USNO Circular 180 Errata Rev. A

(F-29) Chapter 3 Sample Calculations

The sample Fortran code discussed in this chapter can be found in the file sample.f, which is part of the distribution; it is a main program that can be linked to the NOVAS subroutines and executed. If you do so, leave your executable named "a.out" or name it something other than "sample" to avoid confusion with the Unix "sample" command, e.g.

```
gfortran sample.f NOVAS_F3.0.o NOVAS_F3.0_solsys2.o jplsubs.o
```

(If you name your executable "sample," invoke it using "./sample".) The results, when SOLSYS version 2 is used, are given in the file sample.out. (If SOLSYS version 1 is used, the low-order digits of some of the results will be different, most noticeably for the Moon.) The program checkout.f, also part of the distribution, provides other examples of NOVAS subroutine calls.

NOVAS has a number of high-level subroutines that make it easy to obtain frequently needed information on the positions of celestial objects, and some of these will be described below. Before calling these subroutines, however, there are some setup calls that you might want to use. *Note that all floating-point arguments to NOVAS subroutines, input or output, are DOUBLE PRECISION.*

(C-33) Chapter 3 Sample Calculations

The sample C code discussed in this chapter can be found in the file **sample.c**, which is distributed along with NOVAS; it is a main function that can be linked to the NOVAS modules and executed. It requires *solarsystem* version 1, a working copy of the JPL software, and a DEnnn binary file. To use it,

a. compile and link files: **sample.c**, **novas.c**, **novascon.c**, **nutation.c**, **solsys1.c**, **eph_manager.c**, and **readeph0.c**.
b. name the resulting application **sample**.
c. verify that the JPL ephemeris file **JPLEPH**, or an alias to it, is available in the same directory as **sample**.
d. execute the **sample** application. On some Unix systems (e.g., Mac OS X), execute `./sample` to avoid confusion with the system `sample` command.

The results are given in the file **sample-usno.txt**, which was generated at the USNO with the JPL DE405 ephemeris. **sample.c** may be modified for use with *solarsystem* version 2; see the comments within that file for details.

NOVAS has a number of high-level functions that make obtaining frequently needed information on the positions of celestial objects easy; some of these are described below. In addition, Chapter 4 describes many of the functions and all of the structures used in these examples. The checkout programs used to validate a local installation (**checkout-stars.c**, **checkout-stars-full.c**, and **checkout-mp.c**) also provide other examples of NOVAS function calls.

Note that all floating-point arguments to NOVAS functions, input or output, are double precision floating-point values (type double).

USNO Circular 180 Errata Rev. B

(v)

The NOVAS 3.0 algorithms are based on a rigorous vector and matrix formulation and do not use any spherical trigonometry. Objects inside and outside the solar system are treated similarly. The position vectors formed and operated on by NOVAS place each object at its relevant distance (in astronomical units, or AU) from the solar system barycenter. Objects at unknown distance (parallax zero or undetermined) are placed on the "celestial sphere," herein defined to be at a radius of 1 gigaparsec (2.06×10^{14} AU).

(F-23)

NOVAS FORTRAN DISTRIBUTION FILES

File Name	**Contents**
NOVAS_3.0_Guide.pdf	User's Guide to NOVAS F3.0 (PDF)
NOVAS_F3.0.f	Standard NOVAS Fortran subroutines—Version 3.0
NOVAS_F3.0_alt.f	Alternative versions of some NOVAS subroutines
NOVAS_F3.0_solsys1.f	SOLSYS version 1—reads file SS_EPHEM.TXT
NOVAS_F3.0_solsys2.f	SOLSYS version 2—interface to JPL ephemeris software and
NOVAS_F3.0_solsys3.f	SOLSYS version 3—self-contained Earth and Sun
CIO_RA.TXT	Data file of CIO right ascensions as a function of time
SS_EPHEM.TXT	Solar system 1-day ephemeris file read by SOLSYS version 1
checkout.f	Main program for initial validation
checkout.out.1	Results file from checkout.f when SOLSYS version 1 is in use
checkout.out.3	Results file from checkout.f when SOLSYS version 3 is in use
CIO_file.f	Program to create binary direct-access data file from CIO_RA.TXT
sample.f	Main program with sample code from User's Guide Chapter 3
sample.out	Output from sample code (when SOLSYS version 2 is used)

USNO Circular 180 Errata Rev. B

(F-94)

VECTRS, CATRAN – 1/(sin(parallax)) now used to compute distance rather than 1/parallax; an inconsequential change, just to make the expression formally correct. Also, the "Doppler Factor", k, mentioned in the Hipparcos documentation and other papers, is now applied in computing the space-motion vector. The change in the units of proper motion and parallax is also implemented here. The computational distance used for objects of zero parallax has been increased to 1 Gpc (2.06 x 10^{14} AU).

(C-114)

transform_cat and *starvectors* now use $(\sin(\text{parallax}))^{-1}$ to compute distance, rather than parallax^{-1}; an inconsequential change that just makes the expression formally correct. Also, the "Doppler Factor," *k*, mentioned in the Hipparcos documentation and other papers, is now applied in computing the space-motion vector. The computational distance used for objects of zero parallax has been increased to 1 Gpc (2.06 x 10^{14} AU).

(F-8) Citing NOVAS

If you use NOVAS, please send us an e-mail[3] that outlines your application. This information helps justify further improvements to NOVAS. Your comments and suggestions are also welcome.

This user's guide constitutes Part I of USNO Circular 180 (the C guide is Part II), which may be cited as follows:

Kaplan, G., Bangert, J., Bartlett, J., Puatua, W., & Monet, A. 2009, *User's Guide to NOVAS 3.0*, USNO Circular 180[4] (Washington, DC: USNO).

In addition, we ask that you also direct your readers to the NOVAS website.[5]

The official reference for all previous versions of NOVAS is the 1990 software report announcing its release, which is Kaplan, G. (1990) "NOVAS: U. S. Naval Observatory," *Bull. AAS*, 22, 930.

(F-11) Chapter 1 Astronomical Background

At its highest level, NOVAS computes the precise positions of selected celestial objects at specified dates and times, as seen from a given location on or near the Earth. There are a number of coordinate systems in which such positions can be expressed. Dates and times are specified in several astronomical time scales, depending on the application. Users of NOVAS should have a basic knowledge of astronomical coordinate systems and time; terms like right ascension, declination, hour angle, ecliptic, equinox, precession, and sidereal time should be familiar. A number of texts on fundamental astronomy—for example, Green (1985), *Spherical Astronomy* (Cambridge University Press)—provide the essential concepts. For more technical descriptions of the latest international standards on reference systems, USNO Circular 179,[6] cited in the Introduction, can provide the background. Circular 179 documents the algorithms for many important calculations in NOVAS. Others are described in the Kaplan et al. (1989) Astron. J. paper mentioned in the Introduction or in the *Explanatory Supplement to the Astronomical Almanac*. In addition, two glossaries may be useful to NOVAS users: one published in *The Astronomical Almanac*,[7] and one compiled by the IAU Working Group on Nomenclature for Fundamental Astronomy.[8]

[3] http://www.usno.navy.mil/help/astronomy-help
[4] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180
[5] The current version of NOVAS may be obtained at http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas
[6] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[7] http://asa.usno.navy.mil/SecM/Section_M.html
[8] http://syrte.obspm.fr/iauWGnfa/NFA_Glossary.html

(F-18)

As previously mentioned, time is specified within NOVAS as Julian dates, which can be used for any of the above time scales. Julian dates are a simple count of days since noon on 4713 BC January 1, so that any date in recorded human history has a positive Julian date (JD). Over 2.4 million days have elapsed since JD 0, so that, for current dates, seven digits of precision are taken up just by the day count; if the JD is given by a standard double-precision floating-point number, about 9 digits are left to represent the time of day. Thus, a double-precision floating-point JD can represent time to a precision of about 0.1 ms. In those NOVAS routines where more precision is appropriate, the JD can be split between two input arguments, one that carries the high-order part of the JD (e.g., the day count) and the other that carries the low-order part (e.g., the fraction of a day). Note that for 0h (TT, UT1, or TDB), the fractional part of the Julian date is 0.5. An online calendar-date-to-Julian-date converter is available at the AA web site.[14] NOVAS has utility routines to convert between

(Footnote 14 corrected below.)

(F-20) 2.1 List of Distribution Files

The list of the 14 files in the Fortran distribution[21] is given in the table below. Except for the User's Guide, which is in Portable Document Format (PDF), the files are all plain-text ASCII. Files with a .f extension are Fortran source code.

NOVAS FORTRAN DISTRIBUTION FILES

File Name	Contents
NOVAS_3.0_Guide.pdf	User's Guide to NOVAS F3.0 (PDF)
NOVAS_F3.0.f	Standard NOVAS Fortran subroutines—Version 3.0
NOVAS_F3.0_alt.f	Alternative versions of some NOVAS subroutines
NOVAS_F3.0_solsys1.f	SOLSYS version 1—reads file SS_EPHEM.TXT
NOVAS_F3.0_solsys2.f	SOLSYS version 2—interface to JPL ephemeris software
NOVAS_F3.0_solsys3.f	SOLSYS version 3—self-contained Earth and Sun
CIO_RA.TXT	Data file of CIO right ascensions as a function of time
SS_EPHEM.TXT	Solar system 1-day ephemeris file read by SOLSYS version 1
checkout.f	Main program for initial validation
checkout.out.1	Results file from checkout.f when SOLSYS version 1 is in use
checkout.out.3	Results file from checkout.f when SOLSYS version 3 is in use
CIO_file.f	Program to create binary direct-access data file from CIO_RA.TXT
sample.f	Main program with sample code from User's Guide, Chapter 3
sample.out	Output from sample code when SOLSYS version 2 is used

[14] http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv
[21] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas/novas-fortran

(F-25)

If you do not have a JPL ephemeris already installed, you should try SOLSYS version 1, which reads and interpolates the supplied ephemeris file SS_EPHEM.TXT. This file is a formatted (ASCII) file containing the BCRS (barycentric) rectangular coordinates (in AU) of the Sun, eight planets, Pluto, and the Moon, with records at 1-day intervals. The supplied file contains the coordinates for the years 2000 to 2020, inclusive, taken from JPL's DE405 ephemeris. The interpolation errors of SOLSYS version 1 are negligible (a few meters or less) for all objects other than the Moon and Mercury; for the Moon the errors can amount to 450 meters (maximum angular error 0.3 arcsecond) and for Mercury the errors can reach 280 meters (maximum angular error 0.8 milliarcsecond). It would be straightforward to construct a file similar to SS_EPHEM.TXT from an ephemeris other than DE405—the specifications for the file are given in the description of SOLSYS version 1 in Chapter 4, and in the subroutine's prolog.

The command to link your program with NOVAS for this case would be

```
gfortran   myprog.f   NOVAS_F3.0.o   NOVAS_F3.0_solsys1.o
```

(F-101) D.1 Goal

These tests were designed to compare the transformation from the celestial to terrestrial reference systems, GCRS to ITRS, using the IAU 2000A/2006 models for precession and nutation. Specifically, we used NOVAS_F3.0g.f (the "g" beta version of NOVAS 3.0 Fortran) with `mode = 0`, which specifies the use of the CIO-based method at full accuracy. With SOFA, we used the example in the SOFA Cookbook titled "IAU 2006/2000A, CIO based, using classical angles." The goal was to verify that these (mostly) independent software systems produced results that agree at a level that is at least an order of magnitude better than the best observational results.

(F-101) D.2 Procedure

The test programs for NOVAS and SOFA follow as Addenda I and II, respectively. The input parameters, which were taken from the SOFA Cookbook, were as follows:

`UT1 = 2400000.5 + 54195.4999991658` days

`ΔT = 65.25607389` s (SOFA does not use ΔT; this is the difference between the TT and UT1 Julian dates in the SOFA example, expressed in seconds)

Polar coordinates: `XP = 0.0349282, YP = 0.4833163` arcseconds

CIP offsets: `DX = 0.1725, DY = –0.265` milliarcseconds

The SOFA subroutine iau_NUT06A includes small corrections to the nutation series arising from the P03 precession that are not used in the NOVAS calculations. The corrections amount to only a few microarcseconds (μas) for current dates.

(C-5)

(C-20)

As previously mentioned, time is specified within NOVAS as Julian dates, which can be used for any of the above time scales. Julian dates are a simple count of days since noon on 4713 BC January 1, so that any date in recorded human history has a positive JD. Over 2.4 million days have elapsed since JD 0, so that, for current dates, seven digits of precision are taken up just by the day count; if the JD is given by a standard double-precision floating-point number, about 9 digits are left to represent the time of day. Thus, a double-precision floating-point JD can represent time to a precision of about 0.1 ms. In those NOVAS functions where more precision is appropriate, the JD can be split between two input arguments, one that carries the high-order part of the JD (e.g., the day count) and the other that carries the low-order part (e.g., the fraction of a day). Note that for 0h (TT, UT1, or TDB), the fractional part of the Julian date is 0.5. An online calendar-date-to-Julian-date converter is available at the AA Department web site.[15] NOVAS has utility functions to convert between calendar date and Julian date and vice versa. They work for any time scale; that is, their input and output arguments should be considered to be just different ways of expressing the same instant within the same time scale. [*julian_date*, *cal_date*]

(Footnote 15 corrected below.)

(C-27)

Finally, an ASCII text file (**.txt**) containing the right ascensions of the CIO in the GCRS is supplied along with C source code (**.c**) to convert it to a binary direct-access file. The set-up and use of this file is discussed in Section 2.5.

File name	Description
CIO_RA.TXT	right ascensions of the CIO in the GCRS from 1700 to 2300
cio_file.c	main function that converts **CIO_RA.TXT** (ASCII) to **cio_ra.bin** (binary)

[15] http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv

(C-30) 2.5 Using an External CIO File

You have the option of using an external file of CIO right ascension values on the GCRS or of allowing NOVAS to calculate the true right ascension of the CIO (the arc on the instantaneous equator from the equinox to the CIO) using a series expansion. Section 5.3 explains how NOVAS handles these two situations. If you choose to use **CIO_RA.TXT**, which is provided with NOVAS, you must first convert it to a binary direct-access file as follows:

a. Copy **CIO_RA.TXT** and **cio_file.c** to a directory on your local system.
b. Compile **cio_file.c.**
c. Name the resulting application **cio_file**.
d. Run the **cio_file** application. If everything runs smoothly, you should get the following message

```
Results from program cio_file:

Input file identifier: CIO RA P03  @ 1.200d

182657 records read from the input file:
   First Julian date: 2341951.400000
   Last Julian date:  2561138.600000
   Data interval: 1.200000 days

First data point: 2341951.400000  -1.948328
Last data point:  2561138.600000  1.942125

Binary file cio_ra.bin created.
```

e. Verify the presence of **cio_ra.bin**, which should contain approximately 2.9 Mbytes.

(C-47)

place

```
short int place (double jd_tt, object *cel_object,
                 observer *location, double delta_t,
                 short int coord_sys, short int accuracy,

                 sky_pos *output)

   PURPOSE:
      This function computes the apparent direction of a star or solar
      system body at a specified time and in a specified coordinate
      system.

   REFERENCES:
      Kaplan, G. et al. (1989), Astronomical Journal 97, 1197-1210.
      Klioner, S. (2003), Astronomical Journal 125, 1580-1597.

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date for place.
      *cel_object (struct object)
         Specifies the celestial object of interest (structure defined
         in novas.h).
      *location (struct observer)
         Specifies the location of the observer (structure defined in
         novas.h).
      delta_t (double)
         Difference TT-UT1 at 'jd_tt', in seconds of time.
      coord_sys (short int)
         Code specifying coordinate system of the output position.
            = 0 ... GCRS or "local GCRS"
            = 1 ... true equator and equinox of date
            = 2 ... true equator and CIO of date
            = 3 ... astrometric coordinates, i.e., without light
                    deflection or aberration.
      accuracy (short int)
         Code specifying the relative accuracy of the output position.
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      *output (struct sky_pos)
         Output data specifying object's place on the sky at time
         'jd_tt', with respect to the specified output coordinate system
         (struct defined in novas.h).
```

(C-69)

topo_planet

```
short int topo_planet (double jd_tt, object *ss_body, double delta_t,
                       on_surface *position, short int accuracy,

                       double *ra, double *dec, double *dis)

   PURPOSE:
      Computes the topocentric place of a solar system body.

   REFERENCES:
      Kaplan, G. H. et. al. (1989). Astron. Journ. 97, 1197-1210.
      Explanatory Supplement to the Astronomical Almanac (1992),
         Chapter 3.

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date for topocentric place.
      *ss_body (struct object)
         Pointer to structure containing the body designation for the
         solar system body (defined in novas.h).
      delta_t (double)
         Difference TT-UT1 at 'jd_tt', in seconds of time.
      *position (struct on_surface)
         Specifies the position of the observer (structure defined in
         novas.h).
      accuracy (short int)
         Code specifying the relative accuracy of the output position.
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Topocentric right ascension in hours, referred to true equator
         and equinox of date.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date.
      *dis (double)
         True distance from Earth to planet at 'jd_tt' in AU.

   RETURNED
   VALUE:
      (short int)
           =  0 ... Everything OK.
           =  1 ... Invalid value of 'where' in structure 'location'.
           > 10 ... Error code from function 'place'.
```

(C-96)

solarsystem_hp, version 3
(File solsys3.c)

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Input Julian date ('tjd') out of range.
      2...Invalid value of 'body'.
      3...This version of 'solarsystem' not valid for use with
          NOVAS-C.
```

Discussion:
NOVAS does not provide a high-precision counterpart of *solarsystem version 3* that works without requiring an external data file. Thus, this version of *solarsystem_hp* is essentially a dummy function that acts according to the value of variable `action`, which is set within the function itself. If `action = 1` (the default), this function returns an error code of `3` indicating appropriately that the function does not provide high-precision position and velocity of the Earth and Sun. If `action = 2` (must be manually set), this function simply calls function *solarsystem (version 3)* and returns the low-precision position and velocity. An error code of `0` (no error) is also returned. This action may be useful for code testing purposes, but is neither appropriate nor recommended for normal use of NOVAS. Use alternate versions of *solarsystem_hp* (located in the various **solsysn.c** files, where n is an integer) when the highest precision is needed.

(C-107) A.6 New Features

The functions in this section have been added to NOVAS to increase functionality and convenience.

place is new general-purpose apparent place function. Section A.2 describes this function in greater detail.

equ2ecl converts RA and dec to ecliptic longitude and latitude. In addition, *equ2ecl_vec* and *ecl2equ_vec* convert vectors from an equatorial to an ecliptic basis and vice versa, respectively.

equ2gal converts ICRS RA and dec to galactic longitude and latitude.

gcrs2equ converts GCRS (geocentric ICRS) RA and dec to one of the equatorial systems of date.

make_object and *make_observer* are examples of a new set of functions which have been added to facilitate the construction of important data structures used in NOVAS.

(C-109) B.1 New Functions in NOVAS C3.0

cio_array is called from *cio_location*. It reads and returns a set of values of the GCRS right ascension of the CIO, near a given TDB Julian date, from an external, binary direct-access file.

cio_basis returns orthonormal basis vectors for the Celestial Intermediate Reference System with respect to the GCRS. It requires a prior call to *cio_location*.

cio_location returns the RA of the CIO at a given TDB Julian date, either with respect to the GCRS or with respect to the true equator and equinox of date.

cio_ra returns the value of the true RA of the CIO for a given TDB Julian date.

d_light evaluates the difference in light-time to a star between the solar system barycenter and the Earth.

ecl2equ_vec converts an ecliptic position vector to an equatorial position vector.

ee_ct evaluates a 34-term series for "complementary terms" in the equation of the equinoxes based on the work of Capitaine, Wallace, and McCarthy (2003).

equ2ecl converts equatorial RA and dec to ecliptic longitude and latitude.

equ2ecl_vec converts an equatorial position vector to an ecliptic position vector.

equ2gal converts ICRS RA and dec to galactic longitude and latitude.

era evaluates the ERA, θ.

frame_tie sets up the frame tie matrix and transforms a vector from the dynamical mean J2000.0 system to the ICRS, or vice versa. This function implements a first-order matrix with second-order corrections to the diagonal elements, patterned after the work by Hilton and Hohenkerk (2004). Given the smallness of the angles involved and their uncertainties, this approach is quite adequate.

Footnote deleted.

(C-110)

grav_vec calculates the gravitational deflection of light due to a solar system body. It is called by *grav_def*, a new function that replaces *sun_field*.

iau2000a evaluates the IAU 2000A nutation series (nutation only) based on IERS code[46] (Wallace 2003a).

iau2000b evaluates the IAU 2000B nutation series based on IERS code (Wallace 2003b).

ira_equinox returns the value of the Equation of the Origins, i.e., the right ascension of the equinox in the Celestial Intermediate Reference System from an analytical expression. The Equation of the Origins is the arc on the true equator of date from the CIO to the equinox, measured positively to the east.

[46] http://www.iers.org/MainDisp.csl?pid=38-15

(C-115) B.5 References

Capitaine, N. Wallace, P. T., & Chapront. J. 2003, *A&A*, 412, 567 (P03)

Capitaine, N., Wallace, P. T., & McCarthy, D. D. 2003, *A&A*, 406, 1135

Hilton, J. L. & Hohenkerk, C. Y. 2004, *A&A*, 413, 765

Kaplan, G. H. 2003, in *Proceeding of IAU Gen. Assembly XXV, Joint Discussion 16, The International Celestial Reference System, Maintenance and Future Realizations*, ed. R. Guame, D. McCarthy, & J. Souchay (Sydney: The Assembly) 196

Kaplan, G. H. 2005, *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models*, USNO Circular 179 (Washington, DC: USNO) http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179 (USNO Circular 179)

Lieske, J. H., Lederle, T., Fricke, W., & Morando, B. 1977, *A&A*, 1

Lieske, J. H. 1979, *A&A*, 282

Simon, J. L., Bretagnon, P., Chapront, J., Chapront-Touze, M., Francou, G., & Laskar, J. 1994, *A&A*, 282, 663

Standish, E. M. 1998, "JPL Planetary and Lunar Ephemerides, DE405/LE405," JPL IOM 312.F-98-048 (Pasadena, CA: JPL) http://ssd.jpl.nasa.gov/iau-comm4/relateds.html

Original Wallace, P. 2003a deleted.

Wallace, P. 2003a, NU2000A Subroutine, Subroutines for Chapter 5, in *IERS Conventions*, ed. D. McCarthy & G. Petit, IERS Tech. Not. 32 (Frankfurt: IERS) http://www.iers.org/MainDisp.csl?pid=38-15

Wallace, P. 2003b, NU2000B Subroutine, Subroutines for Chapter 5, in *IERS Conventions*, ed. D. McCarthy & G. Petit, IERS Tech. Not. 32 (Frankfurt: IERS) http://www.iers.org/MainDisp.csl?pid=38-15

(C-121) Appendix D A Comparison of SOFA and NOVAS

The Standards of Fundamental Astronomy (IAU 2009a; SOFA)[49] library is the official collection of approved software for positional astronomy, operating under the auspices of International Astronomical Union (IAU) Division 1 (Fundamental Astronomy). Both Fortran and C libraries are available. An international SOFA Reviewing Board manages the collection.

Generally, NOVAS is independent of SOFA although both software libraries include code that is similar to two IERS Fortran modules.[50] Function *iau2000a*, which evaluates the full 1,365-term IAU 2000A nutation series in NOVAS 3.0, is based on the IERS subroutine *NU2000A*. Function *ee_ct*, which evaluates the "complementary terms" in the equation of the equinoxes, is based on IERS function *EECT2000*. The corresponding modules in SOFA are, respectively, *iau_NUT00A* and *iau_EECT00* (Fortran) and *iauNut00a* and *iauEect00* (C).

The document *SOFA Tools for Earth Attitude*, also known as the "SOFA Cookbook", contains several Fortran examples of the transformation between terrestrial and celestial coordinate systems. This appendix examines how one of those examples plays out in the C editions of both NOVAS and SOFA.

D.1 Goal

These tests were designed to compare the transformation from GCRS to ITRS using the IAU 2000A/2006 models for precession and nutation. We compared the CIO-based method in NOVAS C3.0 at full accuracy with a C translation of the example in the SOFA Cookbook titled "IAU 2006/2000A, CIO based, using classical angles." The goal was to verify that the NOVAS libraries, which are (mostly) independent of SOFA, produced results that agree with their SOFA counterparts at a level that is at least an order of magnitude better than the best observational results.

D.2 Procedure

The comparison of the C editions of NOVAS and SOFA was based on an earlier comparison of the Fortran editions, which is described in Appendix D of the *User's Guide to NOVAS Version F3.0*[51] (Fortran section of this circular). The C test functions were basically a line-for-line transliteration of the Fortran **terceltest.f** and **SOFA-TEST.f** using the NOVAS and SOFA C functions, respectively. *User's Guide to NOVAS Version F3.0* includes complete code for the full text of **terceltest.f** and **SOFA-TEST.f.**

[49] http://www.iausofa.org/index.html
[50] http://tai.bipm.org/iers/conv2003/conv2003_c5.html
[51] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180

Function *terceltest* is provided in the addendum; it was run using the following input parameters:

- Universal Time: `UT1 = 2400000.5 + 54195.4999991658` days (Julian date)
  The UT1 value is divided into two parts, i.e., two separate arguments, because of the large number of significant digits needed for precise results. The best agreement between NOVAS and SOFA was obtained when UT1 was split in the exactly the same place. In the Fortran tests, splitting the date differently produced differences of about 3 microarcseconds.
- Difference between TT and UT1: `ΔT = 65.25607389` s
  SOFA does not use ΔT; this value is the difference between the TT and UT1 Julian dates in the SOFA example, expressed in seconds.
- Polar coordinates: `XP = 0.0349282`, `YP = 0.4833163` arcsec
- CIP offsets: `DX = 0.1725`, `DY = -0.265` arcsec

The SOFA function *iauNut06a* includes small corrections to the nutation series arising from the P03 precession that are not used in the NOVAS calculations. The corrections amount to only a few microarcseconds for current dates.

NOVAS does not directly produce an overall GCRS-to-ITRS rotation matrix as SOFA does. The NOVAS rotation matrix was constructed simply by passing the three vectors, (1,0,0), (0,1,0), and (0,0,1), in succession through function *ter2cel*.

A series of tests was done, with and without corrections for polar motion, precession and nutation, and the P03 correction in SOFA. The output of the C functions was compared with output from the corresponding Fortran programs. Both the Fortran and C computations were executed on a 32-bit Intel Apple Macintosh system running the Leopard (Mac OS X 10.5) operating system.

D.3 Results

Table D1 shows that the latest C releases of NOVAS and SOFA agree at the sub-microarcsecond level in the transformation between the celestial and terrestrial reference systems when the same Earth orientation parameters and conventions are used. In this case, including the P03 corrections in the SOFA nutation adds a discrepancy on the order of 1.4 μas. Inclusion of the CIP offsets and polar motion does not significantly add to the differences in the two formulations, as long as the parameters used are identical in the two cases. Use of the external **CIO_RA** file in the NOVAS calculation adds about 0.05 μas to the difference for the above case, while using the equinox method for the NOVAS computations does not have a significant effect on the results.

The results presented in Table D1 were obtained by computing the GCRS to ITRS transformations for the single time discussed in the SOFA Cookbook. Therefore, the values should be typical.

Table D1. Comparison of NOVAS C3.0 and SOFA C

Corrections Applied			Other Options		
Polar Motion	CIP Offsets	P03 Terms	External **CIO_RA**	Equinox method	Difference (μas)
No	No	No	No	No	0.25814
No	No	Yes	No	No	1.6752
No	Yes	Yes	No	No	1.6728
Yes	Yes	Yes	No	No	1.6735
Yes	Yes	No	No	No	0.28679
Yes	Yes	No	Yes	No	0.34369
Yes	Yes	No	No	Yes	0.28644

D.4 References

Capitaine, N., Wallace, P. T., Chapront, J. 2003, A&A, 412, 567 (P03)

Kaplan, G., Bartlett, J., Monet, A., Bangert, J., & Puatua, W., 2009, *User's Guide to NOVAS F3.0* (Washington, DC: USNO) http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180

IAU. 2009a, Standards of Fundamental Astronomy, (SOFA) http://www.iausofa.org/

IAU. 2009b, *SOFA Tools for Earth Attitude*, Software version 4, Document revision 1.1 (SOFA Cookbook) http://www.iausofa.org/sofa_pn.pdf

IERS. 2003, Conventions 2003: Chapter 5 Transformation Between the Celestial and Terrestrial Systems (Frankfurt, Germany: BKG) http://tai.bipm.org/iers/conv2003/conv2003_c5.html

Monet, A., Kaplan, G., & Harris, W. *Testing Coordinate Frame Transformations NOVAS vs SOFA*, USNO/AA Technical Note 2010-04 (Washington, DC: USNO)

William Harris
Alice Monet
George Kaplan
14 July 2010

D.5 Addendum: NOVAS Code

```
void terceltest ()
{

/* Transform vectors from ITRS to GCRS */

   double tjdh, tjdl, xp, yp, delt = 65.25607389, vec1[3], vec2[3], tjd,
      dx, dy, mobl, tobl, ee;

   short num = 0;

   FILE *In_Data = NULL;

   dx = +0.1750;
   dy = -0.2259;

/* Open the input file of Julian dates,CIO coords,ITRS vector */

   In_Data = fopen ("tercel-test-input.dat","r");

   while (!feof(In_Data))
   {
      fscanf(In_Data,"%hi%lf%lf%lf%lf%lf%lf%lf",&num,&tjdh,&tjdl,
         &xp,&yp,&vec1[0],&vec1[1],&vec1[2]);

/*  Set transformation method, accuracy level, and ut1-utc. */

      tjd = tjdh + tjdl;

/*      celpol (tjd,2,dx,dy);*/
      e_tilt (tjd,0, &mobl,&tobl,&ee,&dx,&dy);


/*  Rotate vec1 from ITRS to GCRS = vec2  */

      ter2cel (tjdh,tjdl,delt,1,0,0,xp,yp,vec1, vec2);
      printf ("%i    %20.17f    %20.17f    "
              %20.17f\n",num,vec2[0],vec2[1],vec2[2]);
   }

   fclose (In_Data);

}
```

(Return to Table of Contents)