jd1jd2

1.0

Generated by Doxygen 1.8.3.1

Wed Jun 4 2014 15:33:07

Contents

Chapter 1

Main Page

C++ Classes for astronomical calculations

A C++ library

by Jean-Francois Le Borgne, Astronomer at *IRAP*, *Observatoire Midi-Pyrénées*, Toulouse, F EU

Adapted from c program skycalc by John Thorstensen, Dartmouth College, Hanover, NH USA. Version 6.1, 2005 which uses Jean Meeus' algorithms (**Astronomical Formulae For Calculators, pub. Willman-Bell.**, 1985)

`http://www.dartmouth.edu/~physics/people/faculty/thorstensen.html`

1.1 Introduction

Most comments in code and documentation are from John Thorstensen's code.

1.2 Installation

g++ -o astro observatory.cpp amzer.cpp astronomy.cpp sun.cpp moon.cpp planets.cpp main.cpp

1.3 Usage

./astro [-h] [-f filename]

./astro [-h] [-o name] [-d date] || [-j JD] [-c ra dec equinoxe -m]

-h : help

-v : verbose

-f filename : parameter file

-o name : observatory name

-d date : calendar date (ex. 2014 6 25 19 35 45.6)

-j JD : julian date (ex. 2456834.3165)

-c ra dec equinoxe : equatorial coordiantes of object

-m minimum height of object above horizon (>0)

-l limit acceptable distance to the Moon (degrees)

Examples: ./astro -o "La Silla"

./astro -o Toulouse

./astro -f parameters.txt<BR>

(-o -d -j -c) and -f options are incompatible. if both are used, -f superseed -o

-d and -j are are 2 options for the same parameter, the date. if both are used, -j superseed -d

1.4 how_to

How to use this document

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 amzer Class Reference

Class containing parameters and methods defining a time.

```
#include <amzer.h>
```

Public Member Functions

- amzer ()

  Constructor.
- ∼amzer ()

  Destructor.
- void JD2calendar ()

  Computes calendar date for a given julian date.
- void calendar2JD ()

  Computes julian date for a given calendar date.
- int day_of_week ()

  returns day of week for a jd, 0 = Mon, 6 = Sun.
- double day_of_year ()

  returns day of year
- string print_day (int d)

  Prints in a string day of week given number 0=Mon,6=Sun.
- string print_all (int UT, int secPrec)

  Prints time in a string: year, month, day, hour, minute, second.
- string print_calendar ()

  Prints in a string a year, month, day.
- string print_time (int UT, int secPrec)

  Prints time (h m s) in a string.
- double zone (int use_dst, double stdz, double jd, double jdb, double jde)

  Returns zone time offset when standard time zone is stdz,.

Public Attributes

- double JD

  Julian date.
- int day

day part
- int month

  month part
- int year

  year part
- int hour

  hour part
- int min

  minute part
- double second

  second part

3.1.1 Detailed Description

Class containing parameters and methods defining a time.

3.1.2 Member Function Documentation

3.1.2.1 string amzer::print_all (int *UT,* int *secPrec*)

Prints time in a string: year, month, day, hour, minute, second.

add a statement of whether time is "local" (UT=0) or "ut" (UT=1)

3.1.2.2 string amzer::print_time (int *UT,* int *secPrec*)

Prints time (h m s) in a string.

add a statement of whether time is "local" (UT=0) or "ut" (UT=1)

3.1.2.3 double amzer::zone (int *use_dst,* double *stdz,* double *jd,* double *jdb,* double *jde*)

Returns zone time offset when standard time zone is stdz,.

when daylight time begins (for the year) on jdb, and ends (for the year) on jde. Specifying a negative value of use-_dst reverses the logic for the Southern hemisphere; then DST is assumed for the Southern hemisphere summer (which is the end and beginning∗ of the year.

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/amzer.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/amzer.cpp

3.2 Astronomy Class Reference

Classes for astronomical calculations

```
#include <astronomy.h>
```

Classes

- struct coord

  Structure defining coordinates in sexigesimal format.

Public Member Functions

- Astronomy ()

    Constructor Gives default values to parameters.
- ∼Astronomy ()

    Destructor.
- void horizCoord (double dec, double ha, double lat, double ∗altitude, double ∗azimuth, double ∗parangle)

    Computes horizontal coordinates, altitude and azimuth,.
- double secant_z (double alt)

    Computes the secant of z,.
- double true_airmass (double secz)

    Returns the true airmass for a given secant z.
- double ha_alt (double dec, double lat, double alt)

    Returns hour angle at which object at dec is at altitude alt.
- double atan_circ (double x, double y)

    returns radian angle 0 to 2pi for coords x, y – get that quadrant right !!
- void min_max_alt (double lat, double decc, double ∗min_alt, double ∗max_alt)

    Computes minimum and maximum altitude for a given dec and latitude.
- double subtend (double ra1, double dec1, double ra2, double dec2)

    Angle subtended by two positions in the sky –.
- double lst (double jd, double longitude)

    Returns the local MEAN sidereal time (dec hrs)
- double bab_to_dec (coord bab)

    Converts a "babylonian" (sexigesimal) structure into double-precision floating point ("decimal") number.
- coord dec_to_bab (double deci)

    Function for converting decimal to babylonian hh mm ss.ss.
- string print_time (coord tme, int UT, int secPrec)

    Build a string with coordinate or time in sexigesimal units.
- string print_time (double t, int UT, int secPrec)

    This is an overloaded member function, provided for convenience.
- double adj_time (double x)

    Adjusts a time (decimal hours) to be between -12 and 12,.
- void eclrot (double jd, double ∗x, double ∗y, double ∗z)

    Rotates ecliptic rectangular coords x, y, z to equatorial (all assumed of date.)
- double circulo (double x)

    Assuming x is an angle in degrees, returns modulo 360 degrees.
- void geocent (double geolong, double geolat, double height, double ∗x_geo, double ∗y_geo, double ∗z_geo)

    Computes the geocentric coordinates from the geodetic.
- double etcorr (double jd)

    Given a julian date in 1900-2100, returns the correction delta t
    TDT - UT (after 1983 and before 1998)
    ET - UT (before 1983)
    an extrapolated guess (after 2005).
- float ztwilight (double alt)

    evaluates a polynomial expansion for the approximate brightening of the zenith in twilight
- void xyz_cel (double x, double y, double z, double ∗ra, double ∗dec)

    Converts cartesian coordinate (x,y,z) to right ascension and declination,.
- void aberrate (double epoch, double vec[3], int from_std)

    Corrects celestial unit vector for aberration due to earth's motion.
- void nutation_params (double date_epoch, double ∗del_psi, double ∗del_ep)

    Computes the nutation parameters delta psi and delta epsilon.

- void precess (double rin, double din, double std_epoch, double date_epoch, double ∗rout, double ∗dout, int just_precess, int from_std)

    General routine for precession and apparent place.
- void refract_corr (double ∗ha, double ∗dec, double latitude, double ∗size, int sense)

    Corrects local equatorial coordinates for refraction.
- double refract_size (double alt, double elev)

    Computes refraction size for a given altitude.
- double near_hor_refr (double app_alt, double pressure)

    Computes near horizon refraction.
- void galact (double ra, double dec, double epoch, double ∗glong, double ∗glat)

    Algorithm for 3-d Euler rotation into galactic.
- void eclipt (double ra, double dec, double epoch, double jd, double ∗curep, double ∗eclong, double ∗eclat)

    Converts ra and dec to ecliptic coords.
- void barycor (double jd, double ∗x, double ∗y, double ∗z, double ∗xdot, double ∗ydot, double ∗zdot)

    Corrects heliocentric position and velocity to the solar system barycenter.
- void helcor (double jd, double ra, double dec, double ha, double lat, double elevsea, double ∗tcor, double ∗vcor)

    Computes heliocentric correction for time and velocity.
- void lsrcor (double ra, double dec, double epoch, double ∗vcor)

    Computes the correction from bary/helio centric to local standard of rest.

3.2.1 Detailed Description

Classes for astronomical calculations

Adapted from c program skycalc by John Thorstensen, Dartmouth College. 'skycalc version: 6.1, 2005). JFLB 2014/04

3.2.2 Member Function Documentation

3.2.2.1 void Astronomy::aberrate (double *epoch,* double *vec[3],* int *from_std*)

Corrects celestial unit vector for aberration due to earth's motion.

Uses accurate sun position ... replace with crude one for more speed if needed.

epoch, decimal year ...

vec[]; celestial unit vector ...

from_std; 1 = apply aberration, -1 = take aberration out.

3.2.2.2 double Astronomy::adj_time (double *x*)

Adjusts a time (decimal hours) to be between -12 and 12,.

generally used for hour angles.

3.2.2.3 void Astronomy::barycor (double *jd,* double ∗ *x,* double ∗ *y,* double ∗ *z,* double ∗ *xdot,* double ∗ *ydot,* double ∗ *zdot*)

Corrects heliocentric position and velocity to the solar system barycenter.

This routine takes the position x,y,z and velocity xdot,ydot,zdot, assumed heliocentric, and corrects them to the solar system barycenter taking into account the nine major planets. Routine evolved by inserting planetary data (given above) into an earlier, very crude barycentric correction.

3.2.2.4 void Astronomy::eclipt (double *ra*, double *dec*, double *epoch*, double *jd*, double ∗ *curep*, double ∗ *eclong*, double ∗ *eclat*)

Converts ra and dec to ecliptic coords.

precesses to current epoch first (and hands current epoch back for printing.) ra in decimal hrs, other coords in dec. deg.

3.2.2.5 double Astronomy::etcorr (double *jd*)

Given a julian date in 1900-2100, returns the correction delta t

TDT - UT (after 1983 and before 1998)

ET - UT (before 1983)

an extrapolated guess (after 2005).

For dates in the past ($<=$ 2001 and after 1900) the value is linearly interpolated on 5-year intervals; for dates after the present, an extrapolation is used, because the true value of delta t cannot be predicted precisely. Note that TDT is essentially the modern version of ephemeris time with a slightly cleaner definition.

Where the algorithm shifts there will be a small ($<$ 0.1 sec) discontinuity. Also, the 5-year linear interpolation scheme can lead to errors as large as 0.5 seconds in some cases, though usually rather smaller. One seldom has actual UT to work with anyway, since the commonly-used UTC is tied to TAI within an integer number of seconds.

3.2.2.6 void Astronomy::galact (double *ra*, double *dec*, double *epoch*, double ∗ *glong*, double ∗ *glat*)

Algorithm for 3-d Euler rotation into galactic.

Perfectly rigorous, and with reasonably accurate input numbers derived from original IAU definition of galactic pole (12 49, +27.4, 1950) and zero of long (at PA 123 deg from pole.)

3.2.2.7 void Astronomy::geocent (double *geolong*, double *geolat*, double *height*, double ∗ *x_geo*, double ∗ *y_geo*, double ∗ *z_geo*)

Computes the geocentric coordinates from the geodetic.

(standard map-type) longitude, latitude, and height. These are assumed to be in decimal hours, decimal degrees, and meters respectively. Notation generally follows 1992 Astr Almanac, p. K11.

3.2.2.8 double Astronomy::ha_alt (double *dec*, double *lat*, double *alt*)

Returns hour angle at which object at dec is at altitude alt.

If object is never at this altitude, signals with special return values 1000 (always higher) and -1000 (always lower).

3.2.2.9 void Astronomy::helcor (double *jd*, double *ra*, double *dec*, double *ha*, double *lat*, double *elevsea*, double ∗ *tcor*, double ∗ *vcor*)

Computes heliocentric correction for time and velocity.

Finds heliocentric correction for given jd, ra, dec, ha, and lat. tcor is time correction in seconds, vcor velocity in km/s, to be added to the observed values. Input ra and dec assumed to be at current epoch.

3.2.2.10 void Astronomy::horizCoord (double *dec*, double *ha*, double *lat*, double ∗ *altitude*, double ∗ *azimuth*, double ∗ *parangle*)

Computes horizontal coordinates, altitude and azimuth,.

for declination, hour angle and observatory latitude (decimal degr, hr, degr). Also computes parallactic angle (decimal degr)

3.2.2.11 void Astronomy::lsrcor (double *ra,* double *dec,* double *epoch,* double ∗ *vcor*)

Computes the correction from bary/helio centric to local standard of rest.

i.e. (v wrt lsr) = (v helio) + vcor.

velocity of the sun is taken to be 13 km/s toward 1900 coords 18 0 0 +30 0 0

3.2.2.12 double Astronomy::lst (double *jd,* double *longitude*)

Returns the local MEAN sidereal time (dec hrs)

at julian date jd at west longitude long (decimal hours). Follows definitions in 1992 Astronomical Almanac, pp. B7 and L2. Expression for GMST at 0h ut referenced to Aoki et al, A&A 105, p.359, 1982. On workstations, accuracy (numerical only!) is about a millisecond in the 1990s.

3.2.2.13 double Astronomy::near_hor_refr (double *app_alt,* double *pressure*)

Computes near horizon refraction.

formula for near horizon, function-ized for iteration ... Almanac 1992, p. B62 – ignores temperature variation

3.2.2.14 void Astronomy::nutation_params (double *date_epoch,* double ∗ *del_psi,* double ∗ *del_ep*)

Computes the nutation parameters delta psi and delta epsilon.

at julian epoch (in years) using approximate formulae given by Jean Meeus, Astronomical Formulae for Calculators, Willman-Bell, 1985, pp. 69-70. Accuracy appears to be a few hundredths of an arcsec or better and numerics have been checked against his example. Nutation parameters are returned in radians.

3.2.2.15 void Astronomy::precess (double *rin,* double *din,* double *std_epoch,* double *date_epoch,* double ∗ *rout,* double ∗ *dout,* int *just_precess,* int *from_std*)

General routine for precession and apparent place.

Either transforms from current epoch (given by jd) to a standard epoch or back again, depending on value of the switch "from_std":

1 transforms from standard to current,

-1 goes the other way.

Optionally does apparent place including nutation and annual aberration (but neglecting diurnal aberration,parallax, proper motion, and GR deflection of light); switch for this is "just_precess",

1 does only precession,

0 includes other aberration & nutation.

Precession uses a matrix procedures as outlined in Taff's Computational Spherical Astronomy book. This is the so-called 'rigorous' method which should give very accurate answers all over the sky over an interval of several centuries. Naked eye accuracy holds to ancient times, too. Precession constants used are the new IAU1976 – the 'J2000' system. Nutation is incorporated into matrix formalism by constructing an approximate nutation matrix and taking a matrix product with precession matrix. Aberration is done by adding the vector velocity of the earth to the velocity of the light ray, not kosher relativistically, but empirically correct to a high order for the angle.

rin, din; input ra and dec

rout, dout; output

3.2.2.16 string Astronomy::print_time (coord *tme,* int *UT,* int *secPrec*)

Build a string with coordinate or time in sexigesimal units.

Add a statement of whether time is "local" (UT=0) or "ut" (UT=1) Any other value to add nothing.

3.2.2.17 string Astronomy::print_time (double *t,* int *UT,* int *secPrec*)

This is an overloaded member function, provided for convenience.

The same with input time/coordinate given in decimal unit:

3.2.2.18 void Astronomy::refract_corr (double ∗ *ha,* double ∗ *dec,* double *latitude,* double ∗ *size,* int *sense*)

Corrects local equatorial coordinates for refraction.

if sense == 1 , applies refraction to a true ha and dec;

if sense == -1, de-corrects already refracted coordinates.

The calculation is done by computing xyz coordinates in the horizon system, adding to the vertical component, renormalizing back to a unit vector, rotating to the polar system, and transforming back to ha and dec .... a long way around the barn, but completely general.

3.2.2.19 double Astronomy::refract_size (double *alt,* double *elev*)

Computes refraction size for a given altitude.

Almanac for 1992, p. B 62. Ignores variation in temperature and just assumes T = 20 celsius.

alt; altitude in degrees

elev; elevation in meters

3.2.2.20 double Astronomy::secant_z (double *alt*)

Computes the secant of z,.

assuming the object is not too low to the horizon; returns 100. if the object is low but above the horizon, -100. if the object is just below the horizon.

3.2.2.21 double Astronomy::subtend (double *ra1,* double *dec1,* double *ra2,* double *dec2*)

Angle subtended by two positions in the sky –.

Return value is in radians. Hybrid algorithm works down to zero separation except very near the poles.

3.2.2.22 double Astronomy::true_airmass (double *secz*)

Returns the true airmass for a given secant z.

The expression used is based on a tabulation of the mean KPNO atmosphere given by C. M. Snell & A. M. Heiser, 1968, PASP, 80, 336. They tabulated the airmass at 5 degr intervals from z = 60 to 85 degrees; John Thorstensen fitted the data with a fourth order poly for (secz - airmass) as a function of (secz - 1) using the IRAF curfit routine, then adjusted the zeroth order term to force (secz - airmass) to zero at z = 0. The poly fit is very close to the tabulated points (largest difference is 3.2e-4) and appears smooth. This 85-degree point is at secz = 11.47, so for secz $>$ 12 it just return secz - 1.5 ... about the largest offset properly determined.

3.2.2.23 void Astronomy::xyz_cel (double *x*, double *y*, double *z*, double ∗ *ra*, double ∗ *dec*)

Converts cartesian coordinate (x,y,z) to right ascension and declination,.

returned in decimal hours and decimal degrees.

3.2.2.24 float Astronomy::ztwilight (double *alt*)

evaluates a polynomial expansion for the approximate brightening of the zenith in twilight

(in magnitudes) compared to its value at full night, as function of altitude of the sun (in degrees). To get this expression the author looked in Meinel, A., & Meinel, M., "Sunsets, Twilight, & Evening Skies", Cambridge U. Press, 1983; there's a graph on p. 38 showing the decline of zenith twilight. The author read points off this graph and fit them with a polynomial; the author don't even know what band there data are for! Comparison with Ashburn, E. V. 1952, JGR, v.57, p.85 shows that this is a good fit to his B-band measurements.

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/astronomy.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/astronomy.cpp

3.3 Astronomy::coord Struct Reference

Structure defining coordinates in sexigesimal format.

```
#include <astronomy.h>
```

Public Attributes

- short sign
- double hh
- double mm
- double ss

3.3.1 Detailed Description

Structure defining coordinates in sexigesimal format.

3.3.2 Member Data Documentation

3.3.2.1 double Astronomy::coord::hh

hour or degree part (absolute value)

3.3.2.2 double Astronomy::coord::mm

minute part

3.3.2.3 short Astronomy::coord::sign

carry sign explicitly. Values: 1 or -1

3.3.2.4 double Astronomy::coord::ss

second part

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

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/astronomy.h

3.4 Moon Class Reference

Class containing methods concerning the Moon.

`#include <moon.h>`

Public Member Functions

- Moon ()

  Constructor.
- ∼Moon ()

  Destructor.
- void accumoon (double JD, double geolat, double lst, double elevsea, Moon ∗moon)

  More accurate (but more elaborate and slower) lunar ephemeris.
- void flmoon (int n, int nph, double ∗jdout)

  Gives jd (+- 2 min) of phase nph on lunation n.
- float lun_age (double jd, int ∗nlun)

  Compute age in days of moon since last new, and lunation of last new moon.
- string print_phase (double jd)

  Prints a verbal description of moon phase, given the julian date.
- double lunskybright (double alpha, double rho, double kzen, double altmoon, double alt, double moondist)

  Evaluates predicted LUNAR part of sky brightness,.
- double jd_moon_alt (double alt, double jdguess, double lat, double longit, double elevsea)

  Returns jd at which moon is at a given altitude,.

Public Attributes

- double geora

  Geocentric equatorial coordinates of the Moon: right ascension (decimal hours)
- double geodec

  Geocentric equatorial coordinates of the Moon: right declination (decimal degrees)
- double geodist

  Geocentric distance of the Moon (earth radii.
- double topora

  Topocentric equatorial coordinates of the Moon: right ascension (decimal hours)
- double topodec

  Topocentric equatorial coordinates of the Moon: right declination (decimal degrees)
- double topodist

  Topocentric distance of the Moon (earth radii)

3.4.1 Detailed Description

Class containing methods concerning the Moon.

3.4.2 Member Function Documentation

3.4.2.1 void Moon::accumoon (double *JD,* double *geolat,* double *lst,* double *elevsea,* Moon ∗ *moon*)

More accurate (but more elaborate and slower) lunar ephemeris.

from Jean Meeus' *Astronomical Formulae For Calculators*, pub. Willman-Bell. Includes all the terms given there. inputs units:geolat in decimal degrees, lst in decimal hours., elevsea in meters

3.4.2.2 void Moon::flmoon (int *n,* int *nph,* double ∗ *jdout*)

Gives jd (+- 2 min) of phase nph on lunation n.

This routine implements formulae found in Jean Meeus' *Astronomical Formulae for Calculators*, 2nd edition, Willman-Bell. A very usefulbook!! n, nph lunation and phase; nph = 0 new, 1 1st, 2 full, 3 last jdout jd of requested phase

3.4.2.3 double Moon::jd_moon_alt (double *alt,* double *jdguess,* double *lat,* double *longit,* double *elevsea*)

Returns jd at which moon is at a given altitude,.

given jdguess as a starting point. In current version uses high-precision moon – execution time does not seem to be excessive on modern hardware. If it's a problem on your machine, you can replace calls to 'accumoon' with 'lpmoon' and remove the 'elevsea' argument.

3.4.2.4 double Moon::lunskybright (double *alpha,* double *rho,* double *kzen,* double *altmoon,* double *alt,* double *moondist*)

Evaluates predicted LUNAR part of sky brightness,.

in V magnitudes per square arcsecond, following K. Krisciunas and B. E. Schaeffer (1991) PASP 103, 1033.

alpha = separation of sun and moon as seen from earth, converted internally to its supplement,

rho = separation of moon and object,

kzen = zenith extinction coefficient,

altmoon = altitude of moon above horizon,

alt = altitude of object above horizon

moondist = distance to moon, in earth radii

all are in decimal degrees.

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/moon.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/moon.cpp

3.5 Observatory Class Reference

Constructor: Set parameters for choosen observation site.

```
#include <observatory.h>
```

Public Member Functions

- **Observatory** (string siteName)
- ∼Observatory ()

  Destructor
- void find_dst_bounds (int yr, short stdz, int use_dst, double ∗jdb, double ∗jde)

  Finds jd's at which daylight savings time begins and ends.
- void sunSetRise (double JDmid, double rasun, double decsun, Observatory ∗observatory, double horiz, double jdb, double jde, double ∗jdsunset, double ∗jdsunrise, float ∗set_to_rise, double ∗jdcent)

  computes Sun set
- void twilights (double JDmid, double rasun, double decsun, Observatory ∗observatory, double jdb, double jde, Twilights ∗twi)

  Computes twilights.
- Twilights evening_morning_twilights (double alt, double hatwilight, double rasun, double JDmid, Observatory ∗observatory)

  Computes evening and morning twilights for given altiude of Sun below horizon.
- void print_twilights (Twilights twi, double jdb, double jde, double jdcent, Observatory ∗observatory)

  Print out twilights.
- void print_Moon_coordinates (double georamoon, double geodecmoon, double toporamoon, double topodecmoon)

  Print out Moon coordinates.
- void print_Sun_coordinates (double rasun, double decsun, double topora, double topodec, double sundist, double x, double y, double z)

  Print out Sun coordinates.
- int read_from_file (string filename, Observatory ∗observatory, amzer ∗date, double ∗ra, double ∗dec, string ∗equinoxe, double ∗min_height, double ∗Moon_lim)

  Read observatory parameters from ascii file.
- int **get_parameter_from_string** (string str, string param, string ∗value)
- int **get_parameter_from_string** (string str, string param, short ∗value)
- int **get_parameter_from_string** (string str, string param, int ∗value)
- int **get_parameter_from_string** (string str, string param, double ∗value)
- int **get_parameter_from_string** (string str, string param, amzer ∗date)

Public Attributes

- string name

  Site name.
- double longitude

  West Longitude (fractional hours)
- double latitude

  Latitude (fractional degrees)
- double elevsea

  True elevation above sea level (for absolute location)
- double elevation

  Elevation above the horizon (for rise/set calculations)
- double horiz

  Added zenith distance for rise/set due to elevation.
- short stdz

  standard time zone offset, hours
- string zone_name

  Name of time zone, e. g. Eastern.
- string zabr

  Single-character abbreviation of time zone.
- int use_dst

  Daylight saving time use.
- double magsky [9]

Sky magnitudes.

- double moon_phase

  Moon phase.

- double seeing

  Seeing.

- double airmass

  Airmass.

- double julianDay

  Julian day.

- double **horizon** [N_HORIZON]

3.5.1 Detailed Description

Constructor: Set parameters for choosen observation site.

Predefined sites:

default	Default observatory
Calern	Calern
Pic Du Midi	Pic Du Midi
Toulouse	Toulouse, Jolimont
Hakos	Hakos-Sternwarte, Namibia
Townsville	Townsville ,Australia
New Mexico	New Mexico Skies
Escalquens	Escalquens
Florida	Central Florida
OHP	OHP
Kitt Peak	Kitt Peak [MDM Obs.]
SALT/SAAO	SALT/SAAO, Sutherland, South Africa
La Silla	ESO, Cerro La Silla
Paranal	VLT, Cerro Paranal
Palomar	Palomar Observatory
Cerro Tololo	Cerro Tololo
Las Campanas	Las Campanas Observatory
Mount Hopkins	Mount Hopkins, Arizona
McDonald	McDonald Observatory
Siding Spring	Anglo-Australian Tel., Siding Spring
DAO	DAO, Victoria, BC
Mauna Kea	Mauna Kea, Hawaii
Lick	Lick Observatory
Roque	Roque de los Muchachos
SPM	San Pedro Martir

3.5.2 Member Function Documentation

3.5.2.1 void Observatory::find_dst_bounds (int *yr,* short *stdz,* int *use_dst,* double $*$ *jdb,* double $*$ *jde*)

Finds jd's at which daylight savings time begins and ends.

The parameter use_dst allows for a number of conventions, namely: 0 = don't use it at all (standard time all the time) 1 = use USA convention (1st Sun in April to last Sun in Oct after 1986; last Sun in April before) 2 = use Spanish convention (for Canary Islands) -1 = use Chilean convention (CTIO). -2 = Australian convention (for AAT). Negative numbers denote sites in the southern hemisphere, where jdb and jde are beginning and end of STANDARD time for the year. It's assumed that the time changes at 2AM local time; so when clock is set ahead, time jumps suddenly from 2 to 3, and when time is set back, the hour from 1 to 2 AM local time is repeated. This could be changed in code if need be.

3.5.3 Member Data Documentation

3.5.3.1 double Observatory::elevation

Elevation above the horizon (for rise/set calculations)

Elevations: There are two separate elevation parameters specified.

Variable elevation is the elevation above the horizon used only in rise/set calculations and adjusts rise/set times assuming the parameter is the elevation above flat surroundings (e. g., an ocean). (following skycalc, John Thorstensen, Dartmouth College.)

3.5.3.2 double Observatory::elevsea

True elevation above sea level (for absolute location)

Elevations: There are two separate elevation parameters specified.

Variable elevsea is the true elevation above sea level is used (together with an ellipsoidal earth figure) in determining the observatory's geocentric coordinates for use in the topocentric correction of the moon's position and in the calculation of the diurnal rotation part of the barycentric velocity correction. (following skycalc, John Thorstensen, Dartmouth College.)

3.5.3.3 double Observatory::magsky[9]

Sky magnitudes.

magnitude per square arc second in 9 visible/near infrared filters (default values are given in constructor)

3.5.3.4 int Observatory::use_dst

Daylight saving time use.

use_dst = 0 don't use day save time parameter

1 use USA convention

2 use Spanish convention

$<$ 0 Southern hemisphere (reserved, unimplimented)

(following skycalc, John Thorstensen, Dartmouth College.)

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/observatory.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/observatory.cpp

3.6 Planets::planetary_elements Struct Reference

Structure defining elements of planetary orbits.

```
#include <planets.h>
```

Public Attributes

- string name

  Name of the planet.
- double incl

  Inclination of the orbit.
- double Omega

  Longitude of the ascending node of the orbit.
- double omega

  Longitude of the perihelion of the orbit.
- double a

  Semimajor axis of the orbit.
- double daily

  daily motion
- double ecc

  Eccentricity of the orbit.
- double L_0

  Mean longitude.
- double mass

  Mass of the planet.

3.6.1 Detailed Description

Structure defining elements of planetary orbits.

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

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/planets.h

3.7 Planets Class Reference

Classe defining planets.

```
#include <planets.h>
```

Classes

- struct planetary_elements

  Structure defining elements of planetary orbits.

Public Member Functions

- Planets ()

  Constructor.
- ∼Planets ()

  Destructor.
- void comp_el (double jd)

  Computes and loads mean elements for planets.
- void planetxyz (int p, double jd, double ∗x, double ∗y, double ∗z)

  Produces ecliptic x,y,z coordinates for planet number 'p' at date jd.
- void planetvel (int p, double jd, double ∗vx, double ∗vy, double ∗vz)

  Numerically evaluates planet velocity by brute-force.
- string xyz2000 (double jd, double ∗x, double ∗y, double ∗z)

  Simply transforms a vector x, y, and z to 2000 coordinates and prints.
- void xyz2000xf (double jd, double ∗x, double ∗y, double ∗z)

  Simply transforms a vector x, y, and z to 2000 coordinates.
- void earthview (double ∗x, double ∗y, double ∗z, int i, double ∗ra, double ∗dec)

  Computes ra and dec of i-th planet as viewed from earth.

Public Attributes

- planetary_elements **el** [10]
- double **jd_el**

3.7.1 Detailed Description

Classe defining planets.

Adapted from c program skycalc by John Thorstensen, Dartmouth College. JFLB 2009/07

Planetary part. The intention of this is to compute low-precision planetary positions for general info and to inform user if observation might be interfered with by a planet – a rarity, but it happens. Also designed to make handy low-precision planet positions available for casual planning purposes. Do not try to point blindly right at the middle of a planetary disk with these routines!

3.7.2 Member Function Documentation

3.7.2.1 void Planets::earthview (double ∗ *x,* double ∗ *y,* double ∗ *z,* int *i,* double ∗ *ra,* double ∗ *dec*)

Computes ra and dec of i-th planet as viewed from earth.

Given computed planet positions for planets 1-10, computes ra and dec of i-th planet as viewed from earth (3rd);

3.7.2.2 void Planets::planetvel (int *p,* double *jd,* double ∗ *vx,* double ∗ *vy,* double ∗ *vz*)

Numerically evaluates planet velocity by brute-force.

numerical differentiation. Very unsophisticated algorithm.

3.7.2.3 void Planets::planetxyz (int *p,* double *jd,* double ∗ *x,* double ∗ *y,* double ∗ *z*)

Produces ecliptic x,y,z coordinates for planet number 'p' at date jd.

see 1992 Astronomical Almanac, p. E 4 for these formulae.

3.7.2.4 string Planets::xyz2000 (double *jd*, double ∗ *x*, double ∗ *y*, double ∗ *z*)

Simply transforms a vector x, y, and z to 2000 coordinates and prints.

for use in diagnostics. answer should be in ecliptic coordinates, in AU per day.

3.7.2.5 void Planets::xyz2000xf (double *jd*, double ∗ *x*, double ∗ *y*, double ∗ *z*)

Simply transforms a vector x, y, and z to 2000 coordinates.

and hands it back through pointers. For really transforming it!

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/planets.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/planets.cpp

3.8 Sun Class Reference

Class containing methods concerning the Sun.

```
#include <sun.h>
```

Public Member Functions

- Sun ()

  Constructor.
- ∼Sun ()

  Destructor.
- void lpsun (double jd, double ∗ra, double ∗dec)

  Low precision formulae for the sun,.
- void accusun (double jd, double lst, double geolat, Sun ∗sun)

  More accurate solar ephemeris.
- double jd_sun_alt (double alt, double jdguess, double lat, double longit)

  Returns jd at which sun is at a given altitude,.

Public Attributes

- double ra

  Geocentric equatorial coordinates of the Sun: right ascension (decimal hours)
- double dec

  Geocentric equatorial coordinates of the Sun: right declination (decimal degrees)
- double dist

  Geocentric distance of the Sun (earth radii.
- double topora

  Topocentric equatorial coordinates of the Sun: right ascension (decimal hours)
- double topodec

  Topocentric equatorial coordinates of the Sun: right declination (decimal degrees)
- double x

  x, y, and z are heliocentric equatorial coordinates of the EARTH, referred to mean equator and equinox of date.
- double **y**
- double **z**

3.8.1 Detailed Description

Class containing methods concerning the Sun.

3.8.2 Member Function Documentation

3.8.2.1 void Sun::accusun (double *jd*, double *lst*, double *geolat*, Sun $*$ *sun*)

More accurate solar ephemeris.

Implemenataion of Jean Meeus' more accurate solar ephemeris.

For ultimate use in helio correction! From Astronomical Formulae for Calculators, pp. 79 ff. This gives sun's position wrt *mean* equinox of date, not *apparent*. Accuracy is $<<$ 1 arcmin. Positions given are geocentric. Parallax due to observer's position on earth is ignored. This is up to 8 arcsec; Routine is usually a little better than that.

// – topocentric correction *is* included now. – //

Light travel time is apparently taken into account for the ra and dec, but the author don't know if aberration is and he don't know if distance is simlarly antedated.

x, y, and z are heliocentric equatorial coordinates of the EARTH, referred to mean equator and equinox of date.

3.8.2.2 double Sun::jd_sun_alt (double *alt*, double *jdguess*, double *lat*, double *longit*)

Returns jd at which sun is at a given altitude,.

given jdguess as a starting point.

Uses low-precision sun, which is plenty good enough.

3.8.2.3 void Sun::lpsun (double *jd*, double $*$ *ra*, double $*$ *dec*)

Low precision formulae for the sun,.

from Almanac p. C24 (1990)

ra and dec are returned as decimal hours and decimal degrees.

The documentation for this class was generated from the following files:

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/sun.h
- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/sun.cpp

3.9 Twilights Struct Reference

Observation site parameters.

```
#include <observatory.h>
```

Public Attributes

- double **jdetw**
- double **jdmtw**
- float **twi_to_twi**
- double **jdetw12**
- double **jdmtw12**
- float **twi_to_twi12**

3.9.1 Detailed Description

Observation site parameters.

Site parameters

Longitude

Latitude

Elevation

Sky magnitudes

Observing conditions parameters

Moon phase

Seeing

Airmass

Time parameters

Julian day

Twilights structure

Define twilight times, both nautical (Sun below 12° below horizon) and astronomical (Sun below 18° below horizon) jdetw: Jumian date of evening astronomical twilight jdmtw: Jumian date of morning astronomical twilight twi_to_twi: time interval in day between astronomical twilights (Full darkness) jdetw12: Jumian date of evening nautical twilight jdmtw12: Jumian date of morning nautical twilight twi_to_twi12i: time interval in day between nautical twilights

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

- /home/jean-francois/instruments/GFT/JD1JD2/jd1jd2_Cpp/observatory.h

Index