(TIMER_DIFFMESO);
00299
00300       /* Sedimentation... */
00301       START_TIMER(TIMER_SEDI);
00302 #pragma omp parallel for default(shared) private(ip)
00303       for (ip = 0; ip < atm->np; ip++)
00304         if (gsl_finite(dt[ip]))
00305           module_sedi(&ctl, met0, met1, atm, ip, dt[ip]);
00306       STOP_TIMER(TIMER_SEDI);
00307
00308       /* Isosurface... */
00309       START_TIMER(TIMER_ISOSURF);
00310 #pragma omp parallel for default(shared) private(ip)
00311       for (ip = 0; ip < atm->np; ip++)
00312         module_isosurf(&ctl, met0, met1, atm, ip);
00313       STOP_TIMER(TIMER_ISOSURF);
00314
00315       /* Position... */
00316       START_TIMER(TIMER_POSITION);
00317 #pragma omp parallel for default(shared) private(ip)
00318       for (ip = 0; ip < atm->np; ip++)
00319         module_position(met0, met1, atm, ip);
00320       STOP_TIMER(TIMER_POSITION);
00321
00322       /* Meteorological data... */
00323       START_TIMER(TIMER_METEO);
00324       module_meteo(&ctl, met0, met1, atm, 0);
00325 #pragma omp parallel for default(shared) private(ip)
00326       for (ip = 1; ip < atm->np; ip++)
00327         module_meteo(&ctl, met0, met1, atm, ip);
00328       STOP_TIMER(TIMER_METEO);
00329
00330       /* Decay... */
00331       START_TIMER(TIMER_DECAY);
00332 #pragma omp parallel for default(shared) private(ip)
00333       for (ip = 0; ip < atm->np; ip++)
00334         if (gsl_finite(dt[ip]))
00335           module_decay(&ctl, met0, met1, atm, ip, dt[ip]);
00336       STOP_TIMER(TIMER_DECAY);
00337
00338       /* Write output... */
00339       START_TIMER(TIMER_OUTPUT);
00340       write_output(dirname, &ctl, met0, met1, atm, t);
00341       STOP_TIMER(TIMER_OUTPUT);
00342     }
00343
00344     /* ------------------------------------------------------------
00345        Finalize model run...
00346        ------------------------------------------------------------ */
00347
00348     /* Report timers... */
00349     STOP_TIMER(TIMER_TOTAL);
00350     PRINT_TIMER(TIMER_TOTAL);
00351     PRINT_TIMER(TIMER_INIT);
00352     PRINT_TIMER(TIMER_INPUT);
00353     PRINT_TIMER(TIMER_OUTPUT);
00354     PRINT_TIMER(TIMER_ADVECT);
00355     PRINT_TIMER(TIMER_DECAY);
00356     PRINT_TIMER(TIMER_DIFFMESO);
00357     PRINT_TIMER(TIMER_DIFFTURB);
00358     PRINT_TIMER(TIMER_ISOSURF);
00359     PRINT_TIMER(TIMER_METEO);
00360     PRINT_TIMER(TIMER_POSITION);
00361     PRINT_TIMER(TIMER_SEDI);
00362
00363     /* Report memory usage... */
00364     printf("MEMORY_ATM = %g MByte\n", 2. * sizeof(atm_t) / 1024. / 1024.);
00365     printf("MEMORY_METEO = %g MByte\n", 2. * sizeof(met_t) / 1024. / 1024.);
00366     printf("MEMORY_DYNAMIC = %g MByte\n",
```

```
00367           NP * sizeof(double) / 1024. / 1024.);
00368     printf("MEMORY_STATIC = %g MByte\n",
00369            (((EX + EY) + (2 + NQ) * GX * GY * GZ) * sizeof(double)
00370             + (EX * EY + EX * EY * EP) * sizeof(float)
00371             + (2 * GX * GY * GZ) * sizeof(int)) / 1024. / 1024.);
00372
00373     /* Report problem size... */
00374     printf("SIZE_NP = %d\n", atm->np);
00375     printf("SIZE_TASKS = %d\n", size);
00376     printf("SIZE_THREADS = %d\n", omp_get_max_threads());
00377
00378     /* Free random number generators... */
00379     for (i = 0; i < NTHREADS; i++)
00380       gsl_rng_free(rng[i]);
00381
00382     /* Free... */
00383     free(atm);
00384     free(met0);
00385     free(met1);
00386     free(dt);
00387   }
00388
00389 #ifdef MPI
00390   /* Finalize MPI... */
00391   MPI_Finalize();
00392 #endif
00393
00394   return EXIT_SUCCESS;
00395 }
```

Here is the call graph for this function:



5.34 trac.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 #ifdef MPI
00028 #include "mpi.h"
00029 #endif
00030
00031 /* ------------------------------------------------------------
00032    Defines...
00033    ------------------------------------------------------------ */
00034
00036 #define TIMER_TOTAL 0
00037
00039 #define TIMER_INIT 1
00040
00042 #define TIMER_INPUT 2
00043
00045 #define TIMER_OUTPUT 3
00046
00048 #define TIMER_ADVECT 4
00049
00051 #define TIMER_DECAY 5
00052
00054 #define TIMER_DIFFMESO 6
00055
00057 #define TIMER_DIFFTURB 7
00058
00060 #define TIMER_ISOSURF 8
00061
00063 #define TIMER_METEO 9
00064
00066 #define TIMER_POSITION 10
00067
00069 #define TIMER_SEDI 11
00070
00071 /* ------------------------------------------------------------
00072    Functions...
00073    ------------------------------------------------------------ */
00074
00076 void init_simtime(
00077   ctl_t * ctl,
00078   atm_t * atm);
00079
00081 void module_advection(
00082   met_t * met0,
00083   met_t * met1,
00084   atm_t * atm,
00085   int ip,
00086   double dt);
00087
00089 void module_decay(
00090   ctl_t * ctl,
00091   met_t * met0,
00092   met_t * met1,
00093   atm_t * atm,
00094   int ip,
00095   double dt);
00096
00098 void module_diffusion_meso(
00099   ctl_t * ctl,
00100   met_t * met0,
00101   met_t * met1,
00102   atm_t * atm,
00103   int ip,
00104   double dt,
00105   gsl_rng * rng);
```

```
00106
00108 void module_diffusion_turb(
00109   ctl_t * ctl,
00110   atm_t * atm,
00111   int ip,
00112   double dt,
00113   gsl_rng * rng);
00114
00116 void module_isosurf(
00117   ctl_t * ctl,
00118   met_t * met0,
00119   met_t * met1,
00120   atm_t * atm,
00121   int ip);
00122
00124 void module_meteo(
00125   ctl_t * ctl,
00126   met_t * met0,
00127   met_t * met1,
00128   atm_t * atm,
00129   int ip);
00130
00132 void module_position(
00133   met_t * met0,
00134   met_t * met1,
00135   atm_t * atm,
00136   int ip);
00137
00139 void module_sedi(
00140   ctl_t * ctl,
00141   met_t * met0,
00142   met_t * met1,
00143   atm_t * atm,
00144   int ip,
00145   double dt);
00146
00148 void write_output(
00149   const char *dirname,
00150   ctl_t * ctl,
00151   met_t * met0,
00152   met_t * met1,
00153   atm_t * atm,
00154   double t);
00155
00156 /* ------------------------------------------------------------
00157    Main...
00158    ------------------------------------------------------------ */
00159
00160 int main(
00161   int argc,
00162   char *argv[]) {
00163
00164   ctl_t ctl;
00165
00166   atm_t *atm;
00167
00168   met_t *met0, *met1;
00169
00170   gsl_rng *rng[NTHREADS];
00171
00172   FILE *dirlist;
00173
00174   char dirname[LEN], filename[LEN];
00175
00176   double *dt, t, t0;
00177
00178   int i, ip, ntask = 0, rank = 0, size = 1;
00179
00180 #ifdef MPI
00181   /* Initialize MPI... */
00182   MPI_Init(&argc, &argv);
00183   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
00184   MPI_Comm_size(MPI_COMM_WORLD, &size);
00185 #endif
00186
00187   /* Check arguments... */
00188   if (argc < 5)
00189     ERRMSG("Give parameters: <dirlist> <ctl> <atm_in> <metbase>");
00190
00191   /* Open directory list... */
00192   if (!(dirlist = fopen(argv[1], "r")))
00193     ERRMSG("Cannot open directory list!");
00194
00195   /* Loop over directories... */
00196   while (fscanf(dirlist, "%s", dirname) != EOF) {
00197
00198     /* MPI parallelization... */
```

```
00199     if ((++ntask) % size != rank)
00200       continue;
00201
00202     /* ------------------------------------------------------------
00203        Initialize model run...
00204        ------------------------------------------------------------ */
00205
00206     /* Set timers... */
00207     START_TIMER(TIMER_TOTAL);
00208     START_TIMER(TIMER_INIT);
00209
00210     /* Allocate... */
00211     ALLOC(atm, atm_t, 1);
00212     ALLOC(met0, met_t, 1);
00213     ALLOC(met1, met_t, 1);
00214     ALLOC(dt, double,
00215           NP);
00216
00217     /* Read control parameters... */
00218     sprintf(filename, "%s/%s", dirname, argv[2]);
00219     read_ctl(filename, argc, argv, &ctl);
00220
00221     /* Initialize random number generators... */
00222     gsl_rng_env_setup();
00223     for (i = 0; i < NTHREADS; i++)
00224       rng[i] = gsl_rng_alloc(gsl_rng_default);
00225
00226     /* Read atmospheric data... */
00227     sprintf(filename, "%s/%s", dirname, argv[3]);
00228     read_atm(filename, &ctl, atm);
00229
00230     /* Get simulation time interval... */
00231     init_simtime(&ctl, atm);
00232
00233     /* Get rounded start time... */
00234     if (ctl.direction == 1)
00235       t0 = floor(ctl.t_start / ctl.dt_mod) * ctl.dt_mod;
00236     else
00237       t0 = ceil(ctl.t_start / ctl.dt_mod) * ctl.dt_mod;
00238
00239     /* Set timers... */
00240     STOP_TIMER(TIMER_INIT);
00241
00242     /* ------------------------------------------------------------
00243        Loop over timesteps...
00244        ------------------------------------------------------------ */
00245
00246     /* Loop over timesteps... */
00247     for (t = t0; ctl.direction * (t - ctl.t_stop) < ctl.dt_mod;
00248          t += ctl.direction * ctl.dt_mod) {
00249
00250       /* Adjust length of final time step... */
00251       if (ctl.direction * (t - ctl.t_stop) > 0)
00252         t = ctl.t_stop;
00253
00254       /* Set time steps for air parcels... */
00255       for (ip = 0; ip < atm->np; ip++)
00256         if ((ctl.direction * (atm->time[ip] - ctl.t_start) >= 0
00257              && ctl.direction * (atm->time[ip] - ctl.t_stop) <= 0
00258              && ctl.direction * (atm->time[ip] - t) < 0))
00259           dt[ip] = t - atm->time[ip];
00260         else
00261           dt[ip] = GSL_NAN;
00262
00263       /* Get meteorological data... */
00264       START_TIMER(TIMER_INPUT);
00265       get_met(&ctl, argv[4], t, met0, met1);
00266       STOP_TIMER(TIMER_INPUT);
00267
00268       /* Initialize isosurface... */
00269       START_TIMER(TIMER_ISOSURF);
00270       if (t == t0)
00271         module_isosurf(&ctl, met0, met1, atm, -1);
00272       STOP_TIMER(TIMER_ISOSURF);
00273
00274       /* Advection... */
00275       START_TIMER(TIMER_ADVECT);
00276 #pragma omp parallel for default(shared) private(ip)
00277       for (ip = 0; ip < atm->np; ip++)
00278         if (gsl_finite(dt[ip]))
00279           module_advection(met0, met1, atm, ip, dt[ip]);
00280       STOP_TIMER(TIMER_ADVECT);
00281
00282       /* Turbulent diffusion... */
00283       START_TIMER(TIMER_DIFFTURB);
00284 #pragma omp parallel for default(shared) private(ip)
00285       for (ip = 0; ip < atm->np; ip++)
```

```
00286         if (gsl_finite(dt[ip]))
00287           module_diffusion_turb(&ctl, atm, ip, dt[ip],
00288                                 rng[omp_get_thread_num()]);
00289       STOP_TIMER(TIMER_DIFFTURB);
00290
00291       /* Mesoscale diffusion... */
00292       START_TIMER(TIMER_DIFFMESO);
00293 #pragma omp parallel for default(shared) private(ip)
00294       for (ip = 0; ip < atm->np; ip++)
00295         if (gsl_finite(dt[ip]))
00296           module_diffusion_meso(&ctl, met0, met1, atm, ip, dt[ip],
00297                                 rng[omp_get_thread_num()]);
00298       STOP_TIMER(TIMER_DIFFMESO);
00299
00300       /* Sedimentation... */
00301       START_TIMER(TIMER_SEDI);
00302 #pragma omp parallel for default(shared) private(ip)
00303       for (ip = 0; ip < atm->np; ip++)
00304         if (gsl_finite(dt[ip]))
00305           module_sedi(&ctl, met0, met1, atm, ip, dt[ip]);
00306       STOP_TIMER(TIMER_SEDI);
00307
00308       /* Isosurface... */
00309       START_TIMER(TIMER_ISOSURF);
00310 #pragma omp parallel for default(shared) private(ip)
00311       for (ip = 0; ip < atm->np; ip++)
00312         module_isosurf(&ctl, met0, met1, atm, ip);
00313       STOP_TIMER(TIMER_ISOSURF);
00314
00315       /* Position... */
00316       START_TIMER(TIMER_POSITION);
00317 #pragma omp parallel for default(shared) private(ip)
00318       for (ip = 0; ip < atm->np; ip++)
00319         module_position(met0, met1, atm, ip);
00320       STOP_TIMER(TIMER_POSITION);
00321
00322       /* Meteorological data... */
00323       START_TIMER(TIMER_METEO);
00324       module_meteo(&ctl, met0, met1, atm, 0);
00325 #pragma omp parallel for default(shared) private(ip)
00326       for (ip = 1; ip < atm->np; ip++)
00327         module_meteo(&ctl, met0, met1, atm, ip);
00328       STOP_TIMER(TIMER_METEO);
00329
00330       /* Decay... */
00331       START_TIMER(TIMER_DECAY);
00332 #pragma omp parallel for default(shared) private(ip)
00333       for (ip = 0; ip < atm->np; ip++)
00334         if (gsl_finite(dt[ip]))
00335           module_decay(&ctl, met0, met1, atm, ip, dt[ip]);
00336       STOP_TIMER(TIMER_DECAY);
00337
00338       /* Write output... */
00339       START_TIMER(TIMER_OUTPUT);
00340       write_output(dirname, &ctl, met0, met1, atm, t);
00341       STOP_TIMER(TIMER_OUTPUT);
00342     }
00343
00344     /* ------------------------------------------------------------
00345        Finalize model run...
00346        ------------------------------------------------------------ */
00347
00348     /* Report timers... */
00349     STOP_TIMER(TIMER_TOTAL);
00350     PRINT_TIMER(TIMER_TOTAL);
00351     PRINT_TIMER(TIMER_INIT);
00352     PRINT_TIMER(TIMER_INPUT);
00353     PRINT_TIMER(TIMER_OUTPUT);
00354     PRINT_TIMER(TIMER_ADVECT);
00355     PRINT_TIMER(TIMER_DECAY);
00356     PRINT_TIMER(TIMER_DIFFMESO);
00357     PRINT_TIMER(TIMER_DIFFTURB);
00358     PRINT_TIMER(TIMER_ISOSURF);
00359     PRINT_TIMER(TIMER_METEO);
00360     PRINT_TIMER(TIMER_POSITION);
00361     PRINT_TIMER(TIMER_SEDI);
00362
00363     /* Report memory usage... */
00364     printf("MEMORY_ATM = %g MByte\n", 2. * sizeof(atm_t) / 1024. / 1024.);
00365     printf("MEMORY_METEO = %g MByte\n", 2. * sizeof(met_t) / 1024. / 1024.);
00366     printf("MEMORY_DYNAMIC = %g MByte\n",
00367            NP * sizeof(double) / 1024. / 1024.);
00368     printf("MEMORY_STATIC = %g MByte\n",
00369            (((EX + EY) + (2 + NQ) * GX * GY * GZ) * sizeof(double)
00370             + (EX * EY + EX * EY * EP) * sizeof(float)
00371             + (2 * GX * GY * GZ) * sizeof(int)) / 1024. / 1024.);
00372
```

```
00373     /* Report problem size... */
00374     printf("SIZE_NP = %d\n", atm->np);
00375     printf("SIZE_TASKS = %d\n", size);
00376     printf("SIZE_THREADS = %d\n", omp_get_max_threads());
00377
00378     /* Free random number generators... */
00379     for (i = 0; i < NTHREADS; i++)
00380       gsl_rng_free(rng[i]);
00381
00382     /* Free... */
00383     free(atm);
00384     free(met0);
00385     free(met1);
00386     free(dt);
00387   }
00388
00389 #ifdef MPI
00390   /* Finalize MPI... */
00391   MPI_Finalize();
00392 #endif
00393
00394   return EXIT_SUCCESS;
00395 }
00396
00397 /*****************************************************************************/
00398
00399 void init_simtime(
00400   ctl_t * ctl,
00401   atm_t * atm) {
00402
00403   /* Set inital and final time... */
00404   if (ctl->direction == 1) {
00405     if (ctl->t_start < -1e99)
00406       ctl->t_start = gsl_stats_min(atm->time, 1, (size_t) atm->np);
00407     if (ctl->t_stop < -1e99)
00408       ctl->t_stop = gsl_stats_max(atm->time, 1, (size_t) atm->np);
00409   } else if (ctl->direction == -1) {
00410     if (ctl->t_stop < -1e99)
00411       ctl->t_stop = gsl_stats_min(atm->time, 1, (size_t) atm->np);
00412     if (ctl->t_start < -1e99)
00413       ctl->t_start = gsl_stats_max(atm->time, 1, (size_t) atm->np);
00414   }
00415
00416   /* Check time... */
00417   if (ctl->direction * (ctl->t_stop - ctl->t_start) <= 0)
00418     ERRMSG("Nothing to do!");
00419 }
00420
00421 /*****************************************************************************/
00422
00423 void module_advection(
00424   met_t * met0,
00425   met_t * met1,
00426   atm_t * atm,
00427   int ip,
00428   double dt) {
00429
00430   double v[3], xm[3];
00431
00432   /* Interpolate meteorological data... */
00433   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00434                   atm->lon[ip], atm->lat[ip], NULL, NULL,
00435                   &v[0], &v[1], &v[2], NULL, NULL);
00436
00437   /* Get position of the mid point... */
00438   xm[0] = atm->lon[ip] + dx2deg(0.5 * dt * v[0] / 1000., atm->lat[ip]);
00439   xm[1] = atm->lat[ip] + dy2deg(0.5 * dt * v[1] / 1000.);
00440   xm[2] = atm->p[ip] + 0.5 * dt * v[2];
00441
00442   /* Interpolate meteorological data for mid point... */
00443   intpol_met_time(met0, met1, atm->time[ip] + 0.5 * dt,
00444                   xm[2], xm[0], xm[1], NULL, NULL,
00445                   &v[0], &v[1], &v[2], NULL, NULL);
00446
00447   /* Save new position... */
00448   atm->time[ip] += dt;
00449   atm->lon[ip] += dx2deg(dt * v[0] / 1000., xm[1]);
00450   atm->lat[ip] += dy2deg(dt * v[1] / 1000.);
00451   atm->p[ip] += dt * v[2];
00452 }
00453
00454 /*****************************************************************************/
00455
00456 void module_decay(
00457   ctl_t * ctl,
00458   met_t * met0,
00459   met_t * met1,
```

```
00460   atm_t * atm,
00461   int ip,
00462   double dt) {
00463
00464   double ps, pt, tdec;
00465
00466   /* Check lifetime values... */
00467   if ((ctl->tdec_trop <= 0 && ctl->tdec_strat <= 0) || ctl->
      qnt_m < 0)
00468     return;
00469
00470   /* Set constant lifetime... */
00471   if (ctl->tdec_trop == ctl->tdec_strat)
00472     tdec = ctl->tdec_trop;
00473
00474   /* Set altitude-dependent lifetime... */
00475   else {
00476
00477     /* Get surface pressure... */
00478     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00479                     atm->lon[ip], atm->lat[ip], &ps, NULL,
00480                     NULL, NULL, NULL, NULL, NULL);
00481
00482     /* Get tropopause pressure... */
00483     pt = tropopause(atm->time[ip], atm->lat[ip]);
00484
00485     /* Set lifetime... */
00486     if (atm->p[ip] <= pt)
00487       tdec = ctl->tdec_strat;
00488     else
00489       tdec = LIN(ps, ctl->tdec_trop, pt, ctl->tdec_strat, atm->
      p[ip]);
00490   }
00491
00492   /* Calculate exponential decay... */
00493   atm->q[ctl->qnt_m][ip] *= exp(-dt / tdec);
00494 }
00495
00496 /*****************************************************************************/
00497
00498 void module_diffusion_meso(
00499   ctl_t * ctl,
00500   met_t * met0,
00501   met_t * met1,
00502   atm_t * atm,
00503   int ip,
00504   double dt,
00505   gsl_rng * rng) {
00506
00507   double r, rs, u[16], v[16], w[16], usig, vsig, wsig;
00508
00509   int ix, iy, iz;
00510
00511   /* Calculate mesoscale velocity fluctuations... */
00512   if (ctl->turb_meso > 0) {
00513
00514     /* Get indices... */
00515     ix = locate(met0->lon, met0->nx, atm->lon[ip]);
00516     iy = locate(met0->lat, met0->ny, atm->lat[ip]);
00517     iz = locate(met0->p, met0->np, atm->p[ip]);
00518
00519     /* Collect local wind data... */
00520     u[0] = met0->u[ix][iy][iz];
00521     u[1] = met0->u[ix + 1][iy][iz];
00522     u[2] = met0->u[ix][iy + 1][iz];
00523     u[3] = met0->u[ix + 1][iy + 1][iz];
00524     u[4] = met0->u[ix][iy][iz + 1];
00525     u[5] = met0->u[ix + 1][iy][iz + 1];
00526     u[6] = met0->u[ix][iy + 1][iz + 1];
00527     u[7] = met0->u[ix + 1][iy + 1][iz + 1];
00528
00529     v[0] = met0->v[ix][iy][iz];
00530     v[1] = met0->v[ix + 1][iy][iz];
00531     v[2] = met0->v[ix][iy + 1][iz];
00532     v[3] = met0->v[ix + 1][iy + 1][iz];
00533     v[4] = met0->v[ix][iy][iz + 1];
00534     v[5] = met0->v[ix + 1][iy][iz + 1];
00535     v[6] = met0->v[ix][iy + 1][iz + 1];
00536     v[7] = met0->v[ix + 1][iy + 1][iz + 1];
00537
00538     w[0] = met0->w[ix][iy][iz];
00539     w[1] = met0->w[ix + 1][iy][iz];
00540     w[2] = met0->w[ix][iy + 1][iz];
00541     w[3] = met0->w[ix + 1][iy + 1][iz];
00542     w[4] = met0->w[ix][iy][iz + 1];
00543     w[5] = met0->w[ix + 1][iy][iz + 1];
00544     w[6] = met0->w[ix][iy + 1][iz + 1];
```

```
00545     w[7] = met0->w[ix + 1][iy + 1][iz + 1];
00546
00547     /* Get indices... */
00548     ix = locate(met1->lon, met1->nx, atm->lon[ip]);
00549     iy = locate(met1->lat, met1->ny, atm->lat[ip]);
00550     iz = locate(met1->p, met1->np, atm->p[ip]);
00551
00552     /* Collect local wind data... */
00553     u[8] = met1->u[ix][iy][iz];
00554     u[9] = met1->u[ix + 1][iy][iz];
00555     u[10] = met1->u[ix][iy + 1][iz];
00556     u[11] = met1->u[ix + 1][iy + 1][iz];
00557     u[12] = met1->u[ix][iy][iz + 1];
00558     u[13] = met1->u[ix + 1][iy][iz + 1];
00559     u[14] = met1->u[ix][iy + 1][iz + 1];
00560     u[15] = met1->u[ix + 1][iy + 1][iz + 1];
00561
00562     v[8] = met1->v[ix][iy][iz];
00563     v[9] = met1->v[ix + 1][iy][iz];
00564     v[10] = met1->v[ix][iy + 1][iz];
00565     v[11] = met1->v[ix + 1][iy + 1][iz];
00566     v[12] = met1->v[ix][iy][iz + 1];
00567     v[13] = met1->v[ix + 1][iy][iz + 1];
00568     v[14] = met1->v[ix][iy + 1][iz + 1];
00569     v[15] = met1->v[ix + 1][iy + 1][iz + 1];
00570
00571     w[8] = met1->w[ix][iy][iz];
00572     w[9] = met1->w[ix + 1][iy][iz];
00573     w[10] = met1->w[ix][iy + 1][iz];
00574     w[11] = met1->w[ix + 1][iy + 1][iz];
00575     w[12] = met1->w[ix][iy][iz + 1];
00576     w[13] = met1->w[ix + 1][iy][iz + 1];
00577     w[14] = met1->w[ix][iy + 1][iz + 1];
00578     w[15] = met1->w[ix + 1][iy + 1][iz + 1];
00579
00580     /* Get standard deviations of local wind data... */
00581     usig = gsl_stats_sd(u, 1, 16);
00582     vsig = gsl_stats_sd(v, 1, 16);
00583     wsig = gsl_stats_sd(w, 1, 16);
00584
00585     /* Set temporal correlations for mesoscale fluctuations... */
00586     r = 1 - 2 * fabs(dt) / ctl->dt_met;
00587     rs = sqrt(1 - r * r);
00588
00589     /* Calculate mesoscale wind fluctuations... */
00590     atm->up[ip] =
00591       r * atm->up[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00592                                                        ctl->turb_meso * usig);
00593     atm->vp[ip] =
00594       r * atm->vp[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00595                                                        ctl->turb_meso * vsig);
00596     atm->wp[ip] =
00597       r * atm->wp[ip] + rs * gsl_ran_gaussian_ziggurat(rng,
00598                                                        ctl->turb_meso * wsig);
00599
00600     /* Calculate air parcel displacement... */
00601     atm->lon[ip] += dx2deg(atm->up[ip] * dt / 1000., atm->lat[ip]);
00602     atm->lat[ip] += dy2deg(atm->vp[ip] * dt / 1000.);
00603     atm->p[ip] += atm->wp[ip] * dt;
00604   }
00605 }
00606
00607 /*****************************************************************************/
00608
00609 void module_diffusion_turb(
00610   ctl_t * ctl,
00611   atm_t * atm,
00612   int ip,
00613   double dt,
00614   gsl_rng * rng) {
00615
00616   double dx, dz, pt, p0, p1, w;
00617
00618   /* Get tropopause pressure... */
00619   pt = tropopause(atm->time[ip], atm->lat[ip]);
00620
00621   /* Get weighting factor... */
00622   p1 = pt * 0.866877899;
00623   p0 = pt / 0.866877899;
00624   if (atm->p[ip] > p0)
00625     w = 1;
00626   else if (atm->p[ip] < p1)
00627     w = 0;
00628   else
00629     w = LIN(p0, 1.0, p1, 0.0, atm->p[ip]);
00630
00631   /* Set diffusivitiy... */
```

```
00632   dx = w * ctl->turb_dx_trop + (1 - w) * ctl->turb_dx_strat;
00633   dz = w * ctl->turb_dz_trop + (1 - w) * ctl->turb_dz_strat;
00634
00635   /* Horizontal turbulent diffusion... */
00636   if (dx > 0) {
00637     atm->lon[ip]
00638       += dx2deg(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dx * fabs(dt)))
00639                 / 1000., atm->lat[ip]);
00640     atm->lat[ip]
00641       += dy2deg(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dx * fabs(dt)))
00642                 / 1000.);
00643   }
00644
00645   /* Vertical turbulent diffusion... */
00646   if (dz > 0)
00647     atm->p[ip]
00648       += dz2dp(gsl_ran_gaussian_ziggurat(rng, sqrt(2.0 * dz * fabs(dt)))
00649                / 1000., atm->p[ip]);
00650 }
00651
00652 /*****************************************************************************/
00653
00654 void module_isosurf(
00655   ctl_t * ctl,
00656   met_t * met0,
00657   met_t * met1,
00658   atm_t * atm,
00659   int ip) {
00660
00661   static double *iso, *ps, t, *ts;
00662
00663   static int idx, ip2, n, nb = 100000;
00664
00665   FILE *in;
00666
00667   char line[LEN];
00668
00669   /* Check control parameter... */
00670   if (ctl->isosurf < 1 || ctl->isosurf > 4)
00671     return;
00672
00673   /* Initialize... */
00674   if (ip < 0) {
00675
00676     /* Allocate... */
00677     ALLOC(iso, double,
00678           NP);
00679     ALLOC(ps, double,
00680           nb);
00681     ALLOC(ts, double,
00682           nb);
00683
00684     /* Save pressure... */
00685     if (ctl->isosurf == 1)
00686       for (ip2 = 0; ip2 < atm->np; ip2++)
00687         iso[ip2] = atm->p[ip2];
00688
00689     /* Save density... */
00690     else if (ctl->isosurf == 2)
00691       for (ip2 = 0; ip2 < atm->np; ip2++) {
00692         intpol_met_time(met0, met1, atm->time[ip2], atm->p[ip2],
00693                         atm->lon[ip2], atm->lat[ip2], NULL, &t, NULL, NULL,
00694                         NULL, NULL, NULL);
00695         iso[ip2] = atm->p[ip2] / t;
00696       }
00697
00698     /* Save potential temperature... */
00699     else if (ctl->isosurf == 3)
00700       for (ip2 = 0; ip2 < atm->np; ip2++) {
00701         intpol_met_time(met0, met1, atm->time[ip2], atm->p[ip2],
00702                         atm->lon[ip2], atm->lat[ip2], NULL, &t, NULL, NULL,
00703                         NULL, NULL, NULL);
00704         iso[ip2] = t * pow(P0 / atm->p[ip2], 0.286);
00705       }
00706
00707     /* Read balloon pressure data... */
00708     else if (ctl->isosurf == 4) {
00709
00710       /* Write info... */
00711       printf("Read balloon pressure data: %s\n", ctl->balloon);
00712
00713       /* Open file... */
00714       if (!(in = fopen(ctl->balloon, "r")))
00715         ERRMSG("Cannot open file!");
00716
00717       /* Read pressure time series... */
00718       while (fgets(line, LEN, in))
```

```
00719         if (sscanf(line, "%lg %lg", &ts[n], &ps[n]) == 2)
00720           if ((++n) > 100000)
00721             ERRMSG("Too many data points!");
00722
00723       /* Check number of points... */
00724       if (n < 1)
00725         ERRMSG("Could not read any data!");
00726
00727       /* Close file... */
00728       fclose(in);
00729     }
00730
00731     /* Leave initialization... */
00732     return;
00733   }
00734
00735   /* Restore pressure... */
00736   if (ctl->isosurf == 1)
00737     atm->p[ip] = iso[ip];
00738
00739   /* Restore density... */
00740   else if (ctl->isosurf == 2) {
00741     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
    lon[ip],
00742                     atm->lat[ip], NULL, &t, NULL, NULL, NULL, NULL, NULL);
00743     atm->p[ip] = iso[ip] * t;
00744   }
00745
00746   /* Restore potential temperature... */
00747   else if (ctl->isosurf == 3) {
00748     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
    lon[ip],
00749                     atm->lat[ip], NULL, &t, NULL, NULL, NULL, NULL, NULL);
00750     atm->p[ip] = P0 * pow(iso[ip] / t, -1. / 0.286);
00751   }
00752
00753   /* Interpolate pressure... */
00754   else if (ctl->isosurf == 4) {
00755     if (atm->time[ip] <= ts[0])
00756       atm->p[ip] = ps[0];
00757     else if (atm->time[ip] >= ts[n - 1])
00758       atm->p[ip] = ps[n - 1];
00759     else {
00760       idx = locate(ts, n, atm->time[ip]);
00761       atm->p[ip] = LIN(ts[idx], ps[idx],
00762                        ts[idx + 1], ps[idx + 1], atm->time[ip]);
00763     }
00764   }
00765 }
00766
00767 /*****************************************************************************/
00768
00769 void module_meteo(
00770   ctl_t * ctl,
00771   met_t * met0,
00772   met_t * met1,
00773   atm_t * atm,
00774   int ip) {
00775
00776 #include "topo.c"
00777
00778   static FILE *in;
00779
00780   static char filename[LEN], line[LEN];
00781
00782   static double lon[GX], lat[GY], var[GX][GY],
00783     rdum, rlat, rlat_old = -999, rlon, rvar;
00784
00785   static int year_old, mon_old, day_old, nlon, nlat;
00786
00787   double b, c, ps, p1, p_hno3, p_h2o, t, t1, term1, term2,
00788     u, u1, v, v1, w, x1, x2, h2o, o3, grad, vort, var0, var1;
00789
00790   int day, mon, year, idum, ilat, ilon;
00791
00792   /* Interpolate meteorological data... */
00793   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
    lon[ip],
00794                   atm->lat[ip], &ps, &t, &u, &v, &w, &h2o, &o3);
00795
00796   /* Set surface pressure... */
00797   if (ctl->qnt_ps >= 0)
00798     atm->q[ctl->qnt_ps][ip] = ps;
00799
00800   /* Set pressure... */
00801   if (ctl->qnt_p >= 0)
00802     atm->q[ctl->qnt_p][ip] = atm->p[ip];
```

```
00803
00804   /* Set temperature... */
00805   if (ctl->qnt_t >= 0)
00806     atm->q[ctl->qnt_t][ip] = t;
00807
00808   /* Set zonal wind... */
00809   if (ctl->qnt_u >= 0)
00810     atm->q[ctl->qnt_u][ip] = u;
00811
00812   /* Set meridional wind... */
00813   if (ctl->qnt_v >= 0)
00814     atm->q[ctl->qnt_v][ip] = v;
00815
00816   /* Set vertical velocity... */
00817   if (ctl->qnt_w >= 0)
00818     atm->q[ctl->qnt_w][ip] = w;
00819
00820   /* Set water vapor vmr... */
00821   if (ctl->qnt_h2o >= 0)
00822     atm->q[ctl->qnt_h2o][ip] = h2o;
00823
00824   /* Set ozone vmr... */
00825   if (ctl->qnt_o3 >= 0)
00826     atm->q[ctl->qnt_o3][ip] = o3;
00827
00828   /* Calculate potential temperature... */
00829   if (ctl->qnt_theta >= 0)
00830     atm->q[ctl->qnt_theta][ip] = t * pow(P0 / atm->p[ip], 0.286);
00831
00832   /* Calculate potential vorticity... */
00833   if (ctl->qnt_pv >= 0) {
00834
00835     /* Absolute vorticity... */
00836     vort = 2 * 7.2921e-5 * sin(atm->lat[ip] * M_PI / 180.);
00837     if (fabs(atm->lat[ip]) < 89.) {
00838       intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00839                       (atm->lon[ip] >=
00840                        0 ? atm->lon[ip] - 1. : atm->lon[ip] + 1.),
00841                       atm->lat[ip], NULL, NULL, NULL, &v1, NULL, NULL, NULL);
00842       vort += (v1 - v) / 1000.
00843         / ((atm->lon[ip] >= 0 ? -1 : 1) * deg2dx(1., atm->lat[ip]));
00844     }
00845     intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
00846                     (atm->lat[ip] >=
00847                      0 ? atm->lat[ip] - 1. : atm->lat[ip] + 1.), NULL, NULL,
00848                     &u1, NULL, NULL, NULL, NULL);
00849     vort += (u1 - u) / 1000. / ((atm->lat[ip] >= 0 ? -1 : 1) * deg2dy(1.));
00850
00851     /* Potential temperature gradient... */
00852     p1 = 0.85 * atm->p[ip];
00853     intpol_met_time(met0, met1, atm->time[ip], p1, atm->lon[ip],
00854                     atm->lat[ip], NULL, &t1, NULL, NULL, NULL, NULL, NULL);
00855     grad = (t1 * pow(P0 / p1, 0.286) - t * pow(P0 / atm->p[ip], 0.286))
00856       / (100. * (p1 - atm->p[ip]));
00857
00858     /* Calculate PV... */
00859     atm->q[ctl->qnt_pv][ip] = -1e6 * G0 * vort * grad;
00860   }
00861
00862   /* Calculate T_ice (Marti and Mauersberger, 1993)... */
00863   if (ctl->qnt_tice >= 0 || ctl->qnt_tsts >= 0)
00864     atm->q[ctl->qnt_tice][ip] = -2663.5
00865       / (log10(ctl->psc_h2o * atm->p[ip] * 100.) - 12.537);
00866
00867   /* Calculate T_NAT (Hanson and Mauersberger, 1988)... */
00868   if (ctl->qnt_tnat >= 0 || ctl->qnt_tsts >= 0) {
00869     p_hno3 = ctl->psc_hno3 * atm->p[ip] / 1.333224;
00870     p_h2o = ctl->psc_h2o * atm->p[ip] / 1.333224;
00871     term1 = 38.9855 - log10(p_hno3) - 2.7836 * log10(p_h2o);
00872     term2 = 0.009179 - 0.00088 * log10(p_h2o);
00873     b = term1 / term2;
00874     c = -11397.0 / term2;
00875     x1 = (-b + sqrt(b * b - 4. * c)) / 2.;
00876     x2 = (-b - sqrt(b * b - 4. * c)) / 2.;
00877     if (x1 > 0)
00878       atm->q[ctl->qnt_tnat][ip] = x1;
00879     if (x2 > 0)
00880       atm->q[ctl->qnt_tnat][ip] = x2;
00881   }
00882
00883   /* Calculate T_STS (mean of T_ice and T_NAT)... */
00884   if (ctl->qnt_tsts >= 0) {
00885     if (ctl->qnt_tice < 0 || ctl->qnt_tnat < 0)
00886       ERRMSG("Need T_ice and T_NAT to calculate T_STS!");
00887     atm->q[ctl->qnt_tsts][ip] = 0.5 * (atm->q[ctl->qnt_tice][ip]
00888                                        + atm->q[ctl->qnt_tnat][ip]);
```

```
00889   }
00890
00891   /* Interpolate low-level (750 hPa) wind... */
00892   if (ctl->qnt_gw_wind >= 0)
00893     intpol_met_time(met0, met1, atm->time[ip], 750., atm->lon[ip],
00894                     atm->lat[ip], NULL, NULL, &atm->q[ctl->qnt_gw_wind][ip],
00895                     NULL, NULL, NULL, NULL);
00896
00897   /* Interpolate terrain height variance... */
00898   if (ctl->qnt_gw_sso >= 0) {
00899     ilat = locate(topo_lat, TOPO_NLAT, atm->lat[ip]);
00900     ilon = locate(topo_lon, TOPO_NLON, atm->lon[ip]);
00901     var0 = LIN(topo_lat[ilat], topo_zvar[ilon][ilat],
00902                topo_lat[ilat + 1], topo_zvar[ilon][ilat + 1], atm->lat[ip]);
00903     var1 = LIN(topo_lat[ilat], topo_zvar[ilon + 1][ilat],
00904                topo_lat[ilat + 1], topo_zvar[ilon + 1][ilat + 1],
00905                atm->lat[ip]);
00906     atm->q[ctl->qnt_gw_sso][ip]
00907       = LIN(topo_lon[ilon], var0, topo_lon[ilon + 1], var1, atm->lon[ip]);
00908   }
00909
00910   /* Read variance data for current day... */
00911   if (ip == 0 && ctl->qnt_gw_var >= 0) {
00912     jsec2time(atm->time[ip], &year, &mon, &day, &idum, &idum, &idum, &rdum);
00913     if (year != year_old || mon != mon_old || day != day_old) {
00914       year_old = year;
00915       mon_old = mon;
00916       day_old = day;
00917       nlon = nlat = -1;
00918       sprintf(filename, "%s_%d_%02d_%02d.tab",
00919               ctl->gw_basename, year, mon, day);
00920       if ((in = fopen(filename, "r"))) {
00921         printf("Read gravity wave data: %s\n", filename);
00922         while (fgets(line, LEN, in)) {
00923           if (sscanf(line, "%lg %lg %lg", &rlon, &rlat, &rvar) != 3)
00924             continue;
00925           if (rlat != rlat_old) {
00926             rlat_old = rlat;
00927             if ((++nlat) > GY)
00928               ERRMSG("Too many latitudes!");
00929             nlon = -1;
00930           }
00931           if ((++nlon) > GX)
00932             ERRMSG("Too many longitudes!");
00933           lon[nlon] = rlon;
00934           lat[nlat] = rlat;
00935           var[nlon][nlat] = GSL_MAX(0, rvar);
00936         }
00937         fclose(in);
00938         nlat++;
00939         nlon++;
00940       } else
00941         printf("Missing gravity wave data: %s\n", filename);
00942     }
00943   }
00944
00945   /* Interpolate variance data... */
00946   if (ctl->qnt_gw_var >= 0) {
00947     if (nlat >= 2 && nlon >= 2) {
00948       ilat = locate(lat, nlat, atm->lat[ip]);
00949       ilon = locate(lon, nlon, atm->lon[ip]);
00950       var0 = LIN(lat[ilat], var[ilon][ilat],
00951                  lat[ilat + 1], var[ilon][ilat + 1], atm->lat[ip]);
00952       var1 = LIN(lat[ilat], var[ilon + 1][ilat],
00953                  lat[ilat + 1], var[ilon + 1][ilat + 1], atm->lat[ip]);
00954       atm->q[ctl->qnt_gw_var][ip]
00955         = LIN(lon[ilon], var0, lon[ilon + 1], var1, atm->lon[ip]);
00956     } else
00957       atm->q[ctl->qnt_gw_var][ip] = GSL_NAN;
00958   }
00959 }
00960
00961 /*****************************************************************************/
00962
00963 void module_position(
00964   met_t * met0,
00965   met_t * met1,
00966   atm_t * atm,
00967   int ip) {
00968
00969   double ps;
00970
00971   /* Calculate modulo... */
00972   atm->lon[ip] = fmod(atm->lon[ip], 360);
00973   atm->lat[ip] = fmod(atm->lat[ip], 360);
00974
00975   /* Check latitude... */
```

```
00976   while (atm->lat[ip] < -90 || atm->lat[ip] > 90) {
00977     if (atm->lat[ip] > 90) {
00978       atm->lat[ip] = 180 - atm->lat[ip];
00979       atm->lon[ip] += 180;
00980     }
00981     if (atm->lat[ip] < -90) {
00982       atm->lat[ip] = -180 - atm->lat[ip];
00983       atm->lon[ip] += 180;
00984     }
00985   }
00986
00987   /* Check longitude... */
00988   while (atm->lon[ip] < -180)
00989     atm->lon[ip] += 360;
00990   while (atm->lon[ip] >= 180)
00991     atm->lon[ip] -= 360;
00992
00993   /* Get surface pressure... */
00994   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip],
00995                   atm->lon[ip], atm->lat[ip], &ps, NULL,
00996                   NULL, NULL, NULL, NULL, NULL);
00997
00998   /* Check pressure... */
00999   if (atm->p[ip] > ps)
01000     atm->p[ip] = ps;
01001   else if (atm->p[ip] < met0->p[met0->np - 1])
01002     atm->p[ip] = met0->p[met0->np - 1];
01003 }
01004
01005 /*****************************************************************************/
01006
01007 void module_sedi(
01008   ctl_t * ctl,
01009   met_t * met0,
01010   met_t * met1,
01011   atm_t * atm,
01012   int ip,
01013   double dt) {
01014
01015   /* Coefficients for Cunningham slip-flow correction (Kasten, 1968): */
01016   const double A = 1.249, B = 0.42, C = 0.87;
01017
01018   /* Specific gas constant for dry air [J/(kg K)]: */
01019   const double R = 287.058;
01020
01021   /* Average mass of an air molecule [kg/molec]: */
01022   const double m = 4.8096e-26;
01023
01024   double G, K, eta, lambda, p, r_p, rho, rho_p, T, v, v_p;
01025
01026   /* Check if parameters are available... */
01027   if (ctl->qnt_r < 0 || ctl->qnt_rho < 0)
01028     return;
01029
01030   /* Convert units... */
01031   p = 100 * atm->p[ip];
01032   r_p = 1e-6 * atm->q[ctl->qnt_r][ip];
01033   rho_p = atm->q[ctl->qnt_rho][ip];
01034
01035   /* Get temperature... */
01036   intpol_met_time(met0, met1, atm->time[ip], atm->p[ip], atm->
      lon[ip],
01037                   atm->lat[ip], NULL, &T, NULL, NULL, NULL, NULL, NULL);
01038
01039   /* Density of dry air... */
01040   rho = p / (R * T);
01041
01042   /* Dynamic viscosity of air... */
01043   eta = 1.8325e-5 * (416.16 / (T + 120.)) * pow(T / 296.16, 1.5);
01044
01045   /* Thermal velocity of an air molecule... */
01046   v = sqrt(8 * GSL_CONST_MKSA_BOLTZMANN * T / (M_PI * m));
01047
01048   /* Mean free path of an air molecule... */
01049   lambda = 2 * eta / (rho * v);
01050
01051   /* Knudsen number for air... */
01052   K = lambda / r_p;
01053
01054   /* Cunningham slip-flow correction... */
01055   G = 1 + K * (A + B * exp(-C / K));
01056
01057   /* Sedimentation (fall) velocity... */
01058   v_p =
01059     2. * gsl_pow_2(r_p) * (rho_p -
01060                            rho) * GSL_CONST_MKSA_GRAV_ACCEL / (9. * eta) * G;
01061
```

```
01062   /* Calculate pressure change... */
01063   atm->p[ip] += dz2dp(v_p * dt / 1000., atm->p[ip]);
01064 }
01065
01066 /*****************************************************************************/
01067
01068 void write_output(
01069   const char *dirname,
01070   ctl_t * ctl,
01071   met_t * met0,
01072   met_t * met1,
01073   atm_t * atm,
01074   double t) {
01075
01076   char filename[LEN];
01077
01078   double r;
01079
01080   int year, mon, day, hour, min, sec;
01081
01082   /* Get time... */
01083   jsec2time(t, &year, &mon, &day, &hour, &min, &sec, &r);
01084
01085   /* Write atmospheric data... */
01086   if (ctl->atm_basename[0] != '-' && fmod(t, ctl->atm_dt_out) == 0) {
01087     sprintf(filename, "%s/%s_%04d_%02d_%02d_%02d_%02d.tab",
01088             dirname, ctl->atm_basename, year, mon, day, hour, min);
01089     write_atm(filename, ctl, atm, t);
01090   }
01091
01092   /* Write CSI data... */
01093   if (ctl->csi_basename[0] != '-') {
01094     sprintf(filename, "%s/%s.tab", dirname, ctl->csi_basename);
01095     write_csi(filename, ctl, atm, t);
01096   }
01097
01098   /* Write ensemble data... */
01099   if (ctl->ens_basename[0] != '-') {
01100     sprintf(filename, "%s/%s.tab", dirname, ctl->ens_basename);
01101     write_ens(filename, ctl, atm, t);
01102   }
01103
01104   /* Write gridded data... */
01105   if (ctl->grid_basename[0] != '-' && fmod(t, ctl->grid_dt_out) == 0) {
01106     sprintf(filename, "%s/%s_%04d_%02d_%02d_%02d_%02d.tab",
01107             dirname, ctl->grid_basename, year, mon, day, hour, min);
01108     write_grid(filename, ctl, met0, met1, atm, t);
01109   }
01110
01111   /* Write profile data... */
01112   if (ctl->prof_basename[0] != '-') {
01113     sprintf(filename, "%s/%s.tab", dirname, ctl->prof_basename);
01114     write_prof(filename, ctl, met0, met1, atm, t);
01115   }
01116
01117   /* Write station data... */
01118   if (ctl->stat_basename[0] != '-') {
01119     sprintf(filename, "%s/%s.tab", dirname, ctl->stat_basename);
01120     write_station(filename, ctl, atm, t);
01121   }
01122 }
```

5.35 wind.c File Reference

Create meteorological data files with synthetic wind fields.

Functions

- void add_text_attribute (int ncid, char *varname, char *attrname, char *text)
- int main (int argc, char *argv[])

5.35.1 Detailed Description

Create meteorological data files with synthetic wind fields.

Definition in file wind.c.

5.35.2 Function Documentation

5.35.2.1 void add_text_attribute (int *ncid*, char * *varname*, char * *attrname*, char * *text*)

Definition at line 173 of file wind.c.

```
00177               {
00178
00179   int varid;
00180
00181   NC(nc_inq_varid(ncid, varname, &varid));
00182   NC(nc_put_att_text(ncid, varid, attrname, strlen(text), text));
00183 }
```

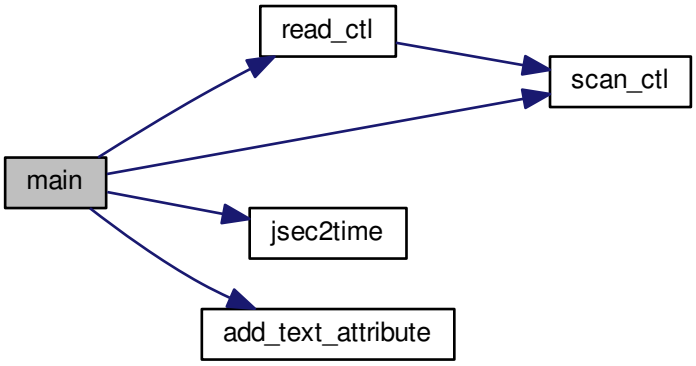
5.35.2.2 int main (int *argc*, char * *argv[ ]*)

Definition at line 41 of file wind.c.

```
00043                {
00044
00045   ctl_t ctl;
00046
00047   static char filename[LEN];
00048
00049   static double r, t0, z0, z1, dataLon[EX], dataLat[EY], dataZ[EP],
00050     u0, u1, alpha;
00051
00052   static float dataT[EP * EY * EX], dataU[EP * EY * EX], dataV[EP * EY * EX],
00053     dataW[EP * EY * EX];
00054
00055   static int ncid, dims[4], timid, levid, latid, lonid, tid, uid, vid, wid,
00056     idx, ix, iy, iz, nx, ny, nz, year, mon, day, hour, min, sec;
00057
00058   /* Check arguments... */
00059   if (argc < 3)
00060     ERRMSG("Give parameters: <ctl> <metbase>");
00061
00062   /* Read control parameters... */
00063   read_ctl(argv[1], argc, argv, &ctl);
00064   t0 = scan_ctl(argv[1], argc, argv, "WIND_T0", -1, "0", NULL);
00065   nx = (int) scan_ctl(argv[1], argc, argv, "WIND_NX", -1, "360", NULL);
00066   ny = (int) scan_ctl(argv[1], argc, argv, "WIND_NY", -1, "181", NULL);
00067   nz = (int) scan_ctl(argv[1], argc, argv, "WIND_NZ", -1, "61", NULL);
00068   z0 = scan_ctl(argv[1], argc, argv, "WIND_Z0", -1, "0", NULL);
00069   z1 = scan_ctl(argv[1], argc, argv, "WIND_Z1", -1, "60", NULL);
00070   u0 = scan_ctl(argv[1], argc, argv, "WIND_U0", -1, "38.587660177302", NULL);
00071   u1 = scan_ctl(argv[1], argc, argv, "WIND_U1", -1, "38.587660177302", NULL);
00072   alpha = scan_ctl(argv[1], argc, argv, "WIND_ALPHA", -1, "0.0", NULL);
00073
00074   /* Check dimensions... */
00075   if (nx < 1 || nx > EX)
00076     ERRMSG("Set 1 <= NX <= MAX!");
00077   if (ny < 1 || ny > EY)
00078     ERRMSG("Set 1 <= NY <= MAX!");
00079   if (nz < 1 || nz > EP)
00080     ERRMSG("Set 1 <= NZ <= MAX!");
00081
00082   /* Get time... */
00083   jsec2time(t0, &year, &mon, &day, &hour, &min, &sec, &r);
00084   t0 = year * 10000. + mon * 100. + day + hour / 24.;
00085
00086   /* Set filename... */
00087   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", argv[2], year, mon, day, hour);
00088
00089   /* Create netCDF file... */
00090   NC(nc_create(filename, NC_CLOBBER, &ncid));
00091
00092   /* Create dimensions... */
00093   NC(nc_def_dim(ncid, "time", 1, &dims[0]));
00094   NC(nc_def_dim(ncid, "lev", (size_t) nz, &dims[1]));
00095   NC(nc_def_dim(ncid, "lat", (size_t) ny, &dims[2]));
00096   NC(nc_def_dim(ncid, "lon", (size_t) nx, &dims[3]));
00097
00098   /* Create variables... */
00099   NC(nc_def_var(ncid, "time", NC_DOUBLE, 1, &dims[0], &timid));
00100   NC(nc_def_var(ncid, "lev", NC_DOUBLE, 1, &dims[1], &levid));
```

```
00101   NC(nc_def_var(ncid, "lat", NC_DOUBLE, 1, &dims[2], &latid));
00102   NC(nc_def_var(ncid, "lon", NC_DOUBLE, 1, &dims[3], &lonid));
00103   NC(nc_def_var(ncid, "T", NC_FLOAT, 4, &dims[0], &tid));
00104   NC(nc_def_var(ncid, "U", NC_FLOAT, 4, &dims[0], &uid));
00105   NC(nc_def_var(ncid, "V", NC_FLOAT, 4, &dims[0], &vid));
00106   NC(nc_def_var(ncid, "W", NC_FLOAT, 4, &dims[0], &wid));
00107
00108   /* Set attributes... */
00109   add_text_attribute(ncid, "time", "long_name", "time");
00110   add_text_attribute(ncid, "time", "units", "day as %Y%m%d.%f");
00111   add_text_attribute(ncid, "lon", "long_name", "longitude");
00112   add_text_attribute(ncid, "lon", "units", "degrees_east");
00113   add_text_attribute(ncid, "lat", "long_name", "latitude");
00114   add_text_attribute(ncid, "lat", "units", "degrees_north");
00115   add_text_attribute(ncid, "lev", "long_name", "air_pressure");
00116   add_text_attribute(ncid, "lev", "units", "Pa");
00117   add_text_attribute(ncid, "T", "long_name", "Temperature");
00118   add_text_attribute(ncid, "T", "units", "K");
00119   add_text_attribute(ncid, "U", "long_name", "U velocity");
00120   add_text_attribute(ncid, "U", "units", "m s**-1");
00121   add_text_attribute(ncid, "V", "long_name", "V velocity");
00122   add_text_attribute(ncid, "V", "units", "m s**-1");
00123   add_text_attribute(ncid, "W", "long_name", "Vertical velocity");
00124   add_text_attribute(ncid, "W", "units", "Pa s**-1");
00125
00126   /* End definition... */
00127   NC(nc_enddef(ncid));
00128
00129   /* Set coordinates... */
00130   for (ix = 0; ix < nx; ix++)
00131     dataLon[ix] = 360.0 / nx * (double) ix;
00132   for (iy = 0; iy < ny; iy++)
00133     dataLat[iy] = 180.0 / (ny - 1) * (double) iy - 90;
00134   for (iz = 0; iz < nz; iz++)
00135     dataZ[iz] = 100. * P(LIN(0.0, z0, nz - 1.0, z1, iz));
00136
00137   /* Write coordinates... */
00138   NC(nc_put_var_double(ncid, timid, &t0));
00139   NC(nc_put_var_double(ncid, levid, dataZ));
00140   NC(nc_put_var_double(ncid, lonid, dataLon));
00141   NC(nc_put_var_double(ncid, latid, dataLat));
00142
00143   /* Create wind fields (Williamson et al., 1992)... */
00144   for (ix = 0; ix < nx; ix++)
00145     for (iy = 0; iy < ny; iy++)
00146       for (iz = 0; iz < nz; iz++) {
00147         idx = (iz * ny + iy) * nx + ix;
00148         dataU[idx] = (float) (LIN(0.0, u0, nz - 1.0, u1, iz)
00149                               * (cos(dataLat[iy] * M_PI / 180.0)
00150                                  * cos(alpha * M_PI / 180.0)
00151                                  + sin(dataLat[iy] * M_PI / 180.0)
00152                                  * cos(dataLon[ix] * M_PI / 180.0)
00153                                  * sin(alpha * M_PI / 180.0)));
00154         dataV[idx] = (float) (-LIN(0.0, u0, nz - 1.0, u1, iz)
00155                               * sin(dataLon[ix] * M_PI / 180.0)
00156                               * sin(alpha * M_PI / 180.0));
00157       }
00158
00159   /* Write wind data... */
00160   NC(nc_put_var_float(ncid, tid, dataT));
00161   NC(nc_put_var_float(ncid, uid, dataU));
00162   NC(nc_put_var_float(ncid, vid, dataV));
00163   NC(nc_put_var_float(ncid, wid, dataW));
00164
00165   /* Close file... */
00166   NC(nc_close(ncid));
00167
00168   return EXIT_SUCCESS;
00169 }
```

Here is the call graph for this function:



5.36 wind.c

```
00001 /*
00002   This file is part of MPTRAC.
00003
00004   MPTRAC 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   MPTRAC 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 MPTRAC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copright (C) 2013-2015 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "libtrac.h"
00026
00027 /* ------------------------------------------------------------
00028    Functions...
00029    ------------------------------------------------------------ */
00030
00031 void add_text_attribute(
00032   int ncid,
00033   char *varname,
00034   char *attrname,
00035   char *text);
00036
00037 /* ------------------------------------------------------------
00038    Main...
00039    ------------------------------------------------------------ */
00040
00041 int main(
00042   int argc,
00043   char *argv[]) {
00044
00045   ctl_t ctl;
00046
00047   static char filename[LEN];
00048
00049   static double r, t0, z0, z1, dataLon[EX], dataLat[EY], dataZ[EP],
00050     u0, u1, alpha;
00051
00052   static float dataT[EP * EY * EX], dataU[EP * EY * EX], dataV[EP * EY * EX],
00053     dataW[EP * EY * EX];
00054
```

```
00055   static int ncid, dims[4], timid, levid, latid, lonid, tid, uid, vid, wid,
00056     idx, ix, iy, iz, nx, ny, nz, year, mon, day, hour, min, sec;
00057
00058   /* Check arguments... */
00059   if (argc < 3)
00060     ERRMSG("Give parameters: <ctl> <metbase>");
00061
00062   /* Read control parameters... */
00063   read_ctl(argv[1], argc, argv, &ctl);
00064   t0 = scan_ctl(argv[1], argc, argv, "WIND_T0", -1, "0", NULL);
00065   nx = (int) scan_ctl(argv[1], argc, argv, "WIND_NX", -1, "360", NULL);
00066   ny = (int) scan_ctl(argv[1], argc, argv, "WIND_NY", -1, "181", NULL);
00067   nz = (int) scan_ctl(argv[1], argc, argv, "WIND_NZ", -1, "61", NULL);
00068   z0 = scan_ctl(argv[1], argc, argv, "WIND_Z0", -1, "0", NULL);
00069   z1 = scan_ctl(argv[1], argc, argv, "WIND_Z1", -1, "60", NULL);
00070   u0 = scan_ctl(argv[1], argc, argv, "WIND_U0", -1, "38.587660177302", NULL);
00071   u1 = scan_ctl(argv[1], argc, argv, "WIND_U1", -1, "38.587660177302", NULL);
00072   alpha = scan_ctl(argv[1], argc, argv, "WIND_ALPHA", -1, "0.0", NULL);
00073
00074   /* Check dimensions... */
00075   if (nx < 1 || nx > EX)
00076     ERRMSG("Set 1 <= NX <= MAX!");
00077   if (ny < 1 || ny > EY)
00078     ERRMSG("Set 1 <= NY <= MAX!");
00079   if (nz < 1 || nz > EP)
00080     ERRMSG("Set 1 <= NZ <= MAX!");
00081
00082   /* Get time... */
00083   jsec2time(t0, &year, &mon, &day, &hour, &min, &sec, &r);
00084   t0 = year * 10000. + mon * 100. + day + hour / 24.;
00085
00086   /* Set filename... */
00087   sprintf(filename, "%s_%d_%02d_%02d_%02d.nc", argv[2], year, mon, day, hour);
00088
00089   /* Create netCDF file... */
00090   NC(nc_create(filename, NC_CLOBBER, &ncid));
00091
00092   /* Create dimensions... */
00093   NC(nc_def_dim(ncid, "time", 1, &dims[0]));
00094   NC(nc_def_dim(ncid, "lev", (size_t) nz, &dims[1]));
00095   NC(nc_def_dim(ncid, "lat", (size_t) ny, &dims[2]));
00096   NC(nc_def_dim(ncid, "lon", (size_t) nx, &dims[3]));
00097
00098   /* Create variables... */
00099   NC(nc_def_var(ncid, "time", NC_DOUBLE, 1, &dims[0], &timid));
00100   NC(nc_def_var(ncid, "lev", NC_DOUBLE, 1, &dims[1], &levid));
00101   NC(nc_def_var(ncid, "lat", NC_DOUBLE, 1, &dims[2], &latid));
00102   NC(nc_def_var(ncid, "lon", NC_DOUBLE, 1, &dims[3], &lonid));
00103   NC(nc_def_var(ncid, "T", NC_FLOAT, 4, &dims[0], &tid));
00104   NC(nc_def_var(ncid, "U", NC_FLOAT, 4, &dims[0], &uid));
00105   NC(nc_def_var(ncid, "V", NC_FLOAT, 4, &dims[0], &vid));
00106   NC(nc_def_var(ncid, "W", NC_FLOAT, 4, &dims[0], &wid));
00107
00108   /* Set attributes... */
00109   add_text_attribute(ncid, "time", "long_name", "time");
00110   add_text_attribute(ncid, "time", "units", "day as %Y%m%d.%f");
00111   add_text_attribute(ncid, "lon", "long_name", "longitude");
00112   add_text_attribute(ncid, "lon", "units", "degrees_east");
00113   add_text_attribute(ncid, "lat", "long_name", "latitude");
00114   add_text_attribute(ncid, "lat", "units", "degrees_north");
00115   add_text_attribute(ncid, "lev", "long_name", "air_pressure");
00116   add_text_attribute(ncid, "lev", "units", "Pa");
00117   add_text_attribute(ncid, "T", "long_name", "Temperature");
00118   add_text_attribute(ncid, "T", "units", "K");
00119   add_text_attribute(ncid, "U", "long_name", "U velocity");
00120   add_text_attribute(ncid, "U", "units", "m s**-1");
00121   add_text_attribute(ncid, "V", "long_name", "V velocity");
00122   add_text_attribute(ncid, "V", "units", "m s**-1");
00123   add_text_attribute(ncid, "W", "long_name", "Vertical velocity");
00124   add_text_attribute(ncid, "W", "units", "Pa s**-1");
00125
00126   /* End definition... */
00127   NC(nc_enddef(ncid));
00128
00129   /* Set coordinates... */
00130   for (ix = 0; ix < nx; ix++)
00131     dataLon[ix] = 360.0 / nx * (double) ix;
00132   for (iy = 0; iy < ny; iy++)
00133     dataLat[iy] = 180.0 / (ny - 1) * (double) iy - 90;
00134   for (iz = 0; iz < nz; iz++)
00135     dataZ[iz] = 100. * P(LIN(0.0, z0, nz - 1.0, z1, iz));
00136
00137   /* Write coordinates... */
00138   NC(nc_put_var_double(ncid, timid, &t0));
00139   NC(nc_put_var_double(ncid, levid, dataZ));
00140   NC(nc_put_var_double(ncid, lonid, dataLon));
00141   NC(nc_put_var_double(ncid, latid, dataLat));
```

```
00142
00143   /* Create wind fields (Williamson et al., 1992)... */
00144   for (ix = 0; ix < nx; ix++)
00145     for (iy = 0; iy < ny; iy++)
00146       for (iz = 0; iz < nz; iz++) {
00147         idx = (iz * ny + iy) * nx + ix;
00148         dataU[idx] = (float) (LIN(0.0, u0, nz - 1.0, u1, iz)
00149                               * (cos(dataLat[iy] * M_PI / 180.0)
00150                                  * cos(alpha * M_PI / 180.0)
00151                                  + sin(dataLat[iy] * M_PI / 180.0)
00152                                  * cos(dataLon[ix] * M_PI / 180.0)
00153                                  * sin(alpha * M_PI / 180.0)));
00154         dataV[idx] = (float) (-LIN(0.0, u0, nz - 1.0, u1, iz)
00155                               * sin(dataLon[ix] * M_PI / 180.0)
00156                               * sin(alpha * M_PI / 180.0));
00157       }
00158
00159   /* Write wind data... */
00160   NC(nc_put_var_float(ncid, tid, dataT));
00161   NC(nc_put_var_float(ncid, uid, dataU));
00162   NC(nc_put_var_float(ncid, vid, dataV));
00163   NC(nc_put_var_float(ncid, wid, dataW));
00164
00165   /* Close file... */
00166   NC(nc_close(ncid));
00167
00168   return EXIT_SUCCESS;
00169 }
00170
00171 /*****************************************************************************/
00172
00173 void add_text_attribute(
00174   int ncid,
00175   char *varname,
00176   char *attrname,
00177   char *text) {
00178
00179   int varid;
00180
00181   NC(nc_inq_varid(ncid, varname, &varid));
00182   NC(nc_put_att_text(ncid, varid, attrname, strlen(text), text));
00183 }
```

Index